NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

SYSTEMS ENGINEERING CAPSTONE REPORT

A DIGITAL ENGINEERING CASE STUDY OF AN UNMANNED UNDERWATER VEHICLE

by

Gregory L. Barr, Nolan J. Bunker, Oscar I. Cedillos,
Tylong Chheung, William Flores, and Adam C. Ortega

December 2021

Advisor: Mark M. Rhoades
Second Reader: Douglas L. Van Bossuyt

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2021	**3. REPORT TYPE AND DATES COVERED** Systems Engineering Capstone Report

4. TITLE AND SUBTITLE
A DIGITAL ENGINEERING CASE STUDY OF AN UNMANNED UNDERWATER VEHICLE

5. FUNDING NUMBERS

6. AUTHOR(S) Gregory L. Barr, Nolan J. Bunker, Oscar I. Cedillos, Tylong Chheung, William Flores, and Adam C. Ortega

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Team Icarus created a digital engineering case study based on an unmanned underwater vehicle (UUV) to provide a robust view of developing an architecture using Cameo Systems Modeler by executing the MagicGrid architecture development methodology. The case study includes connecting this architecture model to directly drive several engineering analysis tools (Excel, MATLAB/Simulink, a Computer Aided Design tool) through middle-ware software (ModelCenter MBSE). The design was refined through a design of experiments and is visualized through software tools (ModelCenter Explore). This case study is provided to Naval Surface Warfare Center–Port Hueneme Division (NSWC PHD) to be a supplement to the training of systems engineers and systems logisticians to fill in the gaps of existing trainings. This case study is also provided to Naval Postgraduate School to supplement the education of current and future students on architecture development and digital engineering.

14. SUBJECT TERMS digital engineering, UUV	**15. NUMBER OF PAGES** 367
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A DIGITAL ENGINEERING CASE STUDY OF AN UNMANNED UNDERWATER VEHICLE

Gregory L. Barr, Nolan J. Bunker, Oscar I. Cedillos,
Tylong Chheung, William Flores, and Adam C. Ortega

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2021

Lead Editor: Gregory L. Barr

Reviewed by:
Mark M. Rhoades
Advisor

Douglas L. Van Bossuyt
Second Reader

Accepted by:
Oleg A. Yakimenko
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Team Icarus created a digital engineering case study based on an unmanned underwater vehicle (UUV) to provide a robust view of developing an architecture using Cameo Systems Modeler by executing the MagicGrid architecture development methodology. The case study includes connecting this architecture model to directly drive several engineering analysis tools (Excel, MATLAB/Simulink, a Computer Aided Design tool) through middle-ware software (ModelCenter MBSE). The design was refined through a design of experiments and is visualized through software tools (ModelCenter Explore). This case study is provided to Naval Surface Warfare Center–Port Hueneme Division (NSWC PHD) to be a supplement to the training of systems engineers and systems logisticians to fill in the gaps of existing trainings. This case study is also provided to Naval Postgraduate School to supplement the education of current and future students on architecture development and digital engineering.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
- A. BACKGROUND....2
- B. STAKEHOLDERS....3
- C. PROJECT OBJECTIVES....4
- D. PROJECT SCOPE....4
- E. ASSUMPTIONS AND CONSTRAINTS....5
- F. TECHNICAL APPROACH....5
- G. REPORT STRUCTURE....9

II. BACKGROUND....11
- A. CURRENT TRAINING INITIATIVES....11
- B. DIGITAL ENGINEERING – BASIC SKILLS NEEDED....12
- C. GAP ANALYSIS BETWEEN CURRENT TRAINING AND DIGITAL ENGINEERING BASIC SKILLS....12
- D. CHAPTER SUMMARY....13

III. UUV NEEDS ANALYSIS AND SCENARIOS....15
- A. EFFECTIVE NEED....15
- B. CAPABILITY GAP....15
- C. OPERATING ENVIRONMENT....16
- D. UUV MISSION SETS....16
- E. UUV PERFORMANCE REQUIREMENTS....17
- F. CHAPTER SUMMARY....19

IV. ARCHITECTURAL DEVELOPMENT....21
- A. MODIFIED MAGICGRID PROCESS EXPLANATION....21
- B. PROBLEM DOMAIN....21
- C. BLACK BOX....23
- D. WHITE BOX....24
- E. SOLUTION DOMAIN....26
- F. IMPLEMENTATION DOMAIN....30
- G. INCORPORATING DE ELEMENTS TO THE ARCHITECTURE MODEL....31
- H. COMBINING MODELCENTER AND CAMEO....37
  1. Architecture Model and Excel Analysis Integration....37
  2. Automating Requirement and Constraint Verification....40
  3. Optimizing the Design....41

I. SCOPE OF INTEGRATION EFFORTS 41
J. CHAPTER SUMMARY 42

V. ORGANIZATION OF TUTORIALS 43
A. CAMEO MODELING AND VIDEO RECORDING 43
B. EDITING AND FINISHING TUTORIALS 44
C. PACKAGING THE TUTORIALS 44
D. TESTING AND EDITING FOR CLARITY 45
E. CHAPTER SUMMARY 45

VI. CONCLUSIONS AND RECOMMENDATIONS 47
A. REVIEW OF OBJECTIVES 47
B. RECOMMENDATIONS FOR FUTURE WORK 48

APPENDIX A. TUTORIALS 49

APPENDIX B. QUESTIONS FOR STAKEHOLDERS 345

LIST OF REFERENCES 347

INITIAL DISTRIBUTION LIST 349

LIST OF FIGURES

Figure 1. The Unified Process. Source: Pressman and Maxim (2019, 32). 6

Figure 2. Team Icarus Modified UP.....8

Figure 3. MagicGrid Architectural Development Process. Source: Aleksandraviciene and Morkevicius (2018, 11). 9

Figure 4. MagicGrid Problem Domain. Adapted from Aleksandraviciene and Morkevicius (2018, 11).....22

Figure 5. MagicGrid Black Box. Adapted from Aleksandraviciene and Morkevicius (2018, 11).....23

Figure 6. MagicGrid White Box. Adapted from Aleksandraviciene and Morkevicius (2018, 11).....24

Figure 7. MagicGrid Black and White Box Analysis. Source: Aleksandraviciene and Morkevicius (2018, 50). 25

Figure 8. MagicGrid Solution Domain. Adapted from Aleksandraviciene and Morkevicius (2018, 11).....26

Figure 9. MagicGrid Problem Domain Decomposition. Source: Aleksandraviciene and Morkevicius (2018, 79). 28

Figure 10. MagicGrid Solution Domain Decomposition. Source: Aleksandraviciene and Morkevicius (2018, 81). 29

Figure 11. MagicGrid Implementation Domain. Adapted from Aleksandraviciene and Morkevicius (2018, 11). 31

Figure 12. Buoyancy Input Parameters Excel File.....33

Figure 13. Buoyancy Calculation Excel File.....33

Figure 14. Image of Analysis Server Interface.....34

Figure 15. ModelCenter UUV Buoyancy Model 36

Figure 16. Cameo UUV Model 39

Figure 17. ModelCenter and Excel Files Link 40

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Capstone Stakeholders 3

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BDA	battle damage assessment
BDD	Block Definition Diagram
CONOPS	Concept of Operations
DAGR	Defense Advanced GPS Receiver
DE	Digital Engineering
DOD	Department of Defense
DON	Department of the Navy
GPS	Global Positioning System
MBSE	Model-based Systems Engineering
MOEs	measures of effectiveness
NPS	Naval Postgraduate School
NSWC PHD	Naval Surface Warfare Center–Port Hueneme Division
PMA	post-mission analysis
RHIB	rubber hull inflatable boat
SE	systems engineering
SoI	System of Interest
UP	Unified Process
VCCS	Vehicle Climate Control System

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Digital engineering (DE) and model-based systems engineering (MBSE) have become the industry standard within the Department of Defense (DOD) and the Department of the Navy (DON) in recent years. Digital engineering is defined as "an integrated digital approach that uses authoritative sources of systems data and models as a continuum across disciplines to support life cycle activities from concept through disposal" (Shepard and Scherb 2020). Many naval organizations have adapted to the digital engineering approach and have started to provide training programs that focus on various components of digital engineering and the tools that help support these processes.

Several of these training programs are strictly focused on one specific component of the digital engineering process. While many of these training programs focus in depth on a particular component, they only show a limited and compartmentalized view of the digital engineering or architectural development method. This lack of fluidity between the different components exposes a gap in teaching digital engineering. The training courses offered do not show how architectural development and engineering analysis tools interact between each other nor how they work together towards a successful digital engineering process. Therefore, the students are limited in conceptualizing the entire architecture development method and exploring any digital engineering techniques that will optimize their architecture designs.

The primary objective for this capstone is to perform a case study of a theoretical unmanned underwater vehicle (UUV) using MBSE with a digital engineering implementation to supplement current training and education. This is completed through three deliverables: an example of a theoretical UUV digital system architecture, written and visual tutorials of the MagicGrid architecture development methodology including the use of engineering analysis software tools, and this final report on the overall case study.

The theoretical UUV is a system concept that is to be transformed into a digital architecture model using Cameo Systems Modeler. Using the MagicGrid architecture development methodology, the UUV architecture model can be connected and analyzed by

different engineering analysis tools [i.e., Excel and MATLAB/Simulink (a computer-aided design tool)], in addition to Model Center MBSE. To illustrate the development process, text-based and video-based tutorials are completed alongside the architecture. Lastly, a design of experiments is performed on the architecture model to test system capabilities and refine the design.

Included with these tutorials is a model template to serve as a supplement to current training and education, providing more depth in MBSE and digital engineering tools, techniques, and processes. This satisfies the stakeholders' objective and requirements, and the final deliverable can also be used to re-evaluate current model-based program execution processes.

Reference

Shepard, David, and Julia Scherb. 2020. "What Is Digital Engineering and How Is It Related to DevSecOps?," *Software Engineering Institute*(blog). November 16, 2021. https://insights.sei.cmu.edu/blog/what-digital-engineering-and-how-it-related-devsecops/.

I. INTRODUCTION

Digital Engineering has become the industry standard within the Department of Defense (DOD) and the Department of the Navy (DON). Many of the agencies and organizations within the DON have migrated to digital engineering and Model-Based Systems Engineering (MBSE) in the recent years. Digital engineering is defined as "an integrated digital approach that uses authoritative sources of systems data and models as a continuum across disciplines to support life cycle activities from concept through disposal" (Shepard and Scherb 2020). The DOD's and the DON's decision on migrating to digital engineering has been very beneficial to government agencies such as Naval Surface Warfare Center Port Hueneme Division (NSWC PHD) in bringing modeling and technical documents to the modern digital world. Since June 2018, the DOD moved toward having digital engineering applied throughout the different branches of the U.S. military. Many naval organizations have adapted to the digital engineering approach and have started to provide training programs that focus on various components of digital engineering and the tools that help support these processes.

Several of these training programs are strictly focused on one specific component of the digital engineering process. For example, the first training session in the program would be an introduction to SysML Lite, but the following training session would be an introduction to Cameo Systems Modeler. This type of training program focuses in depth on a particular component but does not fully explain how that component integrates together with other components of the digital engineering process. The training programs only show a limited and compartmentalized view of the architectural development method. This lack of fluidity between the different components is the digital engineering gap. Similar training on specific engineering analysis tools is also offered at higher education institutions for students to learn and develop hands-on experience. Those not in higher education would be able to independently learn the engineering analysis tool by purchasing the software license and textbooks, but this would not be cost effective.

The trainings offered today do not show how architectural development and engineering analysis tools interact between each other nor how they work together towards a

successful digital engineering process. Therefore, the students are limited in conceptualizing the entire architecture development method and exploring any digital engineering techniques that will optimize their architecture designs.

A. BACKGROUND

In June 2018, the DOD introduced its Digital Engineering Strategy that was going to be used to digitally transform the planning process, development process, and the implementation process across the U.S. military (Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018, 2). Digital Engineering brings traditional approaches of systems engineering to the rapid technological advances of the modern digital world. The DOD's Digital Engineering Strategy outlines five goals that will be the backbone of digital engineering within the U.S. military branches:

1. Formalize the development, integration, and use of models to inform enterprise and program decision making.
2. Provide an enduring, authoritative source of truth.
3. Incorporate technological innovation to improve the engineering practice.
4. Establish a supporting infrastructure and environment to perform activities, collaborate, and communicate across stakeholders.
5. Transform the culture and workforce to adopt and support digital engineering across the life cycle. (Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018, 4).

These five goals work together to define what is necessary to move forward in implementing digital engineering concepts across the military and how the U.S. military will proceed. The DON views digital engineering as a strategic plan that is necessary to better enhance the engineering acquisition concepts and practices applied across the U.S. Navy and Naval organizations. To improve the effectiveness of the warfighter's capabilities, the DON has implemented the use of digital models and data derived from MBSE. Digital Engineering is necessary in managing complexity and capabilities of the warfighter systems. Digital

models provide data that can be analyzed and tested to create a more complex system to test out the capabilities of the warfighter.

B. STAKEHOLDERS

There are multiple groups who have a stake in how this capstone project is directed and shaped. A list of stakeholders and a description of their roles can be identified in Table 1. The primary stakeholders are determined as command representatives at NSWC PHD and the Naval Postgraduate School (NPS), as they have proposed this topic to aid in the training and development of systems engineers and systems logisticians. They provide direct support and expertise to facilitate the successful completion of this capstone project. Other stakeholders consist of subject matter experts and the end users who provide insight and feedback through the course of the project. They also have an influence on the project deliverables for understanding and compatibility. Through the identification of stakeholders, problem definition and requirements are refined.

Table 1. Capstone Stakeholders

Role	Members	Description
Advisors	Mark Rhoades Dr. Van Bossuyt	Project advisors are responsible for advising the team. They are responsible for guiding the team to ensure the project stays on track, provide recommendations, perform internal reviews of documentation, and answer any questions.
Sponsor	Alfredo Gabertan PHE Division NSWC PHD Mark Rhoades NPS SE Dept Lead for Educational Technology	The sponsor is responsible for guiding the team with reducing the scope of the project. Additionally, the sponsor is there to provide expertise on the subject and address questions from the team regarding the project.
Subject Matter Experts	Bill Gieri NSWC PHD c/200 Engineering & Test Office Kanoko Esheim Coralyn Akers NSWC PHD	The SMEs will provide insight into the most efficient techniques and provide a quality check on the content and model(s) developed as part of the case study.
End Users	Systems Engineers/Systems Logisticians	The end users will provide feedback of understanding and compatibility with core deliverables.

C. PROJECT OBJECTIVES

The primary objective for this capstone is to perform a case study of an unmanned underwater vehicle (UUV) using MBSE with a digital engineering implementation to supplement current training and education. This is completed through three deliverables: an example UUV digital system architecture, written and visual tutorials of the MagicGrid architecture development methodology including the use of engineering analysis software tools, and this final report on the overall case study.

The UUV is a system concept that is to be transformed into a digital architecture model using Cameo Systems Modeler. Using the MagicGrid architecture development methodology, the UUV architecture model can be connected and analyzed by different engineering analysis tools (Excel, MATLAB/Simulink, a Computer Aided Design tool), in addition to Model Center MBSE. Text-based and video-based tutorials are completed alongside the architecture to illustrate the development process. Lastly, a design of experiments is performed on the architecture model to test system capabilities. Through these deliverables, a final report is written to explain the details, the process, and recommendations for future work of the case study.

The final deliverable is a model template with documentation and tutorials to serve as a supplement to current training and education, providing more depth in MBSE and digital engineering tools, techniques, and processes. This satisfies the stakeholders' objective and requirements, and the final deliverable can also be used to re-evaluate current model-based program execution processes.

D. PROJECT SCOPE

The scope of the capstone is defined by the UUV system concept to demonstrate implementation of digital engineering in MBSE. This research and development of the capstone is also confined to a nine-month timeframe, in which progress reviews and deliverables are provided. Due to this time constraint, only a single case study is provided to explain how to design and develop a working architecture. Demonstrations of the integration of engineering analysis tools will be representative of integration of a common analysis tool (e.g., SolidWorks) and illustrate how it is done. Not all the required

engineering models that would be required to satisfy a fully functional, physics-based UUV system will be included in the system model.

E. ASSUMPTIONS AND CONSTRAINTS

The first assumptions are made about the end-users of our deliverables. The deliverables of this case study are written for beginning and intermediate systems engineering and system logistician students. In this case, we define beginning students as those who have little to no training in either SysML or Cameo Systems Modeler. Intermediate students are defined as having completed some training in either SysML and/or Cameo Systems Modeler but have not yet received training on the integration of engineering analysis tools with the Cameo Architecture model. The next assumption is that the case study will be used as a supplemental tool to current training and not as a replacement. The case study implements a modified MagicGrid architecture development method but also demonstrates other methodologies that can apply. These assumptions do not change the scope of the project but are used to help shape its progression.

The constraints of this capstone project are the limitations under which the project will be performed. The primary constraints fall into the categories of scope, schedule, and resources. As mentioned previously, the objective of the capstone project is to perform a case study on a concept UUV system with an emphasis on MBSE and digital engineering techniques. The research and analysis are performed within a nine-month period as part of the NPS Systems Engineering master's degree program by six students. There is a limited amount of time to complete each deliverable, in which milestone dates must be met for working schedule preservation. The tutorial deliverable must be compatible with NSWC PHD, NPS, and other desired locations. It is constructed using the software that is available to the team through NPS.

F. TECHNICAL APPROACH

To accomplish this capstone project, the team employs a process similar to the Unified Process (UP) (Pressman and Maxim 2019, 55), combined with an Agile development strategy. The modified UP model with Agile development uses three sprints to complete the architecture development, tutorial, and analysis of capabilities through

engineering software tools. To successfully meet requirements, frequent stakeholder feedback is required to produce effective deliverable results. Therefore, the team concludes that this approach will be the best fit in the case study analysis.

The UP model approach, shown in Figure 1, is an iterative process that consists of four major phases throughout the life cycle: inception, elaboration, construction, and transition. Through these main phases are activities or disciplines that represent an iterative workflow to follow: communication, planning, modeling, construction, deployment, software increment (Hunt 2014).

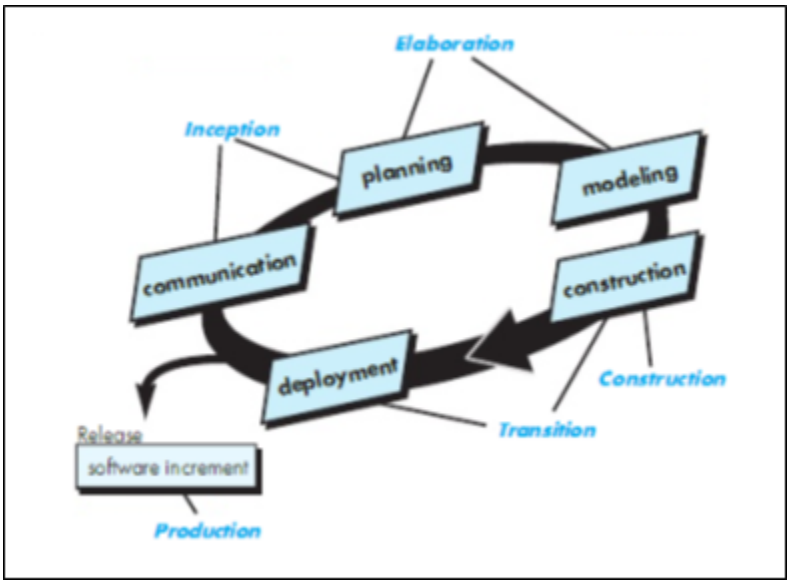


Figure 1. The Unified Process. Source: Pressman and Maxim (2019, 32).

The four phases capture the project in its entirety. The inception phase defines the scope of the project and the feasibility of the proposed solution. In the elaboration phase, functional and non-functional requirements of the system are defined, as well as establishing the creation of the architecture to be used. Through the construction and transition phases, a product is delivered and moved into an operational environment. These phases are considered the major milestones of the project's life cycle, and minor milestones are the activities that are completed within.

The activities of the process are linked together through the four major phases. While these activities are completed step by step, they are iterative and incremental, meaning that some activities may be working in parallel to other activities. They are performed through each phase to bring emphasis on the structure of the system and how the system should perform. The communication and planning activities produce the requirements definition of the stakeholders which are applied to the system architecture. This is followed by modeling in which creation of an initial design is established. Through the construction and deployment activities, a working product is produced and implemented into an operational environment for validation of requirements.

In order to meet the case study objective, the capstone team implements an Agile development methodology to the UP model. In particular, the team utilizes sprints, which are iterative developments, to deliver the project deliverables. As shown in Figure 2, there are three sprints to accomplish the case study. Each sprint results in a working digital architectural model of the UUV system and an audio and visual tutorial of the architectural development. The first sprint will address the Problem Domain. The second sprint will address the Solution Domain. If time permits, the third sprint will address the Implementation Domain.

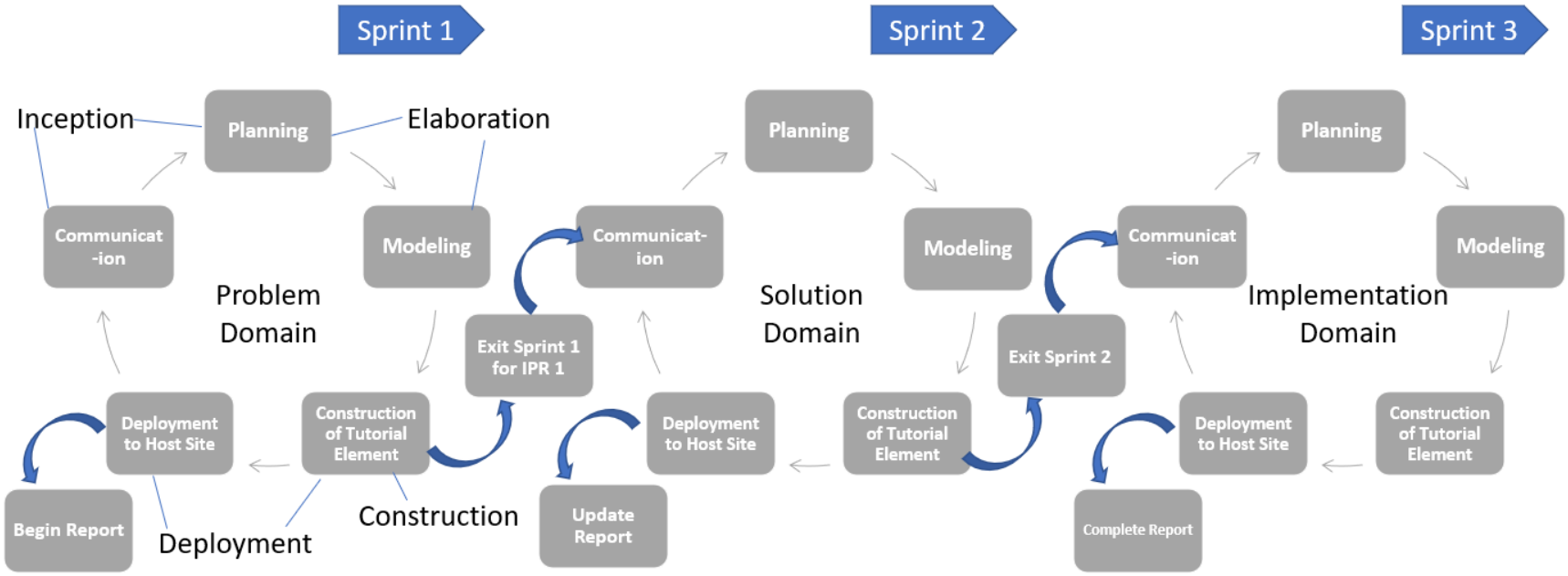


Figure 2. Team Icarus Modified UP

Throughout the three iterations, the MagicGrid methodology is employed to develop the problem and the solution domains in Cameo Systems Modeler, shown in Figure 3. The block activities are divided into the three sprints which overlay the activities of the modified UP model. In-process reviews (IPRs) mark the completion of the iterations and are presented to all relevant stakeholders for feedback and direction. Refinement of stakeholder requirements and traceability are constantly incorporated into the design for further completion.

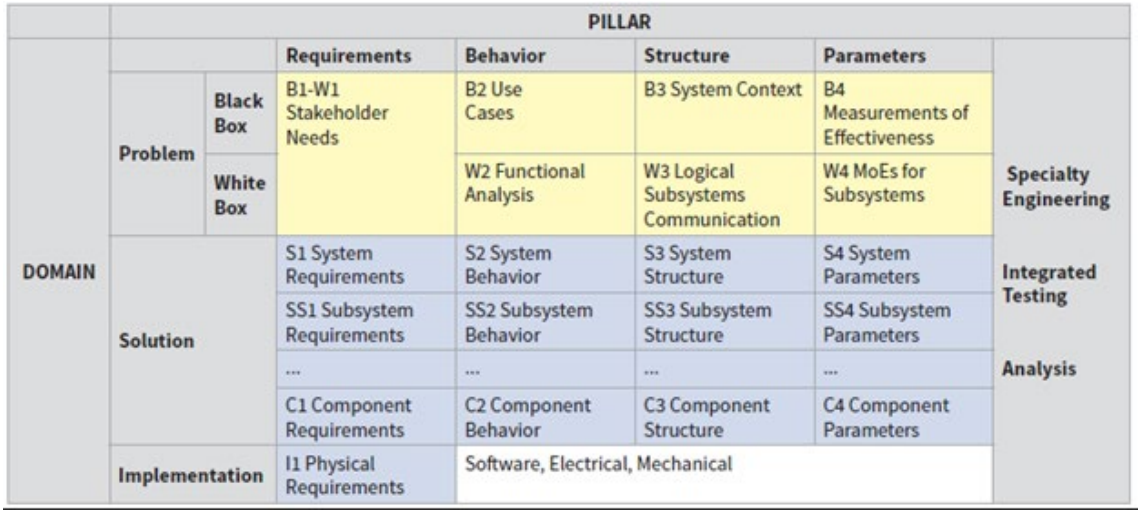


Figure 3. MagicGrid Architectural Development Process. Source: Aleksandraviciene and Morkevicius (2018, 11).

G. REPORT STRUCTURE

Chapter II first examines the current training programs are available at NSWC PHD. Once these are enumerated, the gaps in the programs are identified. Next, proposals are made for how this project will fill those gaps by using the processes in Chapter II. Chapter III discusses the fictional UUV and its concept of operations (CONOPS), use cases, and measures of effectiveness (MOE). Chapter IV details the architectural development of the project and how it will be implemented at NSWC PHD. Chapter V details the tutorial creation process. Chapter VI details how the gaps in previous training programs were filled and how this benefits the stakeholders and NSWC PHD.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The Navy offers a wide range of avenues for self-improvement and growth. The various means of which additional knowledge can be obtained include utilizing the online course catalogs from Navy e-Learning, The Defense Acquisition University (DAU), The Total Workforce Management Services (TWMS), and The Joint Knowledge Online JKO websites. Other training initiatives are offered online or in-person by commands when the training is relevant and beneficial. However, none of these training initiatives provide a comprehensive picture and application of MBSE and DE.

A. CURRENT TRAINING INITIATIVES

In the realm of online courses, examples of current, beneficial training courses include DAU's CLE 084 Models, Simulations, and Digital. This continuous learning module provides high-level information from a systems engineering perspective. It lays out how models, simulations and digital engineering can collectively benefit an entire system throughout its life cycle.

Another example of training offered via the online course catalog from the Navy e-Learning website to register for an in-person classroom course titled NAVWAR-MBSE-C-ILT-100 which is a model-based systems engineering for practitioner's level 1 cohort program. This program is designed for students who have previously completed the program and are looking to get reacquainted in the modeling environment. Some of the various methods used to teach students include weekly video tutorials, online discussions, laboratory assignments, and address topics like Object Orientated Systems Engineering Method OOSEM and SysML.

While these courses might provide opportunities for development in model-based systems engineering and digital engineering, the focus of this project is to first analyze what is provided at NSWC PHD in the format of both in-person and online courses.

PHD offers additional courses that are not actively listed on the online websites. Typically, the command training personnel will reach out to various detachments around the engineering community and offer an opportunity to attend these courses. For this

particular fiscal year, the course most closely related to the MBSE training criteria of this project is an SysML course hosted by Georgia Tech. This course is aimed at the Navy's In-Service Engineering Agent ISEA to support the various combat systems deployed onboard surface platforms—core PHD In-Service Engineer capability for integration, modernization, test and evaluation, and integrated product support throughout the system life cycle.

B. DIGITAL ENGINEERING – BASIC SKILLS NEEDED

The MagicGrid Book of Knowledge (BoK) is a manual that demonstrates a particular framework which is one of many ways to apply MBSE principles (Aleksandraviciene and Morkevicius 2018). This approach supplements existing methods that are assumed to be understood prior to beginning the tutorial that is demonstrated in the BoK. The MagicGrid BoK and the tutorials created for the purpose of this project are intended to provide more in-depth examples of how to apply MBSE concepts, appropriately construct diagrams, and how to link each diagram together. Both the BoK and the tutorials created are targeted for individuals who have a basic understanding of MBSE concepts, either at the beginning or intermediate levels. A beginning level knowledge of these concepts is defined as no basic understanding of these concepts and how to apply them. An intermediate level of knowledge of these concepts indicates that the individual(s) have spent time already learning about system architectures and diagrams and would have a general understanding of how and when to apply them.

C. GAP ANALYSIS BETWEEN CURRENT TRAINING AND DIGITAL ENGINEERING BASIC SKILLS

Although there are many courses available online, there currently exists a training gap specific to the MBSE world, as well as training gaps in introductions to common MBSE tools like Innoslate, Cameo, Magic Draw and Core. The courses mentioned in Section A provide a general overview of MBSE and modeling as well as the benefits of digital engineering. Although users may gain high-level knowledge of MBSE and DE in these courses, the users need both a comprehensive example and tutorials on how make these architectures and link them with the analysis software, which are not provided by

these courses. This project will help bring an individual with beginner or intermediate MBSE and DE knowledge and skills up to a more proficient and knowledgeable systems engineer for their project.

This project intends to fill the gap of in-depth tutorials and applications of MBSE architecture methods that currently exist and are common amongst the existing trainings listed in the previous section. This project will provide an in-depth demonstration of concepts that are taught in other MBSE architecture courses and the method described in the MagicGrid Book of Knowledge. This project is not intended to replace any existing training and should only be used as supplemental training for other courses that lay the foundation for the principles and processes applied.

D. CHAPTER SUMMARY

During the research of what current training courses exist that are offered through PHD, Navy E-Learning, DAU, JKO, and at other commands, only three courses were found with goals that resembled what the purpose of this project is. That being said, it was determined that none of these listed courses covered the intent of this project. It has therefore been determined that this project's goal of acting as supplementary training materials for other existing MBSE courses, to include the ones listed above, will be beneficial to current and future students.

THIS PAGE INTENTIONALLY LEFT BLANK

III. UUV NEEDS ANALYSIS AND SCENARIOS

In conjunction with the BoK framework, a realistic example is needed to help demonstrate the applications of MBSE and DE principles. The example chosen was an unmanned underwater vehicle (UUV). The U.S. Navy is investing large amounts of resources into the development of UUVs as they can put users farther out of harm's way. While the UUV example provided does not currently exist, it has been inspired by commercially available UUVs and different sensor payloads that can be loaded onto them. The following section will detail the mission sets of both the UUV and the theoretical UUV developed to accomplish these missions.

A. EFFECTIVE NEED

While underway, especially in a combat environment, it is often necessary for the captain of a Navy surface ship to inspect their ship for any damage. In a non-combat environment, this inspection could be made to determine the presence and/or severity of damage or equipment casualty. Normally, the process of examining the ship's hull, keel, rudders, screws, etc., would be conducted by Navy divers when the ship is in port. This process this more than just getting divers in their gear and letting them go over the side—complex tag-out safety procedures are conducted to ensure the safety of the divers near moving parts and intake valves. Safety divers must also be on standby to assist in case of emergencies. In addition, the ship remains stationary while in port.

The above scenario of a thorough hull inspection by divers simply is not feasible when underway or in a combat environment. In the past, the captain of the ship would need to make the best determination of the state of the ship given the status of certain systems and subsystems of the ship and hope for the best. The use of a UUV would provide a low-cost, low-risk method of providing a detailed hull inspection.

B. CAPABILITY GAP

In this scenario, the theoretical UUV is connected to the topside control unit via a tether which provides communications to and from the UUV. Due to the presence of a

tether, it would not be recommended to use this UUV to enter a flooded area with large amounts of debris. This is due to a high possibility of the UUV tether becoming tangled in the debris and therefore making recovery of the UUV difficult or impossible. Although commercial off-the-shelf wireless UUVs have been developed, their range is limited depending on conditions and line-of-sight. These wireless UUVs communicate via optical sensor. An additional means of navigation in this would be autonomous flight. This would require a very high degree of software development and integration. This is beyond the scope of this case study.

To ensure a realistic and complete analysis of an UUV that could be developed for use by the U.S. Navy, a thorough description of the UUV, its concept of operations (CONOPS), operational architecture, measures of effectiveness, and mission set descriptions will be provided. Details such as the software development of the UUV system are beyond the scope of this study and intentionally omitted.

C. OPERATING ENVIRONMENT

In most cases, the UUV can be set up for operations directly from the deck of the ship that the UUV will be scanning. The ship will be in open ocean. In cases where the current is high, it would be best if both the ship and the UUV drift with the current. Otherwise, the ship can be anchored, so it does drift while under inspection. The UUV will only operate down to the depth necessary to provide a complete scan of the ship. This depth is no more than 15 meters.

D. UUV MISSION SETS

Mission sets would include hull searches and maintenance or battle damage assessments (BDA). An example of a hull search mission would be strafing the UUV perpendicular to the ship hull with the forward-facing camera and sonar pointed at the hull. Another mission could be a spot inspection of damage, either from regular operations or from combat. The next few paragraphs will describe typical UUV deployment, mission, and recovery.

First, the operator will remove the UUV from storage, connect the topside control console, fiber optic tether, and UUV. Next, they will conduct a deck check to ensure functionality of all major system components, such as the thrusters, sonar, forward and rearward looking cameras, and navigation system. Once the UUV's operational readiness is confirmed, the operator can deploy the UUV. This is done by lowering the UUV by the tether, hand-over-hand, into the water from the side of the deck. Once the UUV is in the water, the operator can maneuver the UUV so that it is a safe distance from the ship. Next, the operator will ensure the UUV has a good navigation solution by using GPS or a Defense Advanced GPS Receiver (DAGR). Once these steps are complete, the operator can commence the mission—whether it be a hull search, a BDA, etc. A similar case would be the same steps as above, but the UUV will be deployed from a Rubber Hull Inflatable Boat (RHIB) or similar vessel deployed from the ship being scanned.

The UUV will scan the outside of ship for any potential damage. From this outside assessment, data will be recorded and reported to the chain of command. It is recommended that the UUV remains outside of the ship unless absolutely necessary to evaluate critical ship systems. This is to prevent the tether or the UUV from becoming tangled with debris when in a flooded space, (e.g., a flooded compartment compromised by battle damage). The UUV will be lowered into and recovered from the water via the tether.

At the conclusion of the mission, data can be downloaded from the topside control unit for post-mission analysis by the operator or by anyone in the chain-of-command to determine the best actions in terms of the operational status of the ship and its subsystems.

E. UUV PERFORMANCE REQUIREMENTS

To support the mission sets and environments previously identified in this section, the UUV will meet the following operational and performance requirements:

1. The UUV will operate in a saltwater undersea environment

(i) Shall operate at depths up to 15 meters.

(ii) Shall maintain position in the x-y-z plane within ¼ meter in Sea State 3.

(iii) Shall maintain position in the x-y plane within ¼ meter while being subjected to a 2-knot current in any direction

(iv) Shall operate in water temperatures from -2 to 40 degrees Celsius.

2. The UUV shall be capable of carrying batteries on board to power itself.

i) The onboard vehicle batteries shall provide at least 3 hours of in-water operational time in a current up to 2 knots.

ii) The batteries shall be removable and rechargeable.

iii) The batteries shall have a total power rating of at least 800-Watt hours.

3. The UUV should be able to allow for partially autonomous flight to assist the user

i) The UUV shall maintain pitch and roll hold inputs within 2 degrees in Sea State 3.

ii) The UUV shall maintain, within 0.2 meters, depth or altitude.

iii) The UUV shall maintain heading to within +/- 3 degrees.

iv) The UUV shall be able to autonomously travel to and from pre-determined waypoints within 200 meters of the operator.

4. The UUV shall carry and be able to export the data of three payloads in .mp4 format for post-mission analysis.

i) The UUV shall record and export HD video on both a forward and rear-facing camera.

ii) The UUV shall record and export sonar data with a resolution of no less than 1 mm at 1 meter.

5. The UUV shall have a topside control unit that displays real-time HD video and sonar data, as well as positioning, guidance, and telemetry data to the operator.

6. The UUV shall be connected to the topside controller via a rugged fiber-optic tether

i) The tether shall have a break strength of at least 200 lbs.

ii) The tether shall have Kevlar coating to resist being severed

iii) The tether shall be at least 300 meters in length.

7. The UUV shall not weigh more than 30 lbs.

8. The UUV's operational footprint shall fit on the deck of a six-meter RHIB.

9. The UUV shall require no more than one multi-tool for user maintenance.

F. CHAPTER SUMMARY

The purpose of this chapter was to provide a thorough explanation of the theoretical UUV for this project. A UUV was chosen due to the relevance of UUVs in the U.S. Navy's mission both in the aforementioned scenarios and in other mission sets. This theoretical UUV concept was inspired by different commercially available UUVs that might be purchased by the U.S. Navy for these mission sets. Although some creative liberties were taken with a few of the design parameters as to not directly copy a single UUV system, these are still well within possible design parameters— and a future UUV capability.

The design parameters describe a realistic UUV but do not go into any overly complex technologies, such as autonomous flight, besides simple station-keeping, heading hold, and altitude hold. It is assumed that the level of autonomy required beyond this is far too complex for an MBSE analysis and would provide no added benefit to this study.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ARCHITECTURAL DEVELOPMENT

Now that the gaps in previous and current training programs have been identified in Chapter III, we discuss the MagicGrid process and how it is being modified to meet the needs of this case study. This section will explain the MagicGrid process and its decomposition into three domains: the problem, solution, and implementation domains. Each domain is further broken down and gives finer details on the architecture of a system such as a UUV.

A. MODIFIED MAGICGRID PROCESS EXPLANATION

For this project, the MagicGrid BoK is used as a building block for the modeling of the UUV architecture. The MagicGrid approach is based on a "Zachman style matrix and is designed to guide the engineers through the modeling process and answer their questions" (Aleksandraviciene and Morkevicius 2018, 10). The BoK uses a tutorial of a Vehicle Climate Control System (VCCS) to demonstrate the various concepts of the MagicGrid Architectural Development Process. While the specific system is obviously different than a UUV, the concepts and processes discussed are applied to the UUV architecture. The only difference between what the BoK demonstrates and what is completed for this project is that the level of detail modeled for the UUV goes deeper than that of the VCCS. All modeling concepts remain the same.

Please note that a considerable portion of time will be used referencing the MagicGrid phases, pillars, domains, and overall structure outlined per section under review. Please note the red outlined boxes over each figure as these will aide in the understanding and tracking of the developmental process.

B. PROBLEM DOMAIN

The problem domain is utilized to capture the stakeholders needs and refine them with SysML models. These models will need to be clear and descriptive of the problems that the System of Interest (SoI) will need to address. Prior to decomposing the problem domain into the two main phases, the first cell of the B1-W1 Stakeholder Needs will be

addressed. The process of initially gathering stakeholders' needs can be conducted in various ways to include, but not limited to, polling information via focus groups, conducting interviews, and questionnaires. The information gathered in this step will be used to develop the behavior, structure, and parameters for the Black and White Boxes. This information will later become the basis for the system requirements (S1) in the follow-on steps.

The first phase, also called the "Black Box," focuses mainly on the interactions with the environment while ignoring any internal structure and behaviors. The second phase, or "White Box," will view the "Black Box" phase within the problem domain to refine what the SoI will address. This will bring up details on how the system shall operate: "In other words, the functional analysis of the SoI is performed. It is also important to note that the problem definition doesn't cover the physical architecture of the SoI" (Aleksandraviciene and Morkevicius 2018, 12).

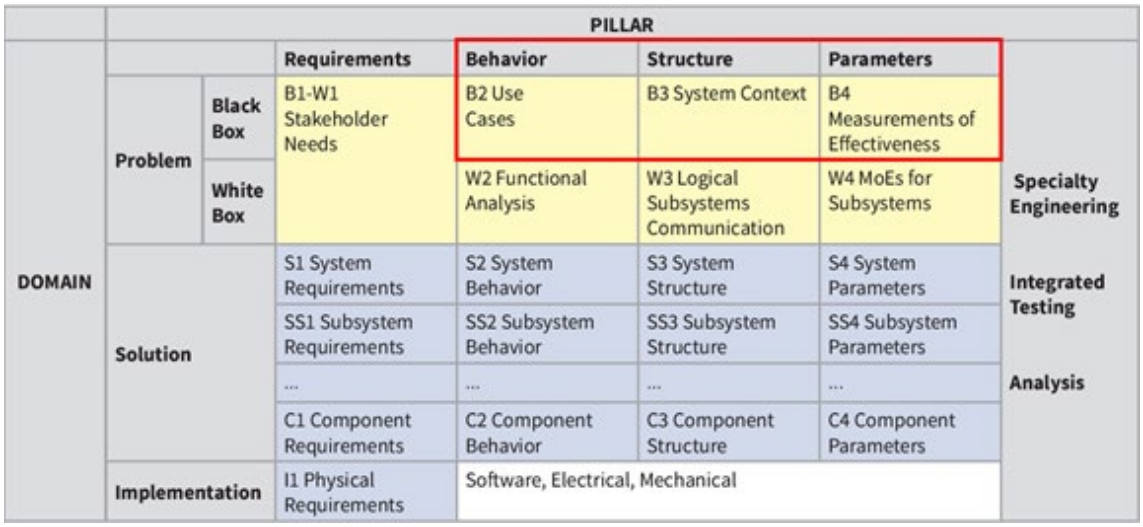
Figure 4 demonstrates from top left to right the problem domain for both the White and Black Box breakdown to describe the pillars requirements, behavior, structure, and parameters, respectively.

			PILLAR				
DOMAIN			**Requirements**	**Behavior**	**Structure**	**Parameters**	
	Problem	**Black Box**	B1-W1 Stakeholder Needs	B2 Use Cases	B3 System Context	B4 Measurements of Effectiveness	**Specialty Engineering**
		White Box		W2 Functional Analysis	W3 Logical Subsystems Communication	W4 MoEs for Subsystems	
	Solution		S1 System Requirements	S2 System Behavior	S3 System Structure	S4 System Parameters	**Integrated Testing**
			SS1 Subsystem Requirements	SS2 Subsystem Behavior	SS3 Subsystem Structure	SS4 Subsystem Parameters	
			...	...	...	...	**Analysis**
			C1 Component Requirements	C2 Component Behavior	C3 Component Structure	C4 Component Parameters	
	Implementation		I1 Physical Requirements	Software, Electrical, Mechanical			

Figure 4. MagicGrid Problem Domain. Adapted from Aleksandraviciene and Morkevicius (2018, 11).

C. BLACK BOX

As previously discussed, the problem domain will begin with gathering the stakeholders needs for both the Black and White Box development. Once this information is obtained the black box behavior (B2), structure (B3), and parameters (B4) can be developed as shown in Figure 5.



			PILLAR				
DOMAIN			Requirements	Behavior	Structure	Parameters	
	Problem	Black Box	B1-W1 Stakeholder Needs	B2 Use Cases	B3 System Context	B4 Measurements of Effectiveness	Specialty Engineering
		White Box		W2 Functional Analysis	W3 Logical Subsystems Communication	W4 MoEs for Subsystems	
	Solution		S1 System Requirements	S2 System Behavior	S3 System Structure	S4 System Parameters	Integrated Testing
			SS1 Subsystem Requirements	SS2 Subsystem Behavior	SS3 Subsystem Structure	SS4 Subsystem Parameters	
			...	...	...	...	Analysis
			C1 Component Requirements	C2 Component Behavior	C3 Component Structure	C4 Component Parameters	
	Implementation		I1 Physical Requirements	Software, Electrical, Mechanical			

Figure 5. MagicGrid Black Box. Adapted from Aleksandraviciene and Morkevicius (2018, 11).

The Black Box was initially constructed from only the inputs and outputs of the system to be analyzed. The level of detail related to the inputs and outputs will be without gathering any data on the internal structure or behavior (Aleksandraviciene and Morkevicius 2018, 12).

These data gathered from the stakeholders will either detail functional or non-functional needs and will drive the functional needs to the B2 Use Cases for further refinement. Use cases are modeling tools that are applied in various scenarios used to model requirements for a SoI. If the needs were identified as non-functional, then the stakeholders will need to determine non-functional characteristics that the SoI will have. These non-functional characteristics will later be used for the measures of effectiveness (MoE) for the parameters of B4. B3 System Context will determine entities external to the

system that do not belong but rather interact with the system. Examples of external entities to the system include humans, weather environments, and organizations.

D. WHITE BOX

After constructing the outline of the Black Box or completion of the operational analysis of the SoI, the White Box phase can be constructed with respect to the SoI. This will provide the detail within the system and how it shall operate, yielding a refined analysis of the system functions used to identify the functional blocks. The functional blocks will be used as the initial steps for the solution architecture of the system. These blocks are show in Figure 6.

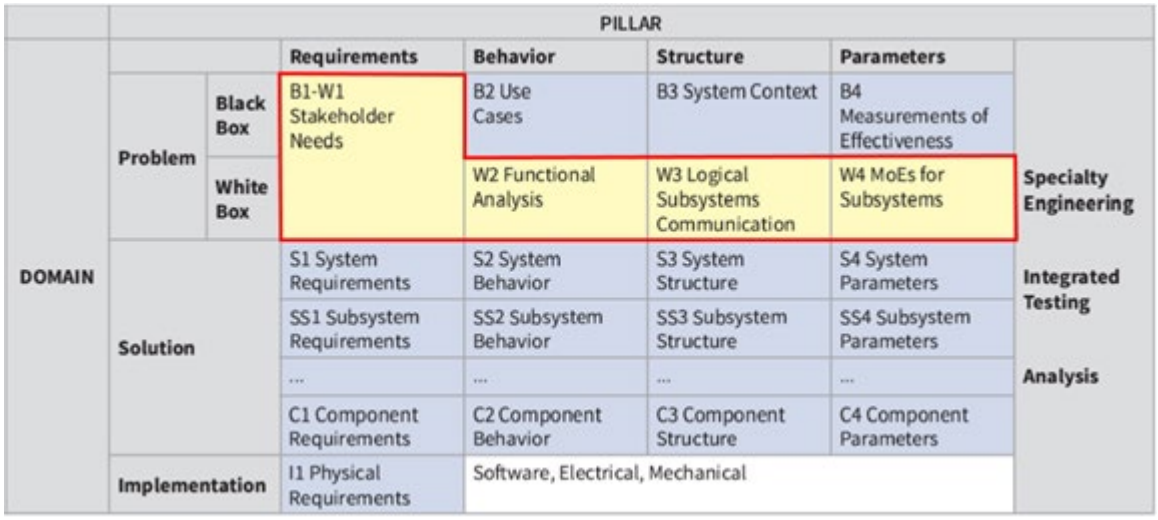


Figure 6. MagicGrid White Box. Adapted from Aleksandraviciene and Morkevicius (2018, 11).

The W2 Functional Analysis is where the data from the B2 Use Case will be further decomposed to capture every function the SoI will perform. The level of refinement and detail captured will be used to further decompose the SoI in the W3 Logical Subsystem Communication step. The decomposing of the functions will help identify functional blocks, also known as the logical subsystem, that directly perform the functions. In essence, using the W3 Subsystem Communication process to back feed additional W2 functions

and associate the subsystem to their respected system will need to be captured and assigned in the W2 Functional Analysis process.

Functional behaviors will be divided into a functional decomposition state. This is reflected in Figure 7, in the third layer structure from subsystem B into layer 4 where subsystem B is decomposed into subsystem B1, B2, and B3. The level of decomposed functions can be as many decomposed functions as necessary to provide a coherent description of the system.

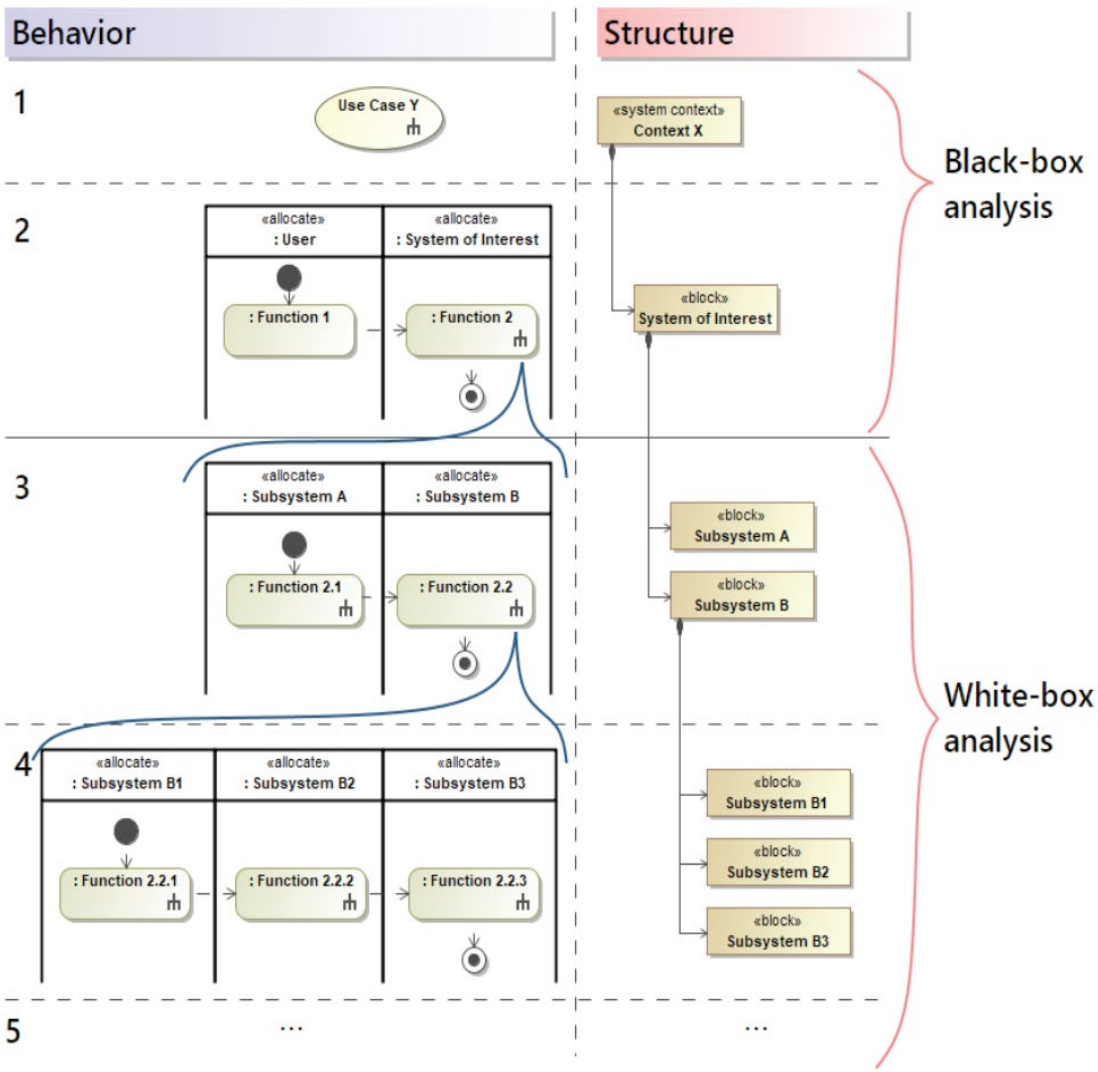


Figure 7. MagicGrid Black and White Box Analysis. Source: Aleksandraviciene and Morkevicius (2018, 50).

The W4 MoEs for Subsystems block is where the specific MoEs per subsystem is captured for the SoI. The MoE will be further refined in this block. Please note that this block may be optional if there are no non-functional stakeholders needs.

E. SOLUTION DOMAIN

After completing the problem domain, the focus will shift into the solution domain. This domain is shown in Figure 8. Being solution-minded will prove key when defining the details in the model. The details gathered will be used to construct the solution architecture resulting in a well-defined solution design of the system or possible various designs. This variability in solutions is represented in Figure 9 and Figure 10, where multiple solutions will need to be compared and a trade-off analysis conducted to optimize the selection in the solution domain. An important takeaway from the problem and solution domain is that they can be developed independently from two different organizations. This can become beneficial when a separate organization requires a diverse solution.

			PILLAR				
DOMAIN			Requirements	Behavior	Structure	Parameters	
	Problem	Black Box	B1-W1 Stakeholder Needs	B2 Use Cases	B3 System Context	B4 Measurements of Effectiveness	Specialty Engineering
		White Box		W2 Functional Analysis	W3 Logical Subsystems Communication	W4 MoEs for Subsystems	
	Solution		S1 System Requirements	S2 System Behavior	S3 System Structure	S4 System Parameters	Integrated Testing
			SS1 Subsystem Requirements	SS2 Subsystem Behavior	SS3 Subsystem Structure	SS4 Subsystem Parameters	
			...	...	...	...	Analysis
			C1 Component Requirements	C2 Component Behavior	C3 Component Structure	C4 Component Parameters	
	Implementation		I1 Physical Requirements	Software, Electrical, Mechanical			

Figure 8. MagicGrid Solution Domain. Adapted from Aleksandraviciene and Morkevicius (2018, 11).

Constructing the solution architecture is done utilizing the following pillar areas, specifying system requirements, behavior, structure, and parameters of the system under design. This is no longer a generic SoI but a specific solution with specific items identified

to perform the various system functions. For example, a generic SoI may contain a power system whereas a solution architecture would contain a specific subsystem/component such as a 4000 mAh battery pack. The construction of the solution architecture will most likely require going into each pillar and decomposing each to deeper level, all the way into the component level or further if the need is identified. Depending upon the number and level of decomposed pillars, this will increase the overall precision of the system description. The construction of the solution architecture can be done in a single model to include the problem domain definition. Most often, it would be easier to produce separate models to better manage the complexity. The utilization of separate models will allow for a greater level of decomposition of the model which in turn will provide better model ownership, change analysis and control, concurrent modifications, and model re-use. As previously mentioned, more than one solution can be used within the solution architecture as is seen below. The High-Level Solution Architecture (HLSA) is the core of the model breakdown; it is from here we are able to decompose further into the individual system and subsystems. Viewing the model from this perspective gives an opportunity for the system architect to see the big picture of the system being developed. The system architect can assign work packages to separate engineering personnel based on top level view. Once all teams have completed their portion of the solution, the system architect will integrate them together into a single model as seen in Figure 10. At this point, the preferred solution from each subsystem will be used to construct one model. With the full model of the system under design built the system engineer can perform a trade-off analysis to verify optimal system configuration.

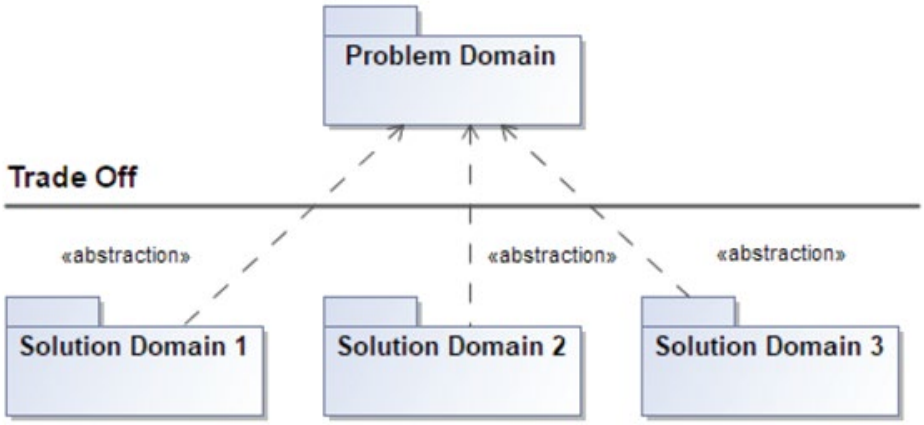


Figure 9. MagicGrid Problem Domain Decomposition. Source: Aleksandraviciene and Morkevicius (2018, 79).

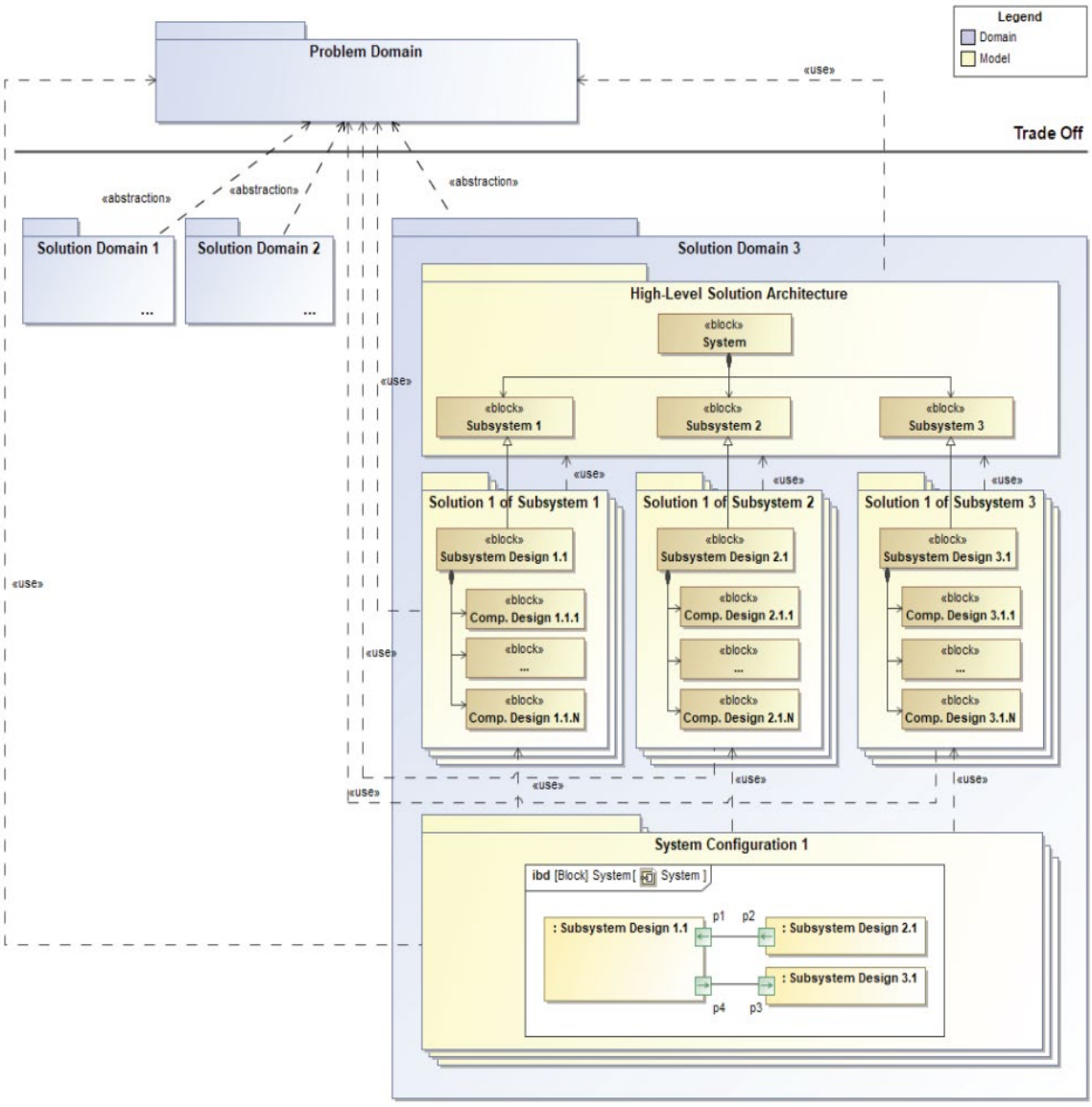


Figure 10. MagicGrid Solution Domain Decomposition. Source: Aleksandraviciene and Morkevicius (2018, 81).

Incorporating the solution architecture into the model will become more complex than the initial problem domain when implementing into the model. “In most cases, the scope and complexity of the solution architecture exceeds the scope and complexity of the problem domain definition up to several times, especially when it includes more than one solution” (Aleksandraviciene and Morkevicius 2018, 80).

F. IMPLEMENTATION DOMAIN

With the solution architecture fully developed the implementation domain follows. Using the optimal system configuration paired with the mindset that the SoI is no longer abstract but a real system. The detailed model-based design (MBD) can be derived using tools similar, but not limited to, electrical and mechanical CAD software. To implement the system, the solution domain gives the detailed physical requirements. These will include all subsystems. As stated in the MagicGrid Book of Knowledge, "The physical requirements specification is given to the engineers of various disciplines, who are responsible for designing different aspects of the system under implementation." (Aleksandraviciene and Morkevicius 2018, 157). Overseeing the physical requirements is remarkably like the managing of the stakeholders needs and system requirements. Similarly, these requirements can be grouped decomposed into subsystems as needed. Each physical requirement should align with at least one or more elements of the solution domain. If this not the case, the solution domain should be updated, or the physical requirements should be altered to conform to the model. Since the implementation domain is largely requirements management developed with domain engineers, and is very solution dependent, the authors consider it out of scope and did not include it in the tutorial material. As seen in Figure 11, the position of the implementation domain is below both the problem and solution domains in MagicGrid.

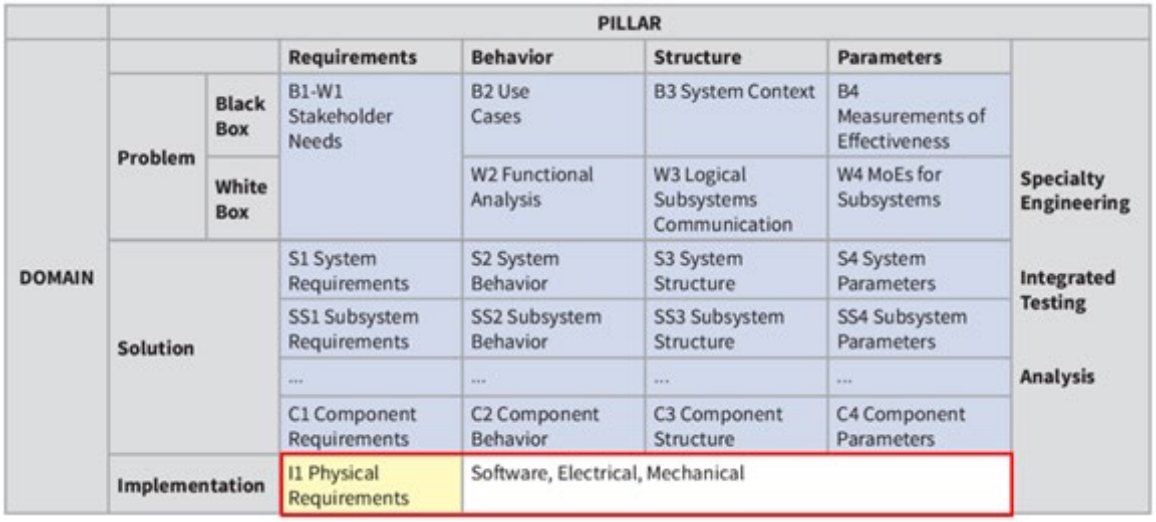


Figure 11. MagicGrid Implementation Domain. Adapted from Aleksandraviciene and Morkevicius (2018, 11).

G. INCORPORATING DE ELEMENTS TO THE ARCHITECTURE MODEL

The digitalization of engineering tasks is becoming ever more prevalent. Where completing prototype projects used to take multiple people performing multiple hours of work in creating hand drawings, cost estimations, material research, and more, DE has allowed industry to combine many of these tasks in a virtual environment. Digital engineering has allowed the combination of creating models and being able to test and configure to optimal design virtually instead of having to hand draw each iteration of changes. To successfully implement DE concepts, the program and architecture must be able to provide real-world data to assist the engineer with making decisions, especially decisions early in the product or systems life cycle. By pairing analysis tools with the systems architecture model, engineers can specify the system conditions and perform analyses that yield informative and reliable results. Integrating analysis tools with the architecture model also allows for multidisciplinary engineering principles be applied to a MBSE architecture, allowing for a more complete picture and representation of the system being designed.

The goal of this project is to incorporate the use of a middleware analysis software, ModelCenter Integration and ModelCenter MBSE by Phoenix Integration, to provide

engineering analysis calculations for the UUV architecture designed in Cameo. Input parameters will be created and uploaded into ModelCenter to calculate specific data. The data that is calculated within ModelCenter Integration is then linked to Cameo Enterprise Architecture using ModelCenter MBSE. This allows the Cameo UUV architecture to be able to receive determined values and implement them into their respective component locations. The resulting data with the MBSE software Cameo Enterprise Architecture.

ModelCenter provides many different methods to input variables for calculations. Some of these methods include Excel, SolidWorks, and MATLAB. For this project, Microsoft Excel is the primary focus of variable input. ModelCenter utilizes Microsoft Excel by allowing the user to import an excel file already created. The Excel file is used for input and output variables. These Excel files can also be linked to other Excel files to calculate desired outcomes. The desired outcomes for this project are buoyancy of the vehicle, drag force against the vehicle, and power required to operate the vehicle.

When creating the Excel files, two different Excel files are created per output calculation. These Excel files are specific to each desired output; one Excel file for buoyancy input and one for the buoyancy output. The two different Excel files are created to demonstrate ModelCenter's ability to link multiple files together, however, a single Excel file with the inputs and outputs linked together in one Excel file will also give a similar result. The two different Excel files can be seen in Figures 12 and 13, with Figure 13 showing an example of a range name.

Item	Value
Vehicle Length (Buoyancy)	650
Vehicle Height (Buoyancy)	260
Vehicle Height (Drag)	260
Vehicle Width (Buoyancy)	450
Vehicle Width (Drag)	450
Vehicle Density	30
Fluid Density (Buoyancy)	997
Fluid Density (Drag)	997
Gravity (Buoyancy)	9.8
Gravity (Drag)	9.8
Velocity	1.19
Drag Coefficient	0.15
Battery Life	890
Voltage	25
Run Time	3.591896
Vehicle_Volume	0.04394
Vehicle Mass	1.3182
Area_Facing_Fluid	0.117
Drag Force	12.389
Buoyancy	429.3202
Object Weight	12.91836
Power_Consumption	247.78

Figure 12. Buoyancy Input Parameters Excel File

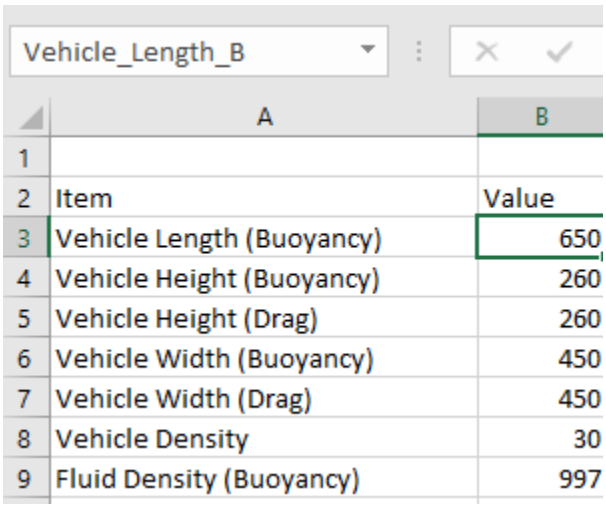


Figure 13. Buoyancy Calculation Excel File

The Excel file must be properly formatted to be read by ModelCenter. For ModelCenter to read the values in the cells containing the input or output values, these cells must have range names. Without these range names and values, ModelCenter will not be able to import the appropriate values. A more in-depth explanation of how to set up the Excel files is discussed in the tutorial created for this section (see Appendix). A critical aspect of making ModelCenter and Cameo work together is the analysis server. The analysis server allows the ModelCenter software to save the results from the calculations conducted after the model has been established. These results can then be brought into ModelCenter MBSE and connected to the Cameo value properties that the user creates. In order for the analysis server to operate correctly, the user needs to either create or select a folder where the analysis server will save the results. A detailed description of how to set up the save path is discussed in the tutorial (see Appendix). A standard window for the analysis server is shown in Figure 14.

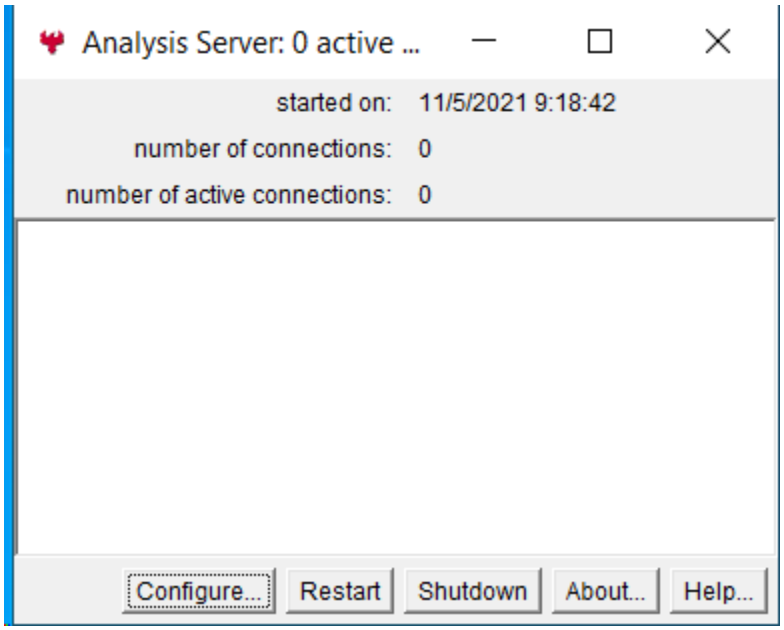


Figure 14. Image of Analysis Server Interface

Once the Excel file(s) are created and the analysis server is running correctly, the user can begin importing the Excel files into ModelCenter. For the model created in the

tutorial, three different analyses were created using the respective Excel file pairs, e.g., the buoyancy inputs file and the buoyancy calculation file. These file pairs were repeated with drag and power. A detailed walkthrough of how to import the Excel file pairs into ModelCenter is listed in the tutorial (see Appendix). The resulting model that was created in the tutorial is shown in Figure 15.

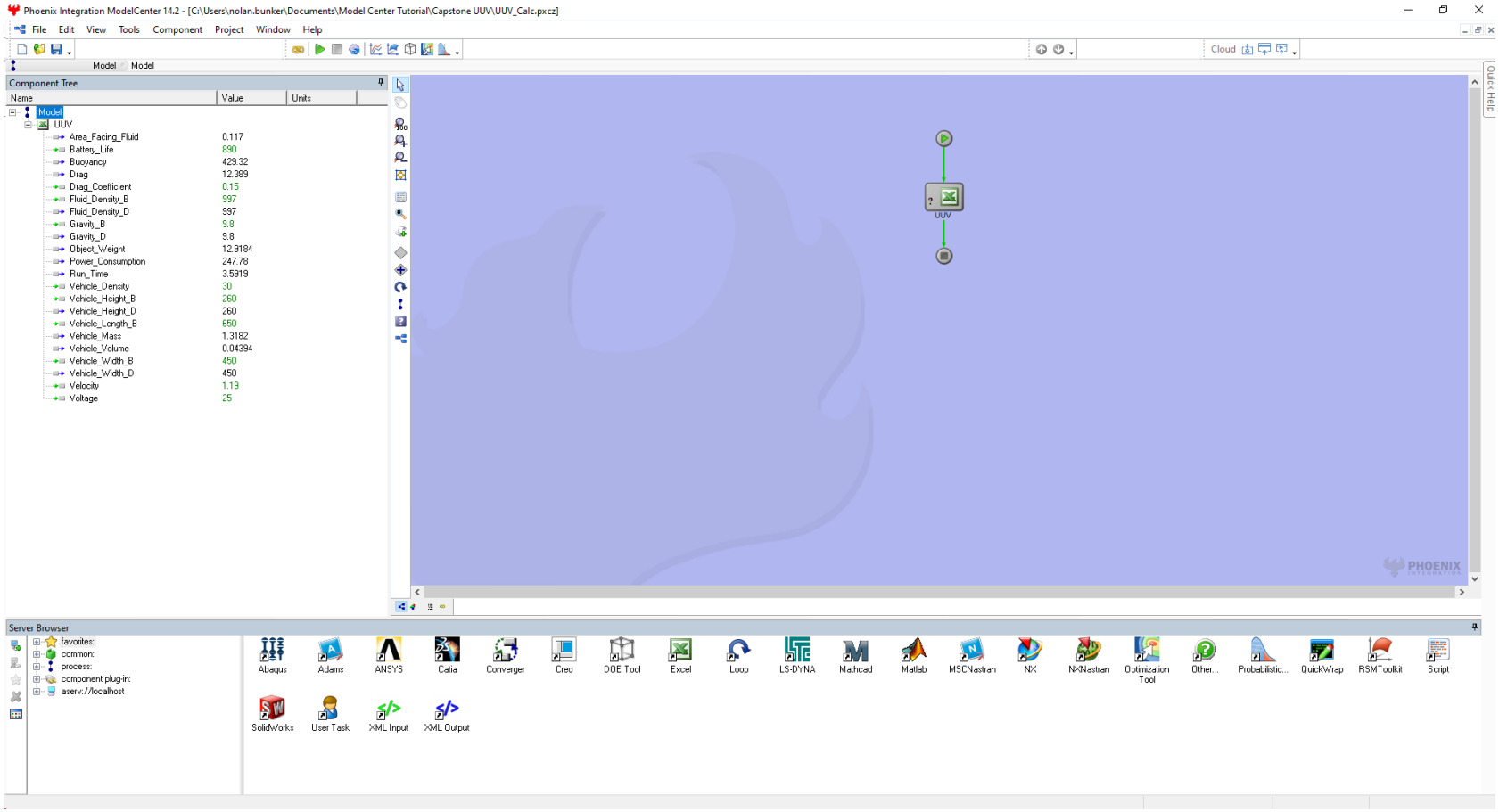


Figure 15. ModelCenter UUV Buoyancy Model

H. COMBINING MODELCENTER AND CAMEO

The integration of ModelCenter into Cameo is useful because allows for analysis tools to assist with design decisions. With the foundation of the analysis created in ModelCenter using Excel file pairs, it is time to incorporate that work into Cameo. This section will detail how a user can incorporate ModelCenter into Cameo and create a link between these two programs that utilizes their respective functionalities for design decisions.

1. Architecture Model and Excel Analysis Integration

The first step is to ensure that the ModelCenter analysis server is running. The analysis server is a critical component for the link between ModelCenter and Cameo. Once this has been verified, the user can open the Cameo model he or she has created. If the user has been following along with the tutorials presented from this project, a good example of a Cameo model will already have been created.

The next step is to create a new package in the architecture model to have a single location for all the data to be collected, as well as act as the primary location in which the link to ModelCenter. This allows for easier creation and tracking of the diagrams and elements utilized when using the ModelCenter MBSE plug-in found in the Cameo Tools menu.

Next, one must create a Block Definition Diagram (BDD). We created a BDD that contained all of the inputs and outputs for each variable being calculated for buoyancy, power, and drag, shown in Figure 16. This setup allows for easier linking of the analysis and the Cameo value properties. When creating the calculation and input blocks, ensure that these input blocks match the Excel file variables. This assists with ensuring that each value is accounted for appropriately. Ensure that a default value is entered in for each value property otherwise an error will occur during the ModelCenter MBSE linking process.

The next step is to create a Block Definition Diagram (BDD). A BDD that contains all of the inputs and outputs for each variable being calculated for buoyancy, power, and drag is shown in Figure 16. This setup allows for easier linking of the analysis and the

Cameo value properties. When creating the calculation and input blocks, ensure that these input blocks match the Excel file variables. This assists with ensuring that each value is accounted for appropriately. Ensure that a default value is entered in for each value property otherwise an error will occur during the ModelCenter MBSE linking process.

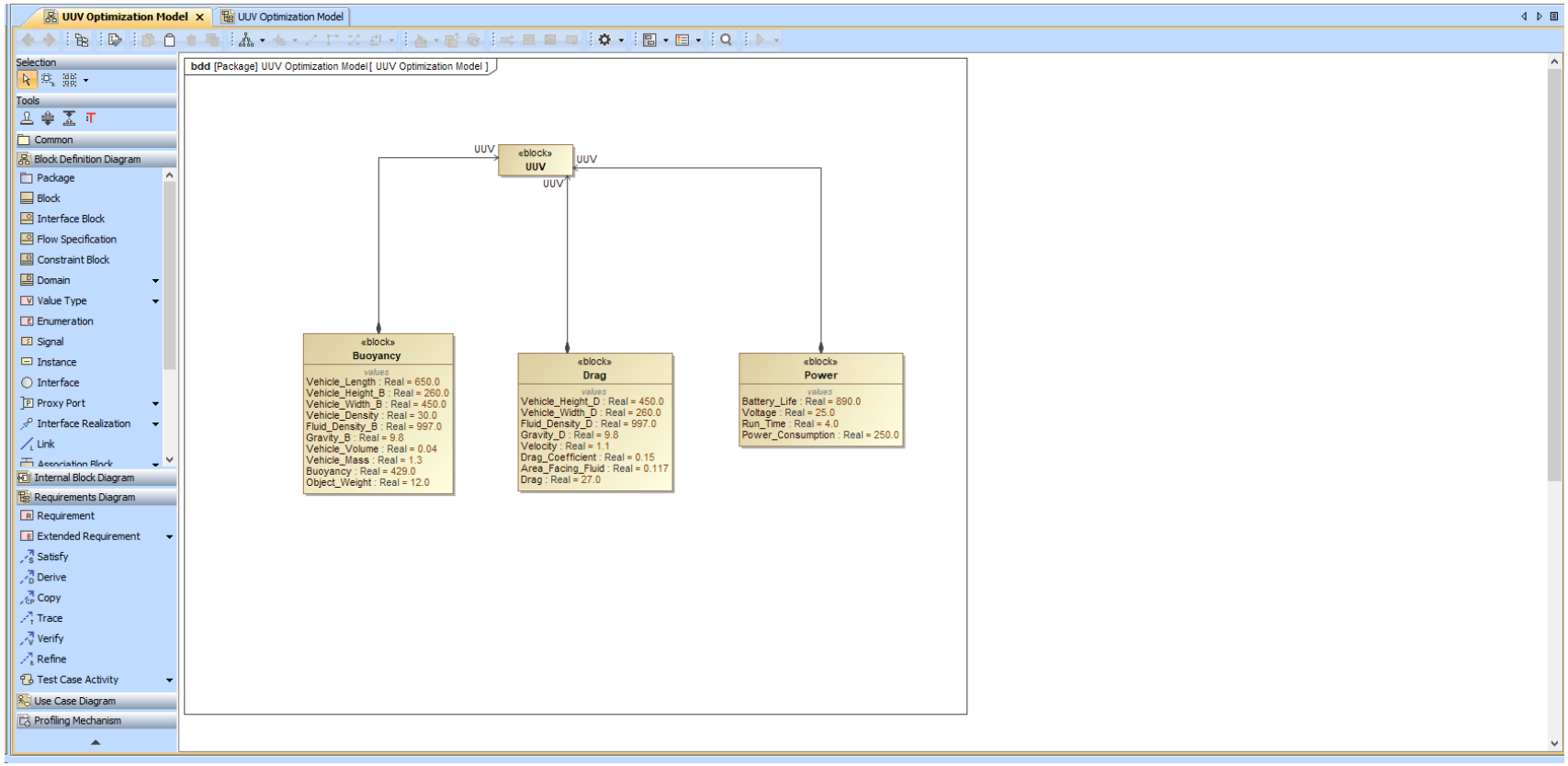


Figure 16. Cameo UUV Model

Once the block definition diagram is created, the user can link the ModelCenter analyses. This can be done by using Cameo's ModelCenter MBSE tool. The model is linked to each respective analysis in a similar manner as to how the Excel files are linked within ModelCenter; Figure 17 shows how the link between the model and the respective analyses can look.

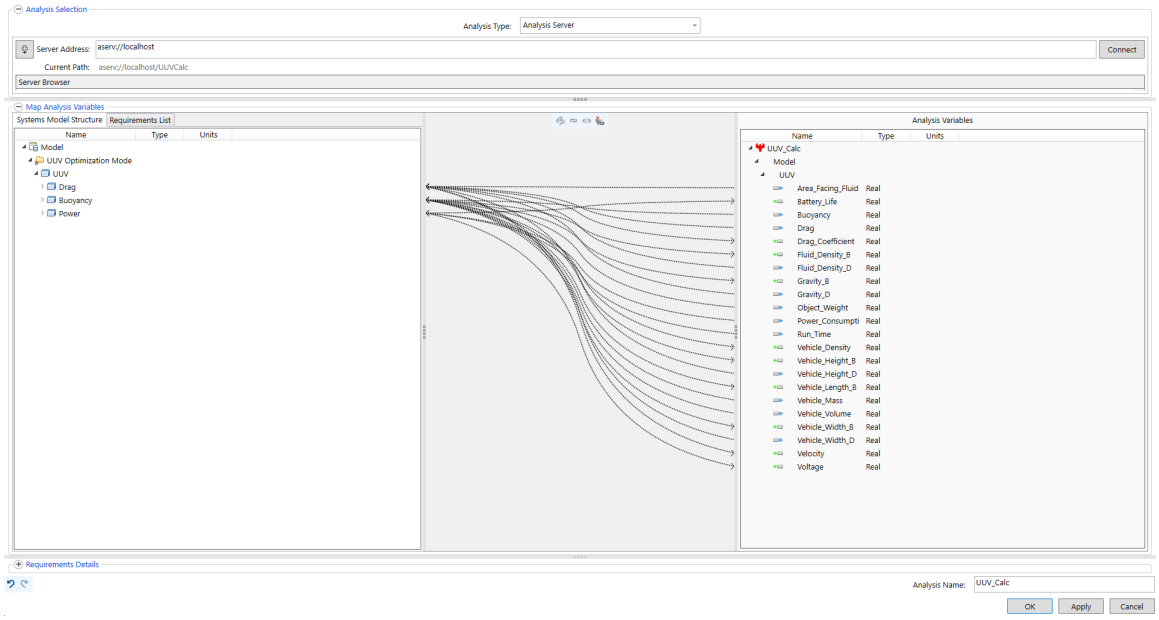


Figure 17. ModelCenter and Excel Files Link

At this point, the Cameo model can supply information to the external analysis tool, Excel, which then conducts the calculations needed and allows manual trade space exploration. Once this is completed, the user can now set up requirements and constraints that the model must meet.

2. Automating Requirement and Constraint Verification

Following the same process described for creating UUV model, the use can create requirements and constraints within the SysML model. The step-by-step process for how the requirements and constraints were created and linked within Cameo and ModelCenter MBSE is described in the Appendix under ModelCenter MBSE Tutorial Part 2. These constraints and requirements allow the user to see whether the requirements and constraints

are met during manual trade space exploration. They are integral to the optimization process that can also be completed within ModelCenter MBSE.

3. Optimizing the Design

The ability to use the optimization tool within ModelCenter and ModelCenter MBSE allows the user to optimize the design of the modeled system. For the purpose of this project, physical characteristics are used with physical characteristic constraints that needed to be met. A detailed walkthrough on how to create a design optimization is shown in the ModelCenter MBSE Tutorial Part 3 (see Appendix). ModelCenter has a host of optimization methods, but due to time constraints, only one optimization method is demonstrated in the tutorial. The combination of ModelCenter, ModelCenter MBSE, and Cameo creates a powerful tool combination for digital engineering. It allows subject matter experts to make design decisions without having to build the physical model and can adjust for changing requirements with little disruption or inconvenience.

I. SCOPE OF INTEGRATION EFFORTS

For this project, only Microsoft Excel was used to conduct these analyses and provide design decisions. In addition to Excel, integration of the CAD software SolidWorks into the analyses process was attempted. The goal was to design a representative 3D model with the same input parameters in SolidWorks as in Excel before incorporating the software into the ModelCenter analyses. By adding a SolidWorks 3D model into ModelCenter, the values that from the results of the optimization study would also be added into the 3D model. This would be an extremely useful tool for designers to easily allow for design changes within their models. Unfortunately, the integration for SolidWorks and ModelCenter could not be accomplished during the allotted time for this project. In an attempt to correct for the issues that were being experienced, the team contacted Phoenix Integration for any potential solutions. Based on the feedback received, the theory is that there were two potential reasons for the errors. The first reason was due to the type of license being used. Since the license allotted to NPS is a student license, there may be something within the license agreement that prevents outside collaboration with other programs. The second potential reason for the errors is the way that SolidWorks was

installed on the various computers. Due to the time limitations of the project, the team could not further investigate the source of these errors. It is recommended that future work be placed into discovering how to fix the errors that were experienced when attempting to incorporate SolidWorks into ModelCenter.

J. CHAPTER SUMMARY

The purpose of the material presented in Chapter IV was to demonstrate the processes of the MagicGrid architecture to model the UUV in the case study through the Problem, Solution, and Implementation Domains. The models derived from using the MagicGrid architecture are used to demonstrate the utilization of MBSE and its transitions into the Cameo DE modeling structure; they will not be used to perform a deep dive into the UUV itself. Now that the UUV and MBSE architecture has been defined and connected through ModelCenter and Cameo, the next chapter will discuss how the tutorials are created to demonstrate these concepts and guide users to developing their own models.

V. ORGANIZATION OF TUTORIALS

After a thorough description of the background, purpose, and methodologies of this study, Chapter V discusses the process of developing an architecture model of the UUV and the creation of tutorials for each step of the MagicGrid methodology. To maintain consistency, this process will be applicable to the problem and solution domains of the architecture. As noted in the previous chapter, the implementation domain was considered out of scope for this project and not included in the tutorial. The goal is to create a standardized architectural model development and tutorial process that is both detailed and concise, as well as easy to follow for all potential users.

A. CAMEO MODELING AND VIDEO RECORDING

As referenced in Chapter I, the methodology used in this project was a modified Unified Process Model (see Figure 1), which is coupled with an Agile development methodology and reiterated throughout the problem, solution, and implementation domains of the MagicGrid architecture model. First, the communication and planning steps serve to assign different tasks to group members and detail when they should start and end. Examples of these steps include designating those in the group who work directly with the Cameo software to make the model. For accuracy, these individuals also coordinate with the advisors to verify and validate each part of the architecture. These team members often perform the modeling task at least twice on the same model, as at least one modeling effort is recorded. Using MS Stream software, the modeling and recording team members record their screens and narrate the process of creating their respective part of the model. Once the videos and audio of the modeling process have finished, they are downloaded and converted into .mp4 format for editing. The modeling and recording team members submit their recording to the video editor. This process repeats for each part of the MagicGrid model which contains a part that is essential to the architecture and therefore needs a tutorial. Other parts of the model, such as repeated parts (e.g., UUV subsystems), will only be shown in one video tutorial and will be noted as a process that can be repeated for others,

as necessary. This modeling and video recording will also take place in the DE effort with other programs (e.g., Excel) to show how to link them to Model Center.

B. EDITING AND FINISHING TUTORIALS

Once the videos tutorials are recorded and converted into .mp4 format, they are sent to the video editor team member. Next, the videos receive introductory slides that display the name of the part of the architecture that they will be covering. Since thoroughly covering some parts of the architecture would take longer than 10 minutes, some of the sections are divided up into separate parts and given their own video tutorials.

In case the modeler makes a mistake in the video, they verbally say to not include that part and try again until it is correct. The video editor then removes this part of the video and makes it so that the video and audio flow smoothly. Once the video editor is satisfied that the video and audio are consistent and that all parts of the tutorial video are included, a small conclusion slide is placed at the end to include credits for any background music used. The video remains saved in .mp4 format as this provides adequate video quality and is a widely used format.

C. PACKAGING THE TUTORIALS

An additional duty of the video editor is to write up a step-by-step guide that shows the user through the tutorial. These written guides do not always follow the dictation of the modeler and recording team member verbatim but remains equally concise. Included with this written guide are screenshots of each major action performed by the modeling team member in the video.

The next step is to combine the .mp4 tutorial video, screenshots, and written tutorial into a document on MS Word. Once these have been organized with the video at the beginning and written tutorial directions immediately following, the MS Word document is converted into an PDF document. These can be downloaded as a compressed file (.zip) and accessed by the user. This can be useful if the user saves the tutorials on their computer for later use in an environment without reliable connectivity to the Internet.

This process is the final part of the sprint methodology mentioned in Chapter I. The process is repeated throughout the MagicGrid architecture until a useful UUV model is finished and there are complete tutorials for each step of the process. These tutorial packages will be organized in a folder on Box, SharePoint, or other file hosting service for download.

D. TESTING AND EDITING FOR CLARITY

The first steps of the editing for clarity take place at the modeling team member level. Each team member ensures that all steps in the modeling process are covered in the raw video before submitting it to the video editor. Next, the video editor monitors the video for flow and clarity. Once this video is complete and the tutorial packages finished, the videos are available for other team members to review. The final reviewer before the stakeholders and potential end users is the capstone advisor. Questions asked to stakeholders are in the Appendix.

E. CHAPTER SUMMARY

The modified UP Model with Agile development sprints proved to be an ideal method for organizing, planning, and executing this case study by developing models and tutorials. Its strength lies in dividing and conquering the MagicGrid architecture method and then assigning specific tasks to specific people. Once the team members were familiar with the process, we simply repeated the process throughout the remainder of the project. A major lesson in the tutorial development was finding a format that worked for all users. The final formats agreed upon were .pdf, .mp4, and .doc due to the ubiquity of these file types and their ease of use.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

In order to help further the DOD's goals to adopt and support digital engineering throughout the DOD and across the life cycle of different projects, this project was designed as a way to advance the understanding of MBSE and DE concepts for both beginning and intermediate users. While there are a few existing training courses that cover the basics of MBSE and DE, none of them cover the full spectrum of how to incorporate analysis tools into a working MBSE architecture. These important parts are scattered but never brought together into a single useful example.

A. REVIEW OF OBJECTIVES

The goal of this project was twofold: first, develop a thorough architectural model of a theoretical UUV and demonstrate how to link it to various engineering analysis software; and second, to develop a series of videos and written tutorials on how to construct this architecture model in Cameo Enterprise Architecture. The main assumption was that the users understand some core concepts of MBSE and DE but having experience in applying the concepts or in modeling was not essential. In choosing this assumption, it allowed the project scope to include useful instruction for all potential users. The next assumption was that the project is made to augment but not replace existing training. This allowed the project to be standalone and available for use when needed, including piecemeal application.

The result was a comprehensive MBSE architectural model of a relevant and realistic complex system: a UUV. This model does not have to be used specifically for designing system architecture for only a UUV but can also be applied to many other complex systems—especially unmanned systems. Breaking down each step of architecture development with the MagicGrid process allowed for an understandable flow of design and work. From this breakdown, the tutorials were made so that they would provide enough information and direction without overwhelming the user. The user could also select method(s) of delivery: high-definition video, audio, written tutorial, or any combination of these. Since preferred delivery methods depend on factors such as user preference,

equipment, and environment, the choice of options again allows for greater overall usability.

We are confident that this project has attained the original goals of developing a comprehensive MBSE model of a theoretical UUV and informative tutorials illustrating the details of the system architecture development process. The tutorials and examples using Model Center and relevant engineering analysis software have gone one step further to enhance the user's understanding of DE: while they made have been aware of the capabilities of Model Center and Cameo to link engineering analysis tools to an MBSE architecture, they now have an example on how it is executed. These deliverables have accomplished the goals of providing the user a thorough example and tutorial of how to design, create, manage, and use an MBSE architecture and implement digital engineering to conduct analysis.

B. RECOMMENDATIONS FOR FUTURE WORK

In case of other individuals or groups are curious about this project's future potential, there lies a plethora of possible continuations. First, users might apply this architecture to a real-life UUV or other unmanned system. Next, they can go into greater detail on the subsystems and perhaps some of their subcomponents. The greatest potential lies in the DE implementation. Since only a few examples of rudimentary calculations used in real-life UUV design could be used as examples in this project (e.g., buoyancy), future work may entail a complex analysis of all parts of a UUV or related system to include buoyancy, drag, power, and pressure. This analysis can be conducted via computer-aided design programs and their add-ons such as SolidWorks, SolidEdge, or Creo Parametric. Drag analysis can be performed using computational fluid dynamics software such as ANSYS Fluent. The overall push of MBSE and DE in the DOD will provide for many opportunities to use this case study as an example and groundwork for future efforts.

APPENDIX A. TUTORIALS

For this project, the MagicGrid BoK is used as a building block for the modeling of the UUV architecture. The MagicGrid approach is based on a "Zachman style matrix and is designed to guide the engineers through the modeling process and answer their questions" (Aleksandraviciene and Morkevicius 2018, 10). This tutorial applies the MagicGrid method to develop the UUV architecture. The level of detail included in this tutorial goes deeper than that illustrated in the MagicGrid BoK. All modeling concepts remain the same.

The tutorial is divided up into sections that follow the MagicGrid phases, pillars, domains, and overall structure shown below.

			PILLAR				
DOMAIN			Requirements	Behavior	Structure	Parameters	
	Problem	Black Box	B1-W1 Stakeholder Needs	B2 Use Cases	B3 System Context	B4 Measurements of Effectiveness	Specialty Engineering
		White Box		W2 Functional Analysis	W3 Logical Subsystems Communication	W4 MoEs for Subsystems	
	Solution		S1 System Requirements	S2 System Behavior	S3 System Structure	S4 System Parameters	Integrated Testing
			SS1 Subsystem Requirements	SS2 Subsystem Behavior	SS3 Subsystem Structure	SS4 Subsystem Parameters	
			...	...	...	...	Analysis
			C1 Component Requirements	C2 Component Behavior	C3 Component Structure	C4 Component Parameters	
	Implementation		I1 Physical Requirements	Software, Electrical, Mechanical			

Each section below illustrates the steps taken to complete the identified task with screen shots of the steps. The team created a corresponding video file that walks the user through each the steps. One can obtain the video files by sending an email request to Mark Rhoades (mmrhoade@nps.edu), who was one of our advisors at NPS.

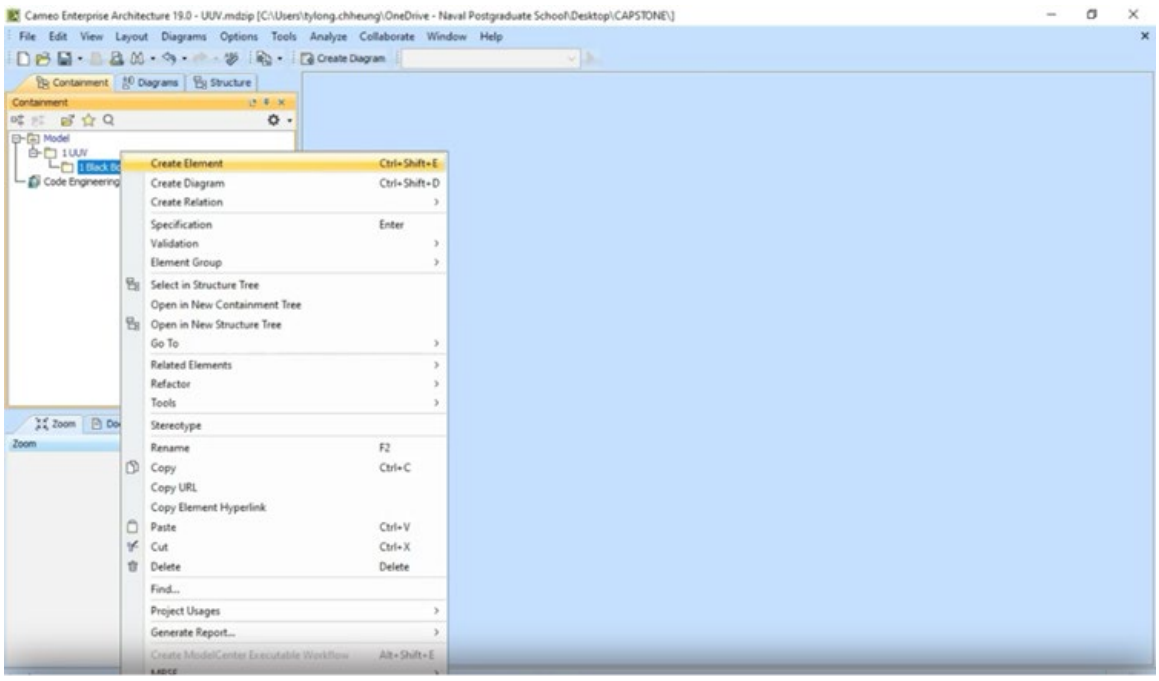
B1: Creating Stakeholder Needs Tutorial

Background:

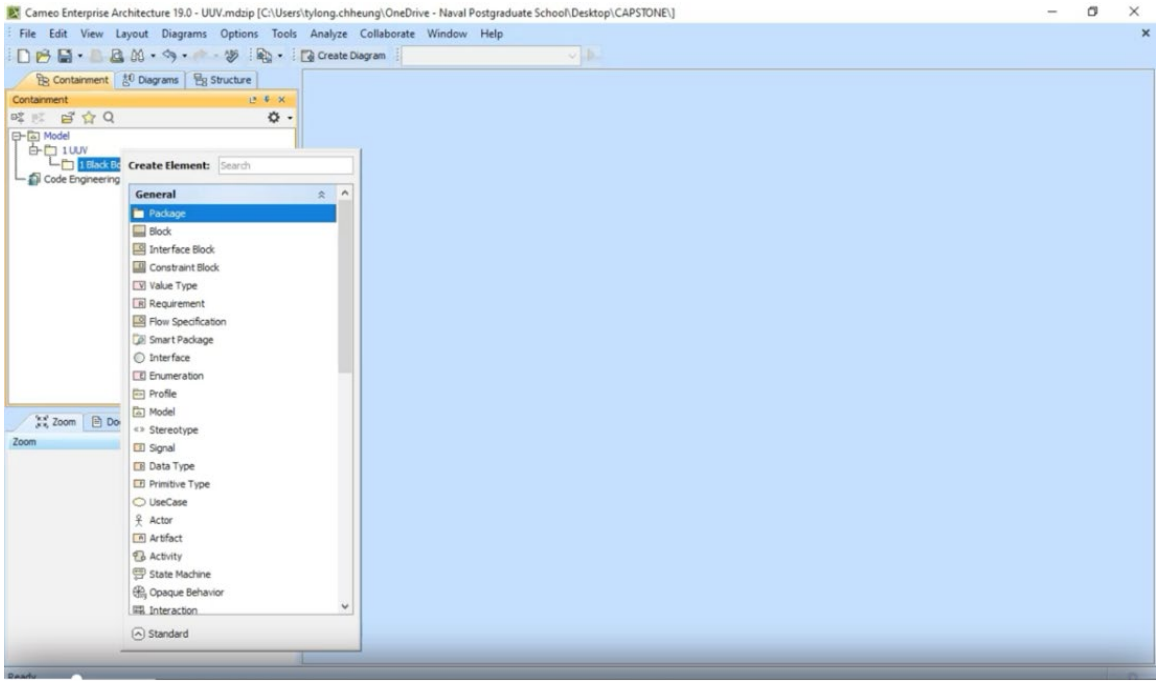
The stakeholders of the system can consist of the primary users, project teams, customers, or other parties or entities involved with product development. It is important to identify key stakeholders to ensure that the system design responds to key requirements and needs. In this section, the stakeholder needs and requirements are captured in the architectural model and are analyzed and redefined later in the process. Initial needs and requirements produce system requirements and both are used to establish traceability within the system design. The stakeholder needs in this section are captured in a SysML requirements table

Step-By-Step Guide:

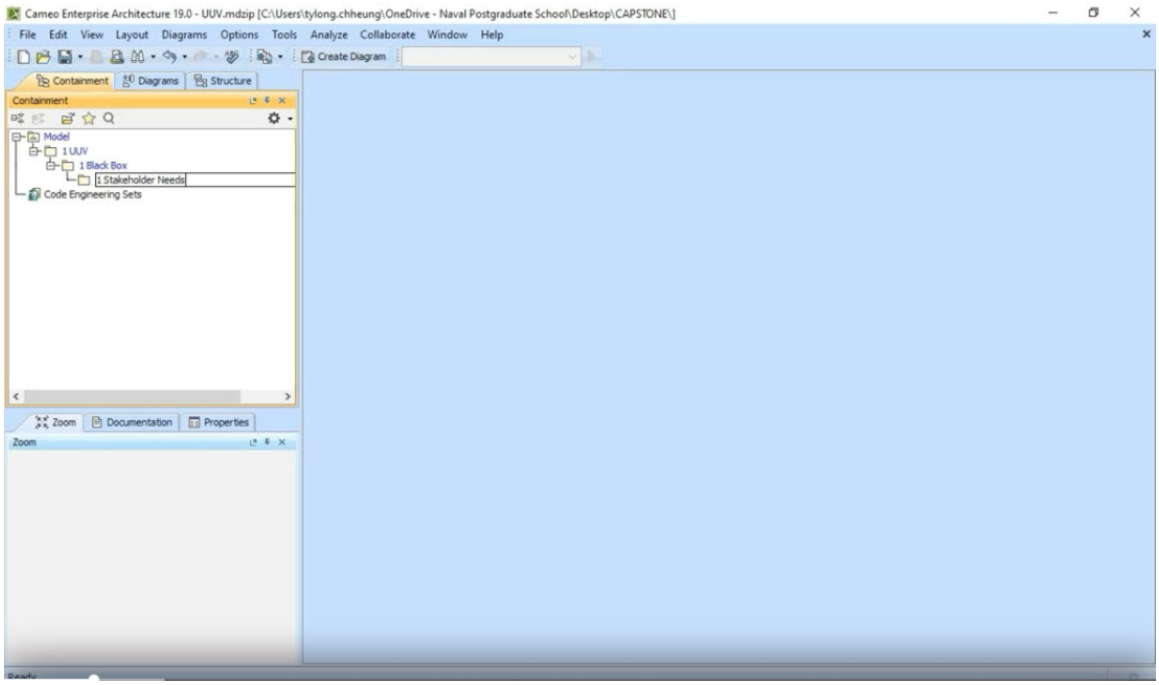
Today, we are going to cover how to do the Stakeholder Needs table for the Unmanned Underwater Vehicle (UUV) in Black Box. First, you start by highlighting and right click on Black Box and select *Create an Element*.

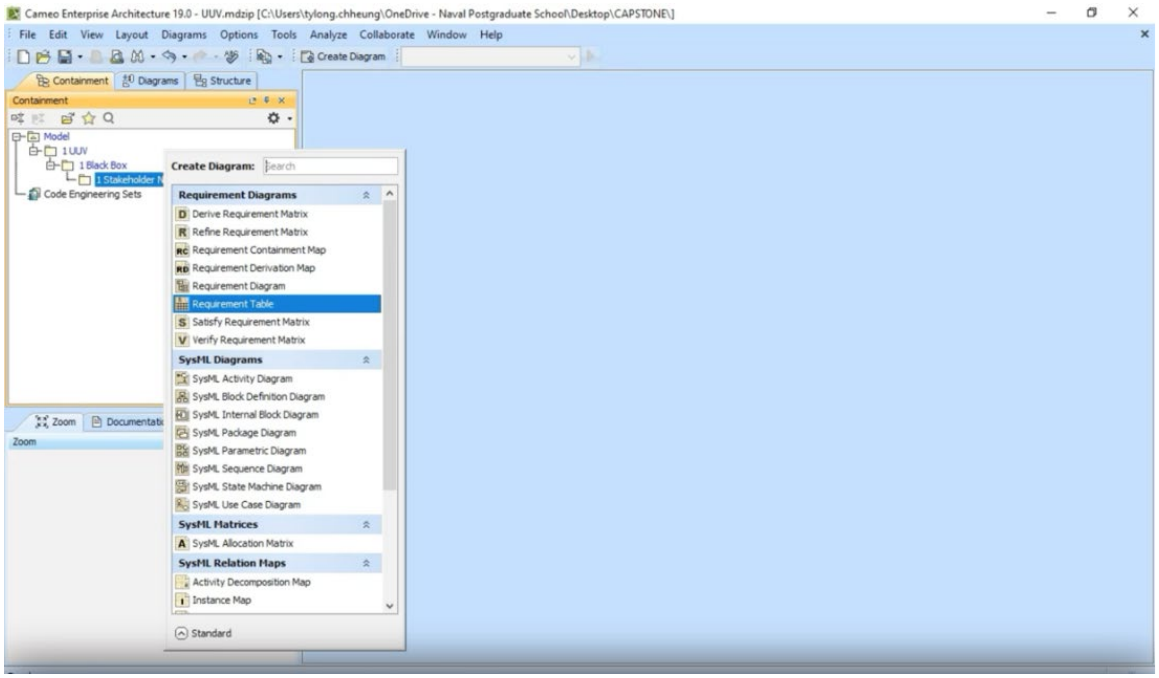


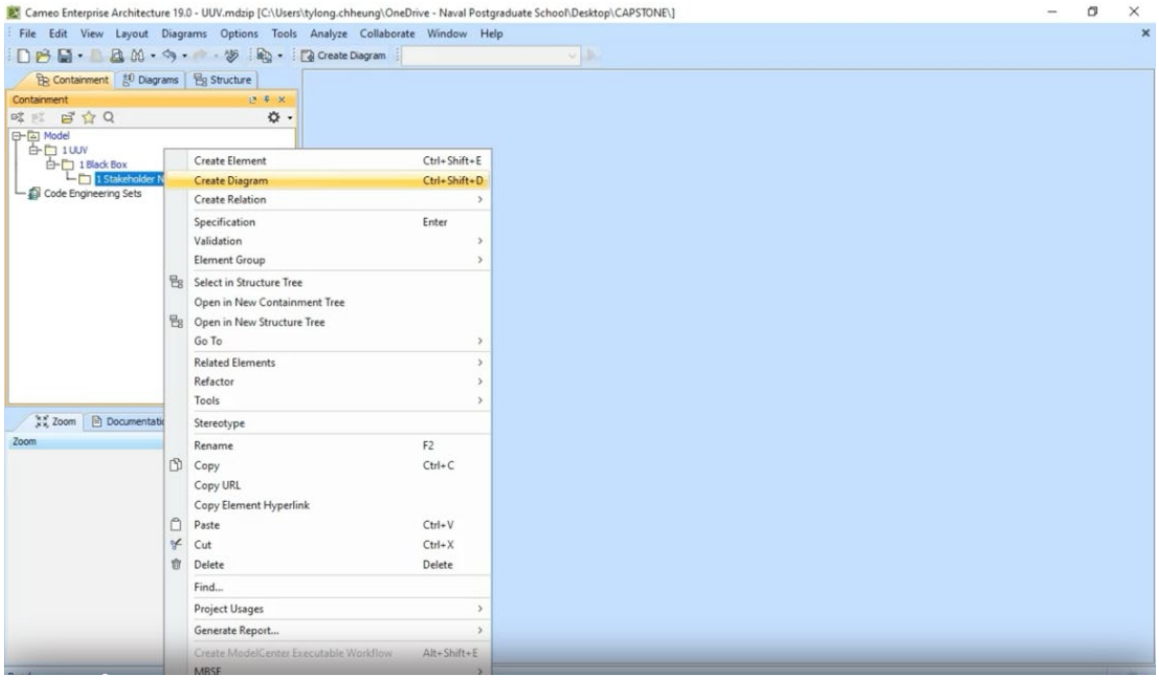
From here you click on Package and label ***Stakeholder Needs***.



Once that is completed, you highlight Stakeholder Needs and right click to select *Create Diagram*. Under the Create Diagrams tab select the *Requirements Table*.

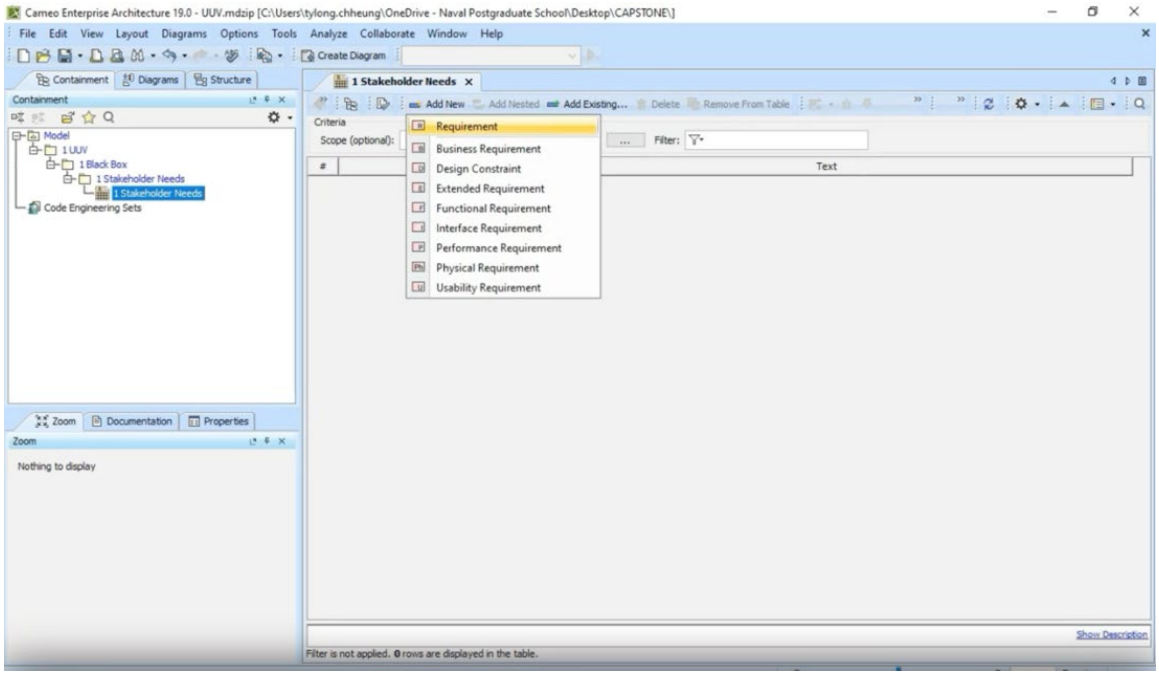


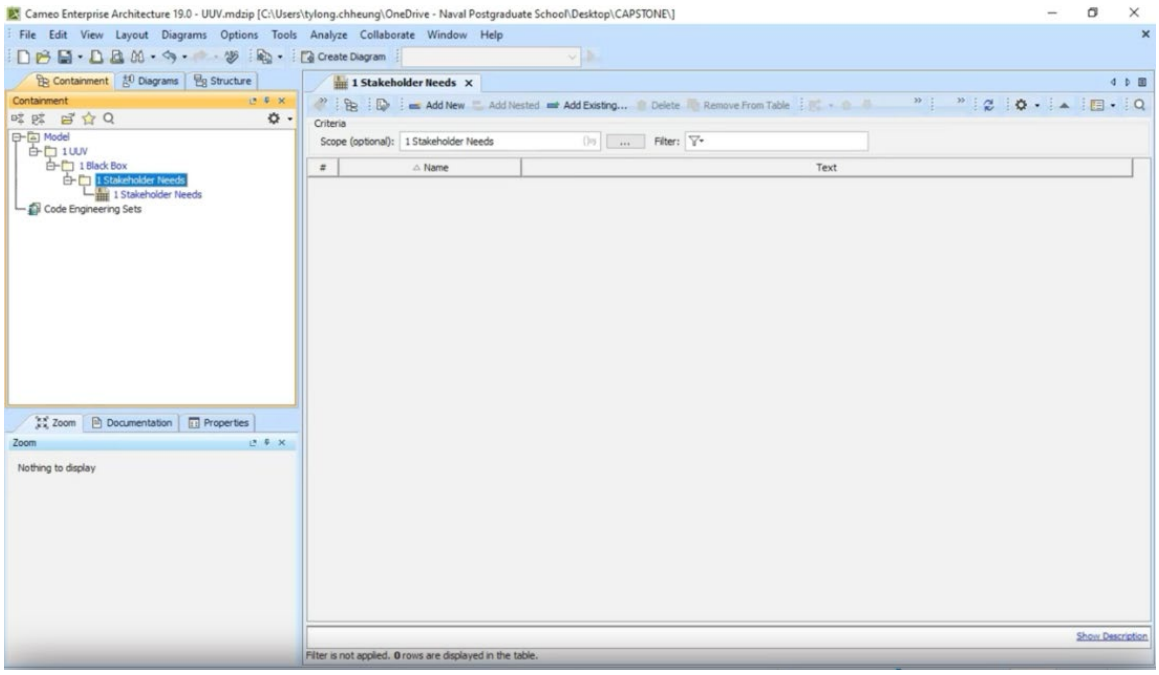




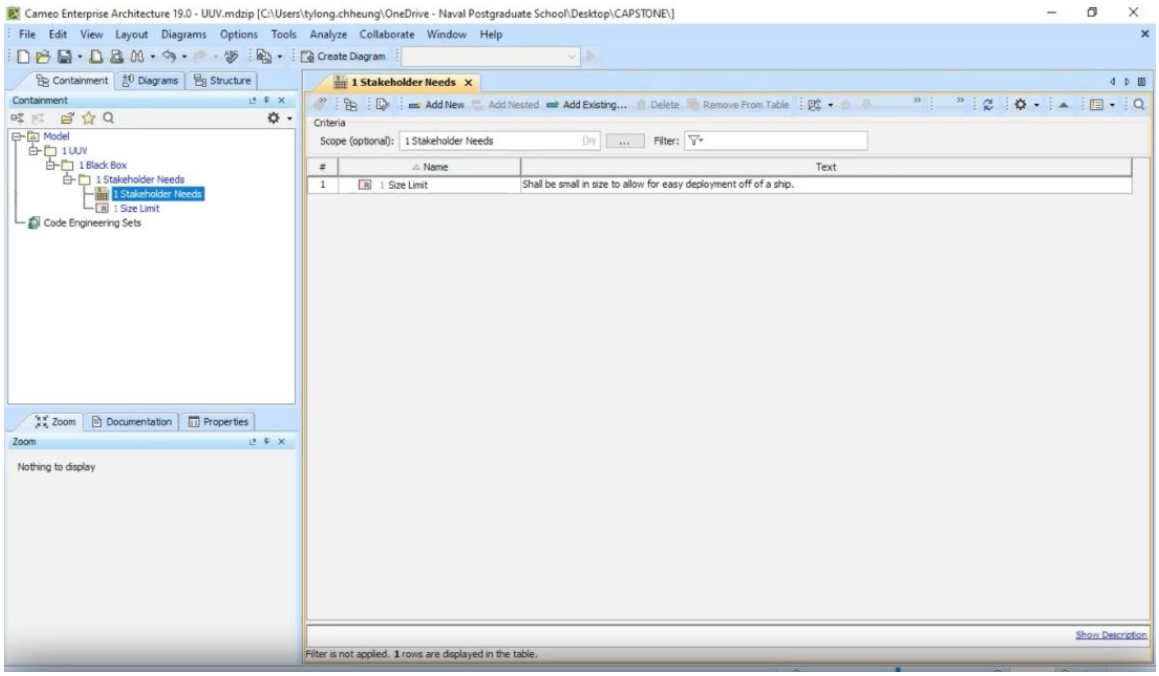
And it will build up the requirements table for you. At this point, you click and drag the package and drop it into scope. So that any future changes to the diagram will reflect in the Stakeholder Needs Table.

Now, we can start by adding our first Needs Requirement. Click on *Add New* tab and from the drop-down menu select *Requirement*.

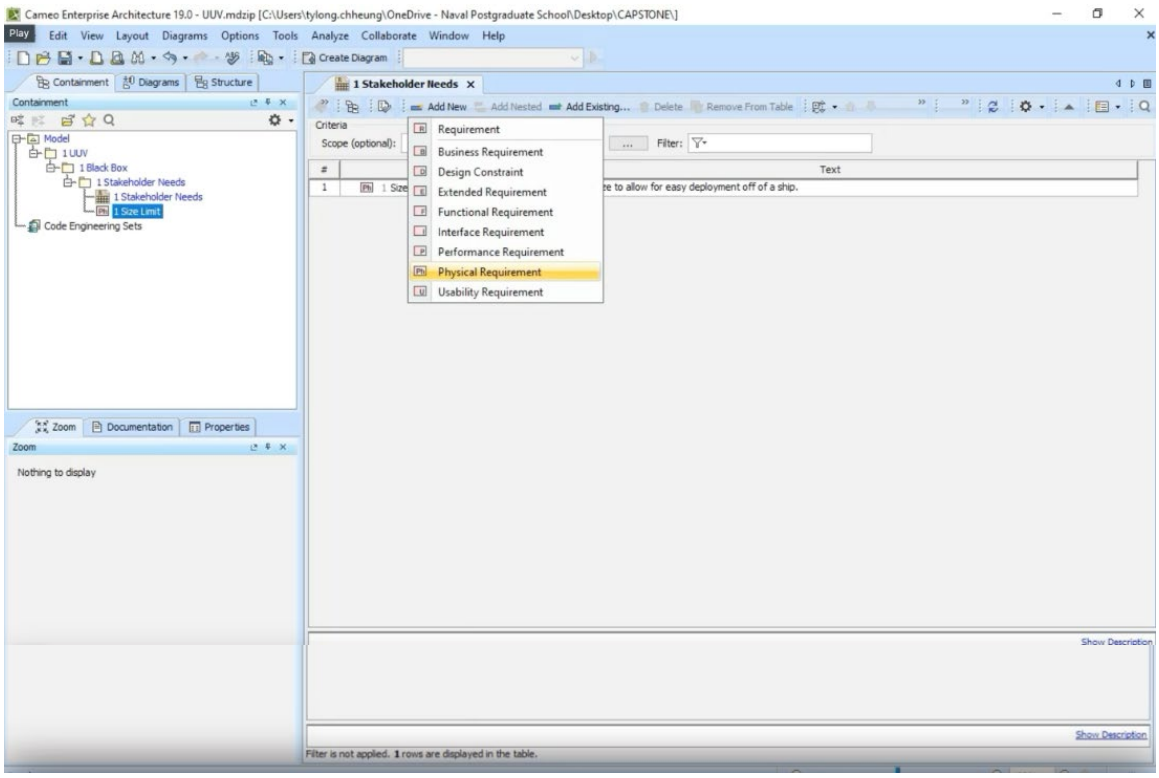




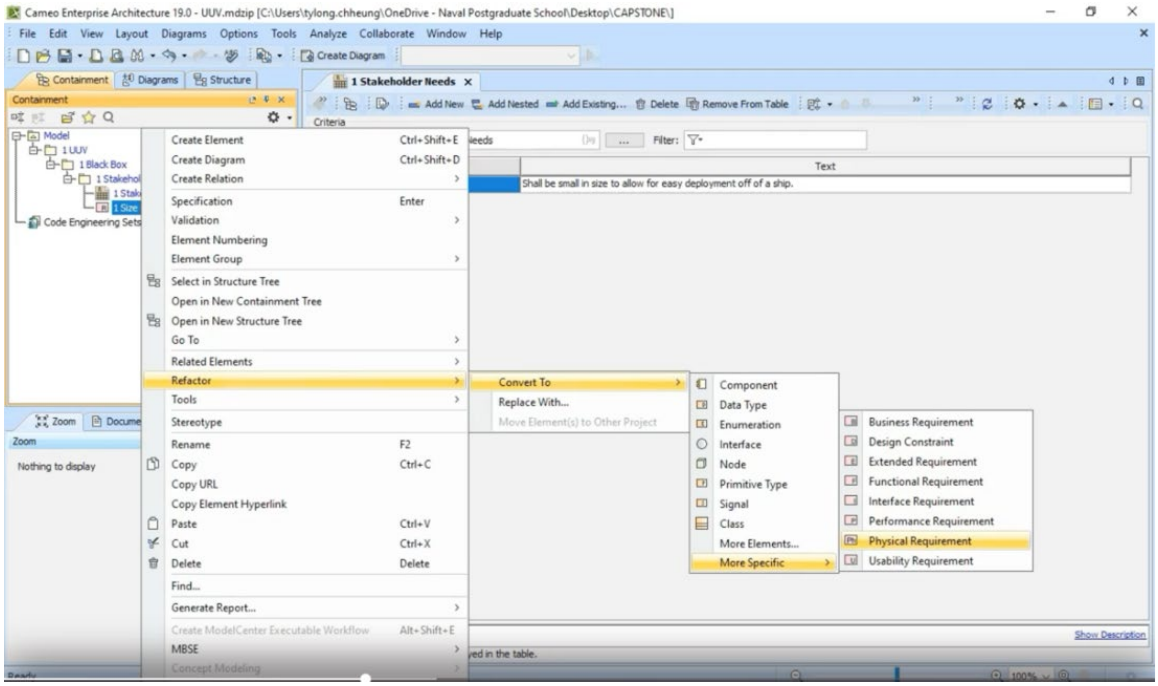
In our case, we will be including a ***Size Limit*** for the UUV, which will be entered into the Name Column. In the Text column, input the following statement: ***Shall be small in size to allow for easy deployment off of a ship***.



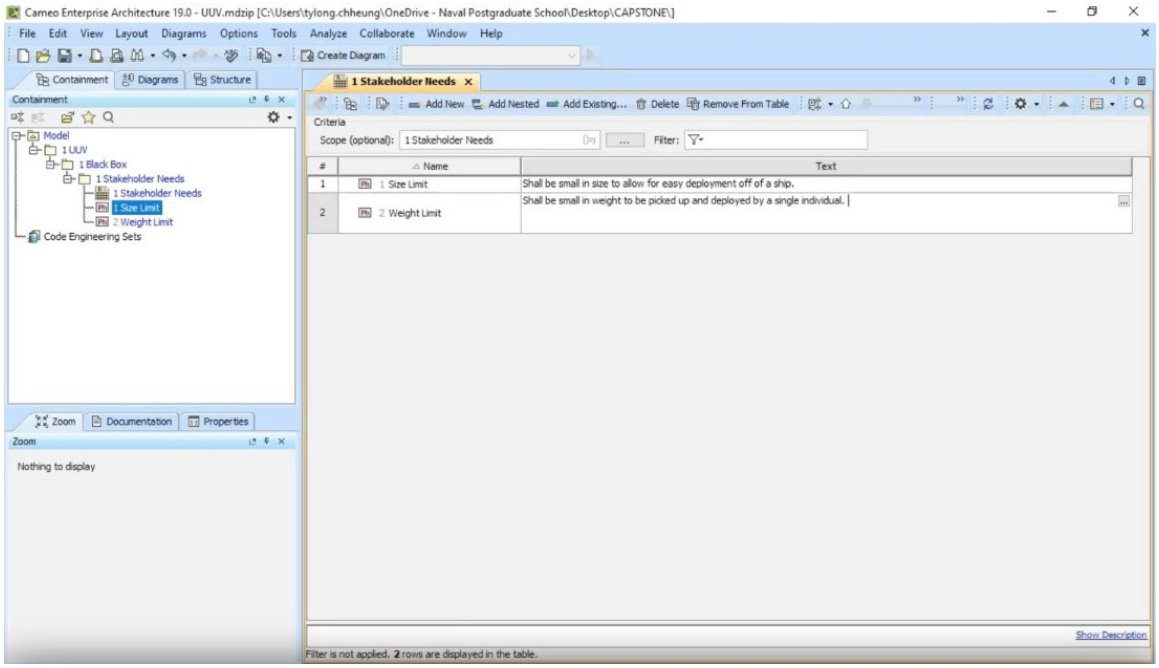
Once you completed both columns, you see that it is the first thing of our requirements. The way to allocate this as a specific type of requirement will be to right click on the requirement itself. In this case, you will highlight ***Size Limit*** and right click and go to the refactor option and select *Convert To* and then select *More Specific*. From the *More Specific* option you are able to select what kind of requirement can be or it is. In our case, ***Size Limit*** is a physical requirement, therefore, we will select the physical requirement option. And as you can see, it changed it from a regular requirement R to a PH for physical requirement.



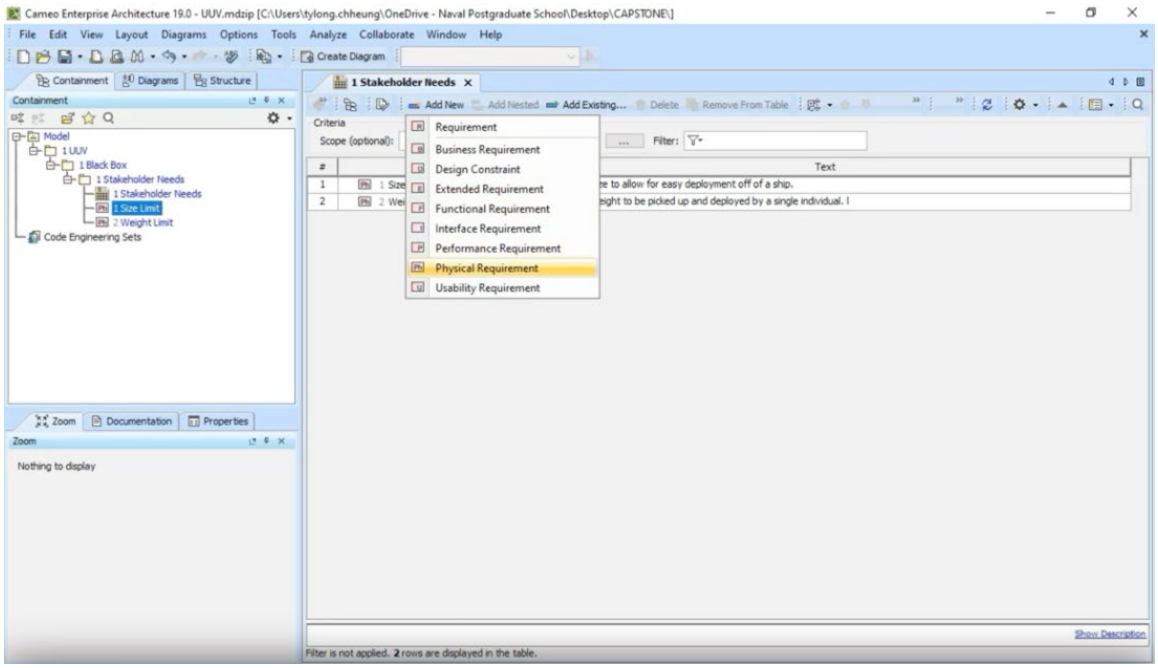
Another way to do this will also be to click on the *Add New* tab and from the drop-down menu select *Physical Requirements*.



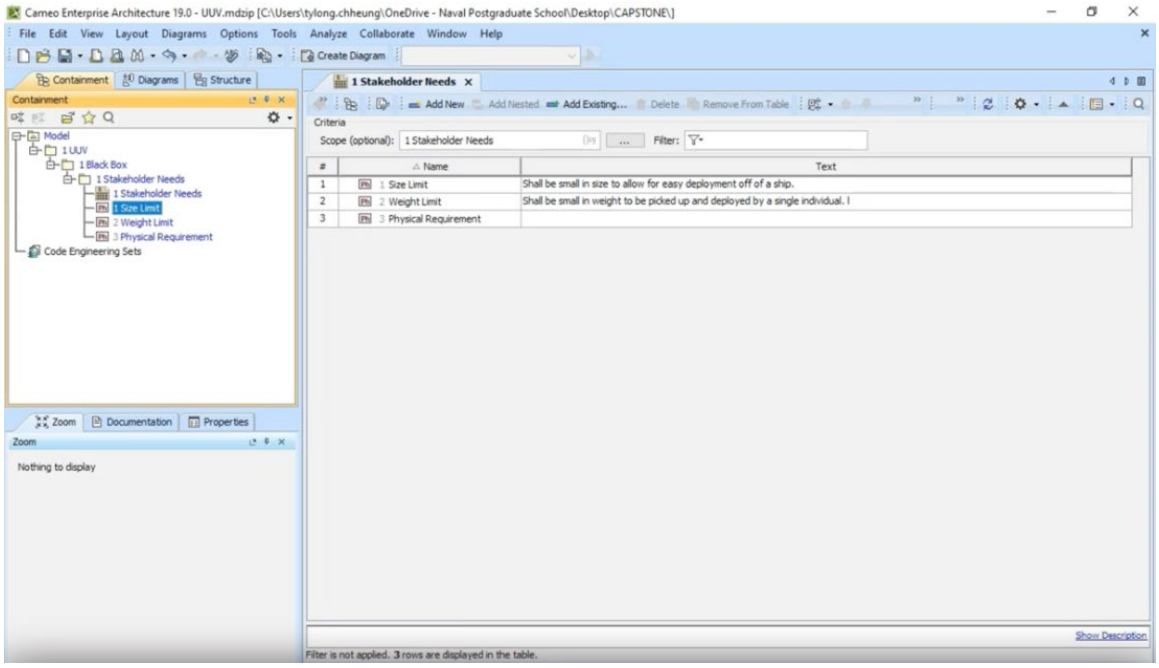
Our second Physical Requirement is the Weight Limit of the UUV. In the Name column, you type in ***Weight Limit***. In the Text column, input the following statement: ***Shall be small in weight to be picked up and deployed by a single individual***.

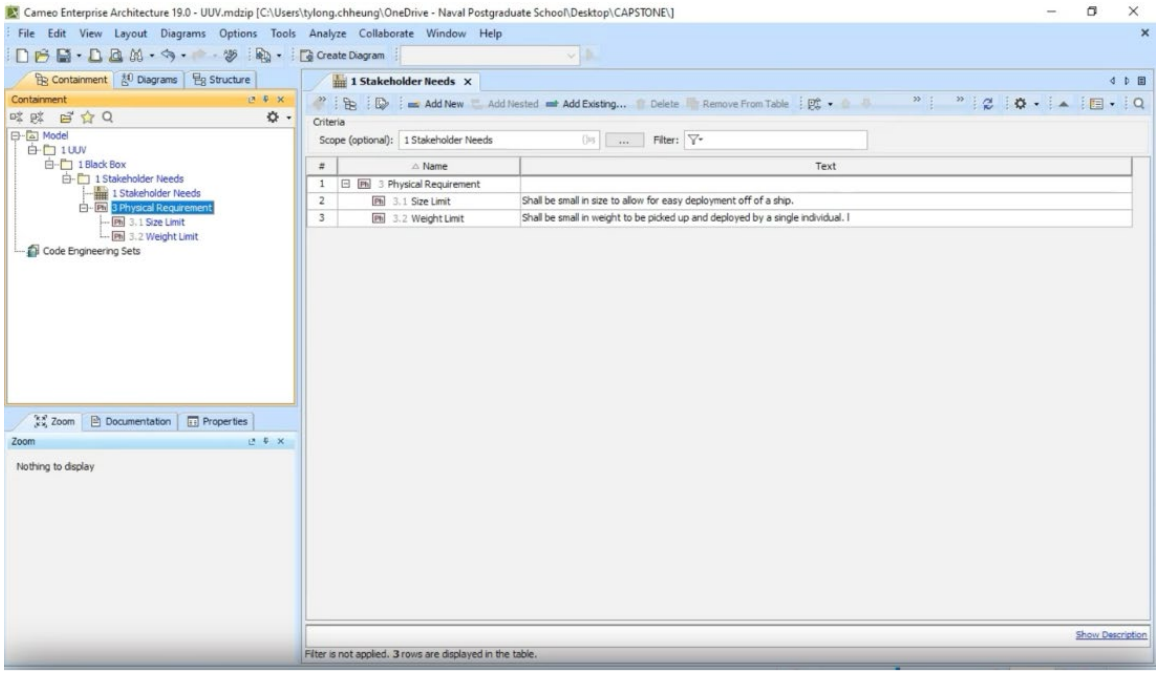


Now that we have these two Physical Requirements in order for us to categorize that as a Physical Requirement, we click on the *Add New* tab and select *Physical Requirement*. Then we can label the new row on the Name column *Physical Requirement*.

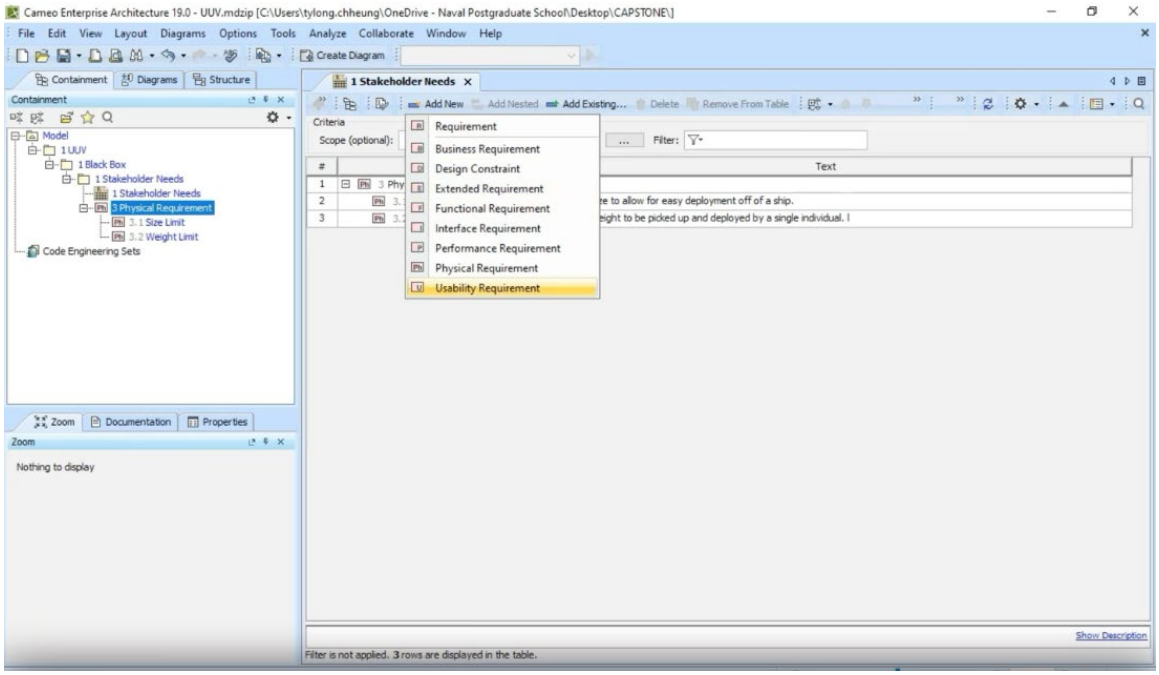


After adding its title, what we can do is to highlight both the ***Weight Limit*** and the ***Size Limit*** and drag that into the *Physical Requirement*. This will create a nest branch from the *Physical Requirement*.

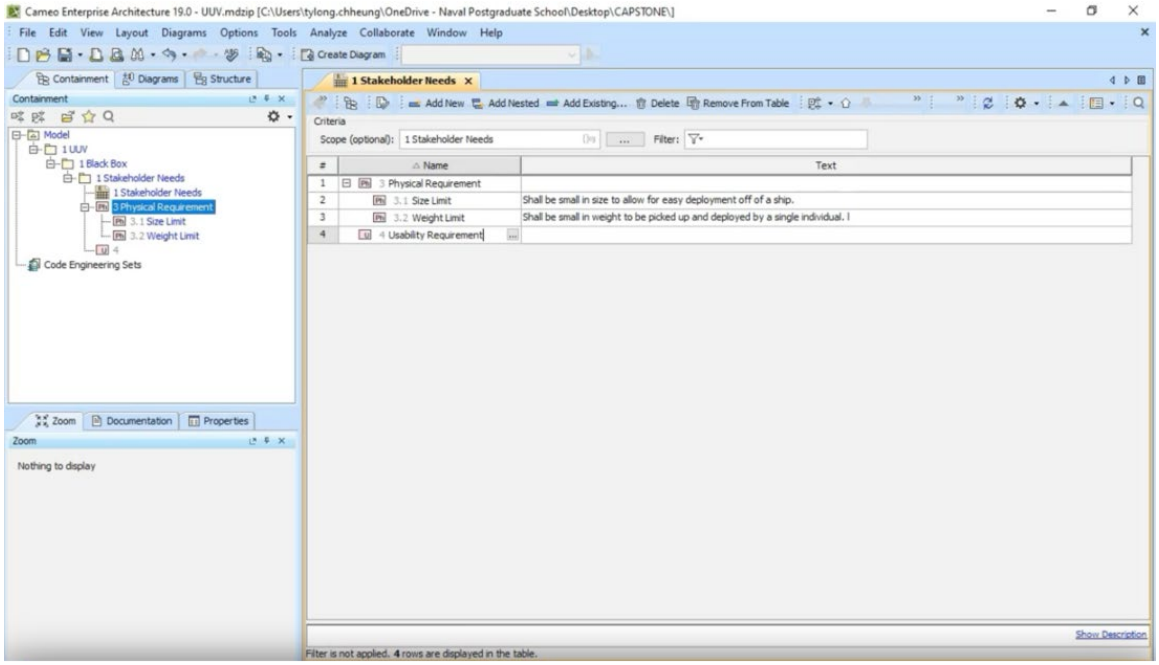


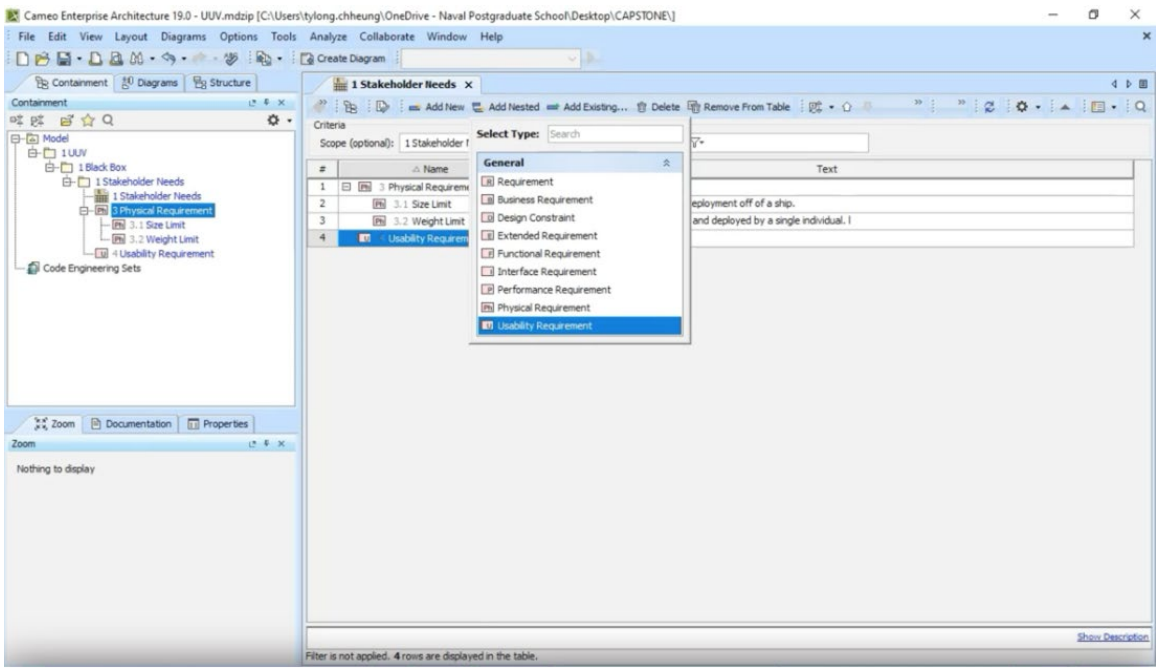


Continuing we are moving on to the Usability Requirement. Click on the *Add New* tab and from the drop-down menu select *Usability Requirement*.

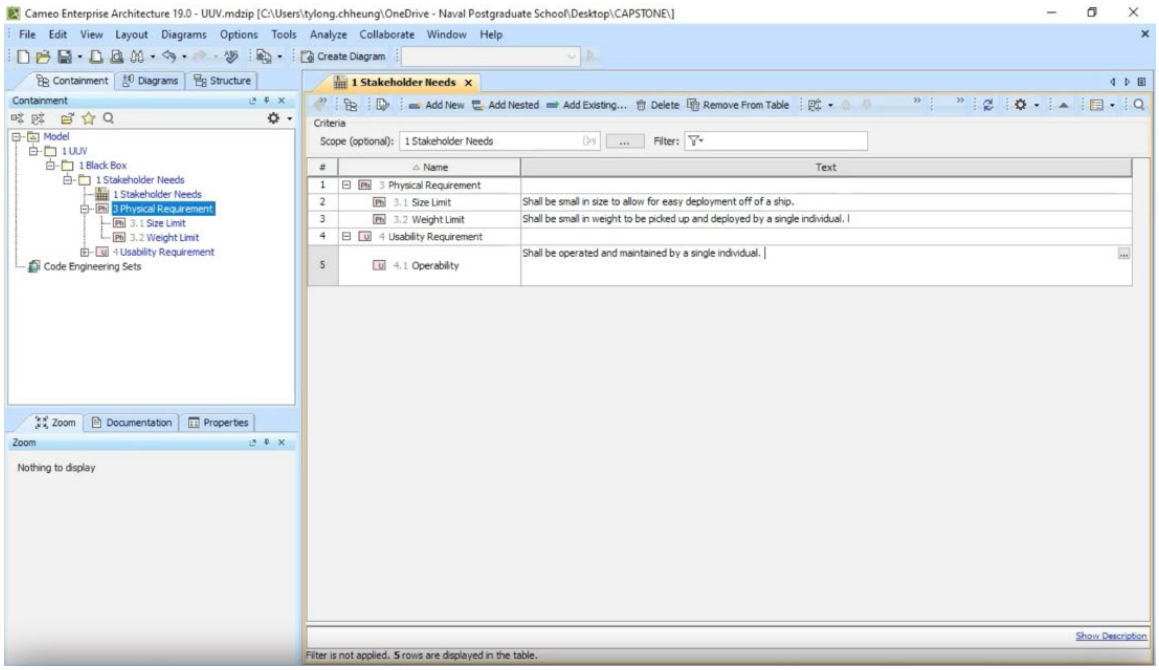


Another way we can do this is also start off with creating the title itself. Then once you have that title, highlighted and you can add a nest by clicking on the *Add Nested* and select from the drop-down menu *Usability Requirement*.

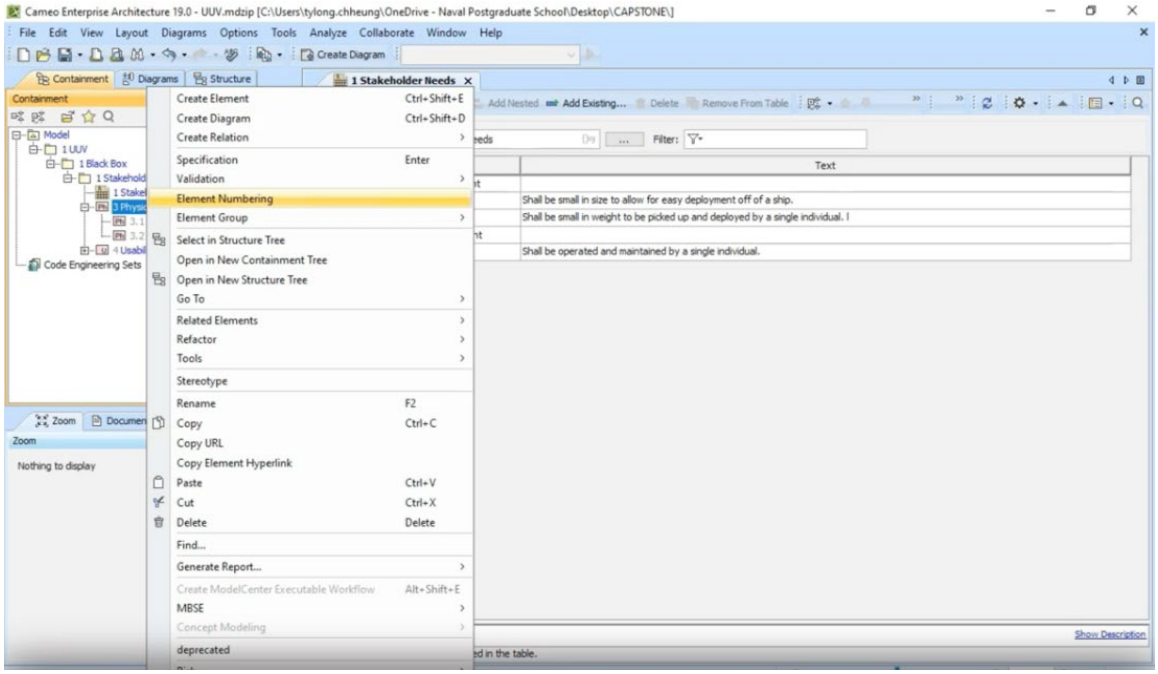




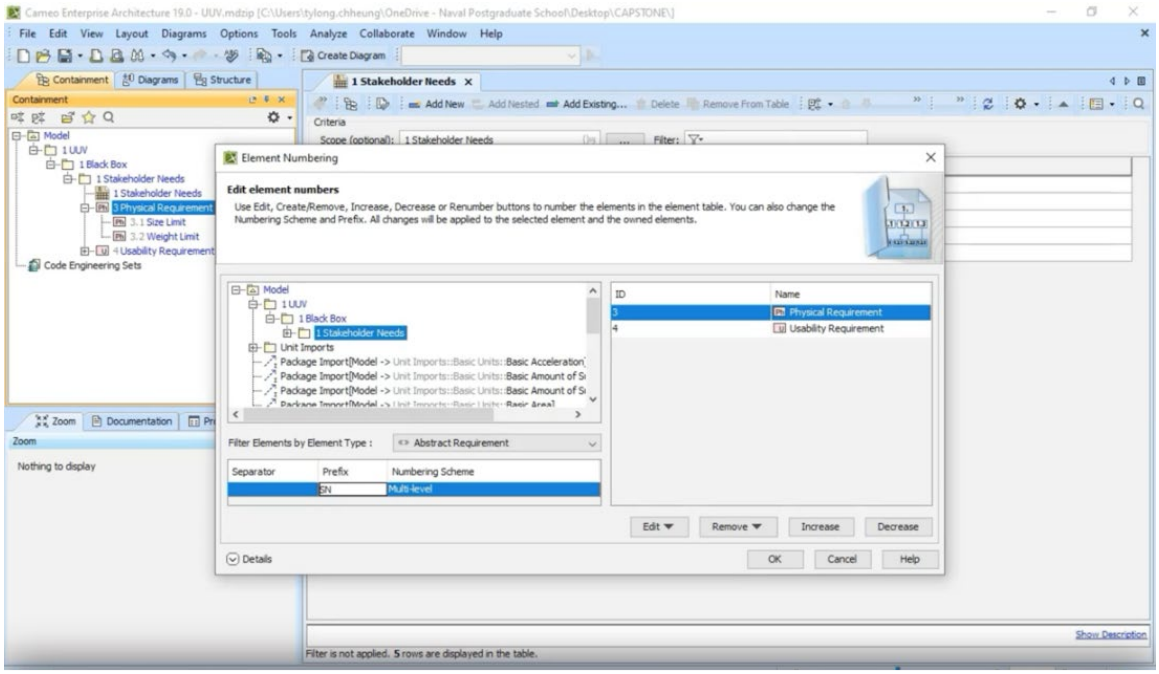
This is where we can enter the Usability Requirement for the UUV. In this case its Operability. Therefore, in the new row under the *Usability Requirement* on the Name column input ***Operability***. In the Text column, input the following statement: ***Shall be operated and maintained by a single individual***.

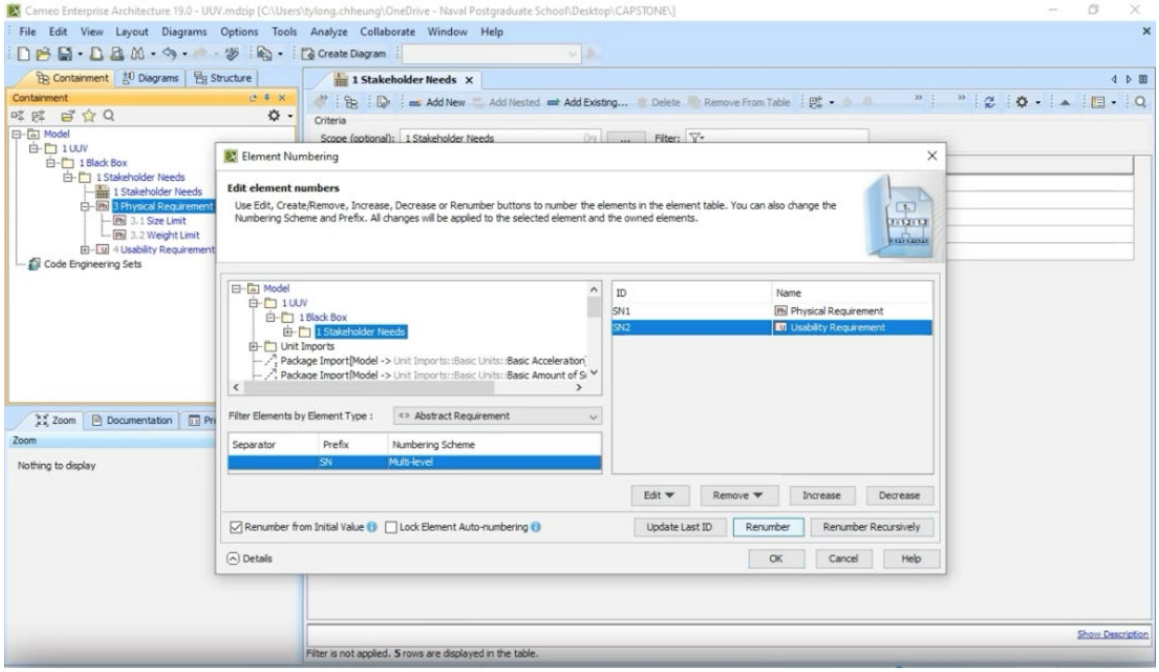


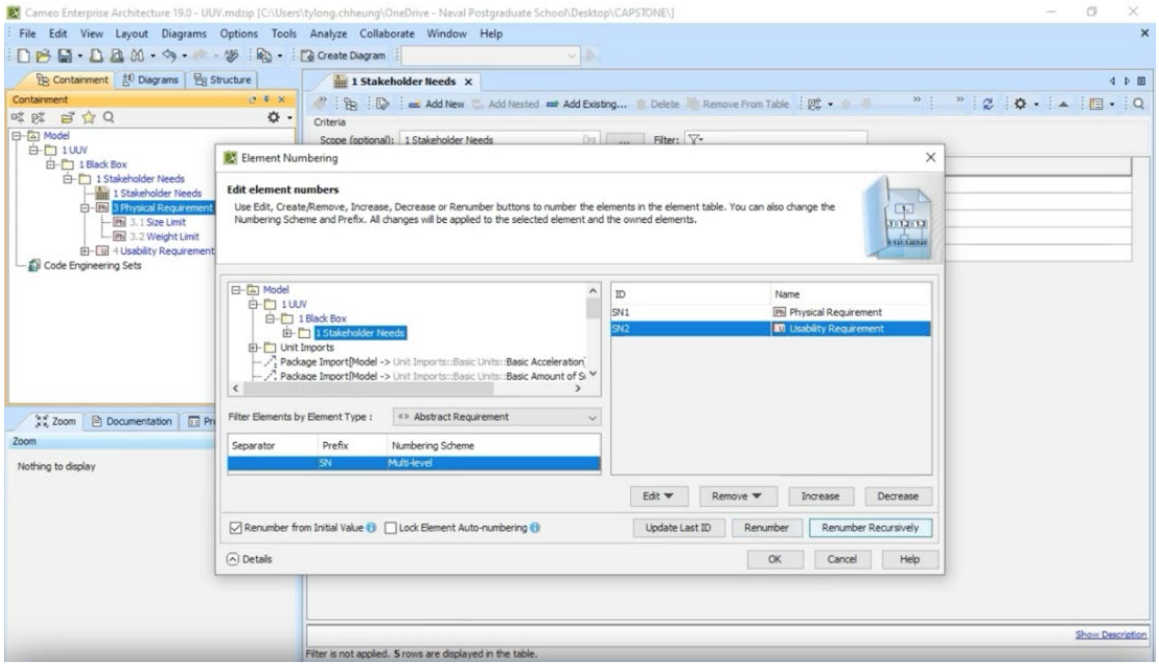
I have shown you two ways on how to create the title requirement and also the allocating and refactoring of the Physical Requirements. Also, the type of requirements from a regular requirement to its either Physical or Usability. Now let's cover the numbering sequence by right clicking on *Physical Requirement* and from the drop-down menu and select *Element Numbering*. A popup window for Element Numbering will appear on the screen and we are going to enter ***SN*** into the textbox under Prefix for Stakeholder Needs.



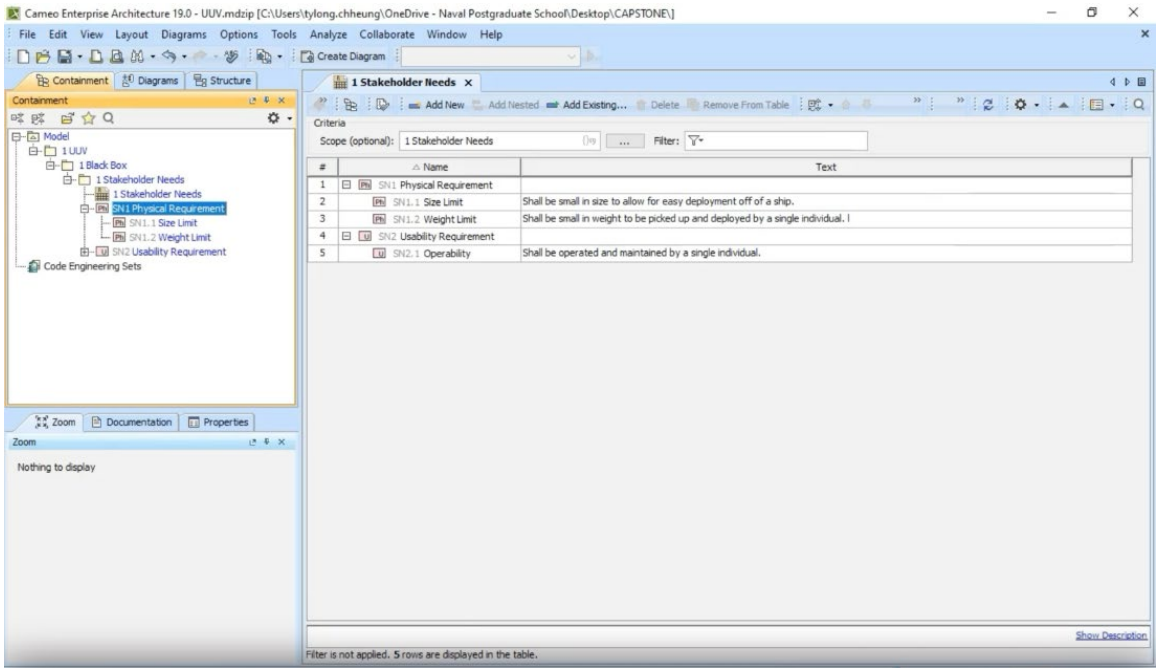
Then select the *Renumber from Initial Value* box. Then restart the number from one and click on the *Renumber* button and then click on the *Renumber Recursively* button.



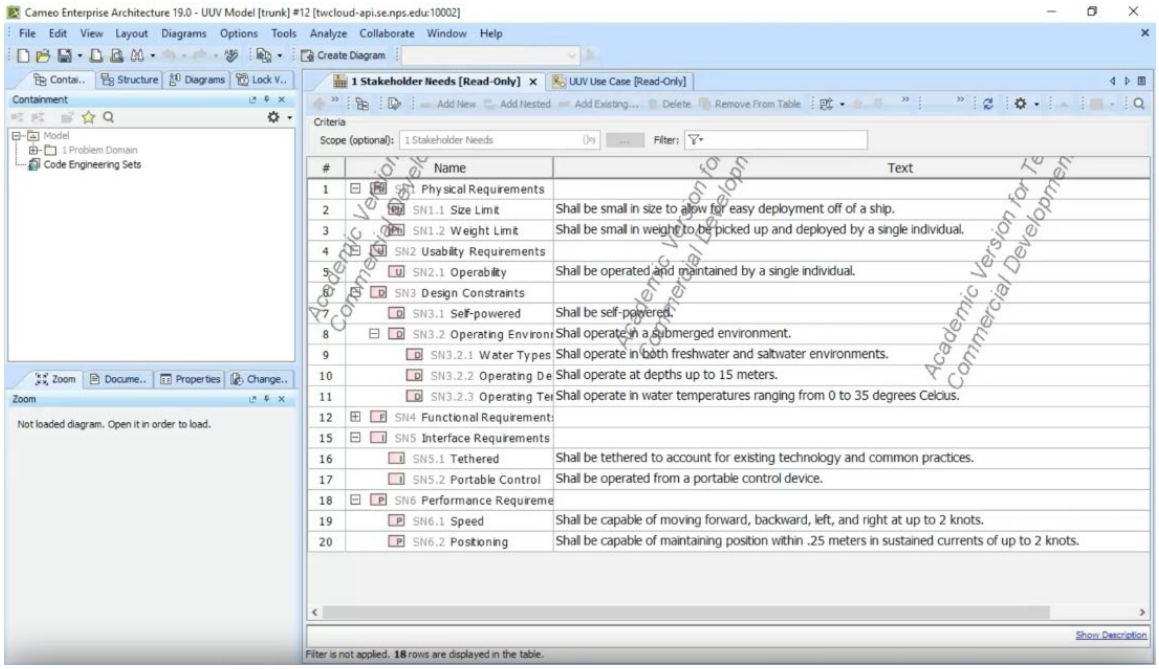

Element Numbering
Edit element numbers
Use Edit, Create/Remove, Increase, Decrease or Renumber buttons to number the elements in the element table. You can also change the Numbering Scheme and Prefix. All changes will be applied to the selected element and the owned elements.
Filter Elements by Element Type :
Abstract Requirement
Separator
Prefix
Numbering Scheme
Multi-level
Edit
Remove
Increase
Decrease
Renumber from Initial Value
Lock Element Auto-numbering
Update Last ID
Renumber
Renumber Recursively
Details
OK
Cancel
Help


Element Numbering
Edit element numbers
Use Edit, Create/Remove, Increase, Decrease or Renumber buttons to number the elements in the element table. You can also change the Numbering Scheme and Prefix. All changes will be applied to the selected element and the owned elements.
Filter Elements by Element Type :
Abstract Requirement
Separator
Prefix
Numbering Scheme
Multi-level
Edit
Remove
Increase
Decrease
Renumber from Initial Value
Lock Element Auto-numbering
Update Last ID
Renumber
Renumber Recursively
Details
OK
Cancel
Help

Then click on the *Ok* button to close the Element Numbering popup window. You will see that your requirements were renumber as ***Stakeholder Needs 1 (SN1)*** and ***Stakeholder Needs 2 (SN2)***.



Repeating the steps will generate you a Stakeholder Needs requirement table looking like the figure below.



This is how the final product would look like and has covered Physical Requirements, Usability Requirements, Design Constrains, Operating Environment, Functional Requirements, Interface Requirements, and Performance Requirements. Anything else and other requirements needed can be created via the two different steps that were shown in this tutorial video. Thank You.

B1-W1: Final Stakeholder Needs Tutorial

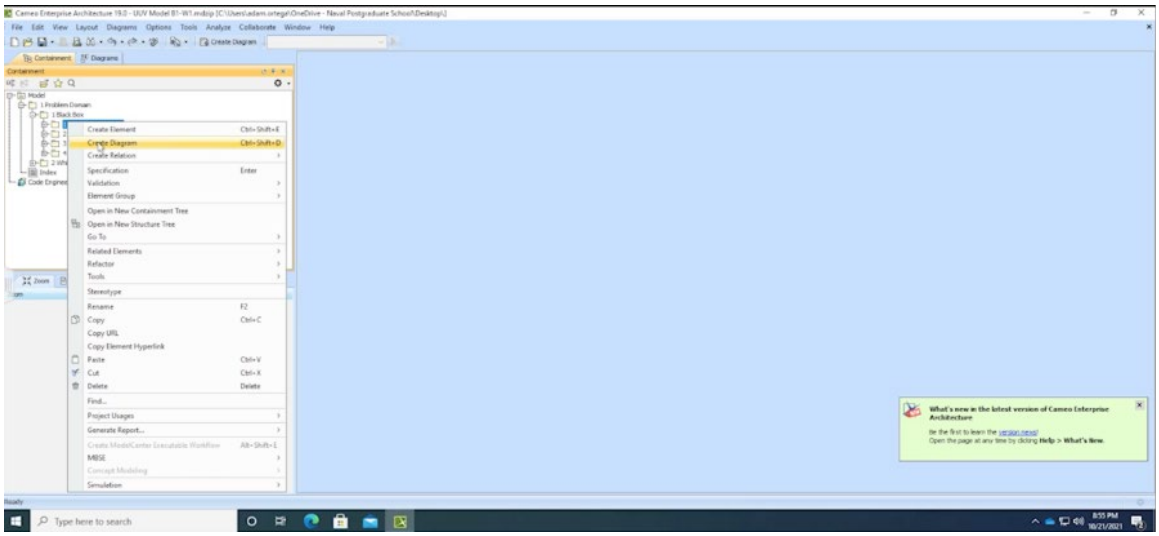
Background:

Traceability relationships need to be established to complete the problem domain. These relationships ensure that stakeholder needs are met by the various functions, logical architecture, and non-functional characteristics of the system of interest. A Refine Requirement Matrix is created to connect refine relationships between the stakeholder needs and elements of the logical architecture.

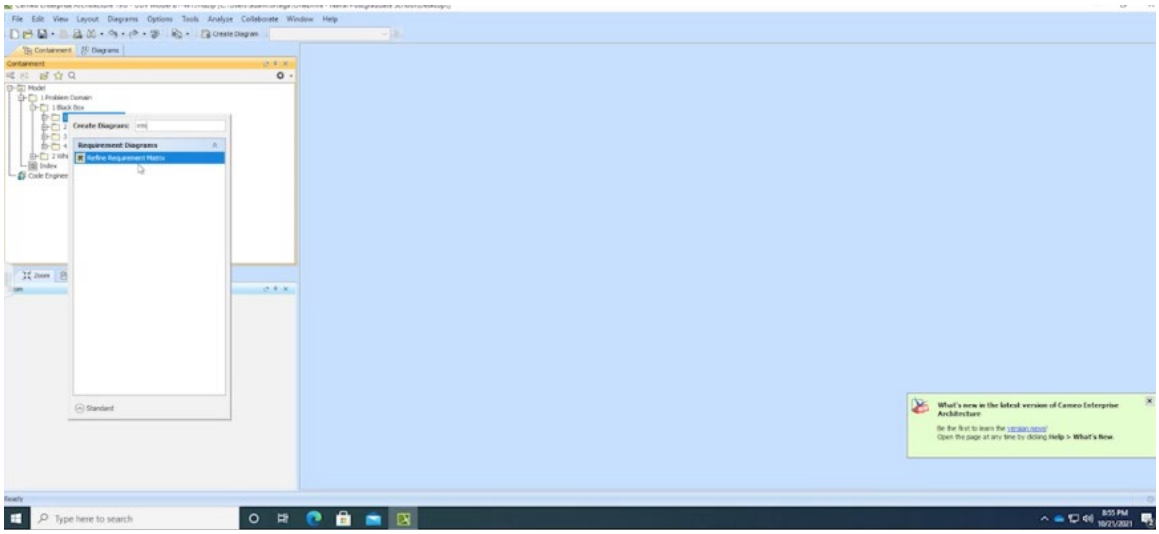
Step-By-Step Guide:

In this tutorial we will be completing the final phase of the Stakeholder Needs by creating traceability relationships between the stakeholder needs and the element that capture these. First, we will go ahead and expand the *Problem Domain* package and the

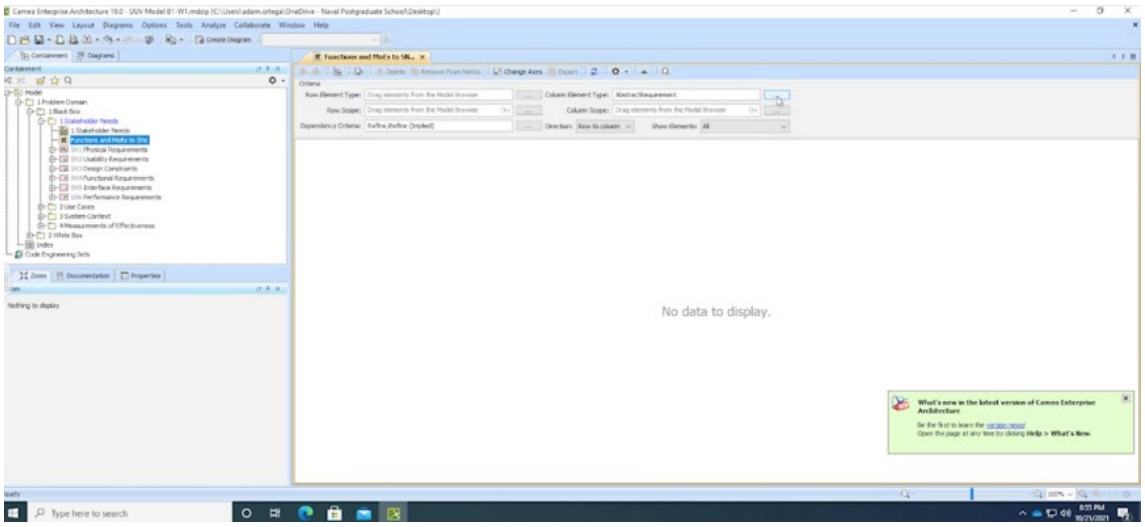
Black Box package. Then we will right click the *Stakeholder Needs* package and from the drop-down menu we will select *Create Diagram*.



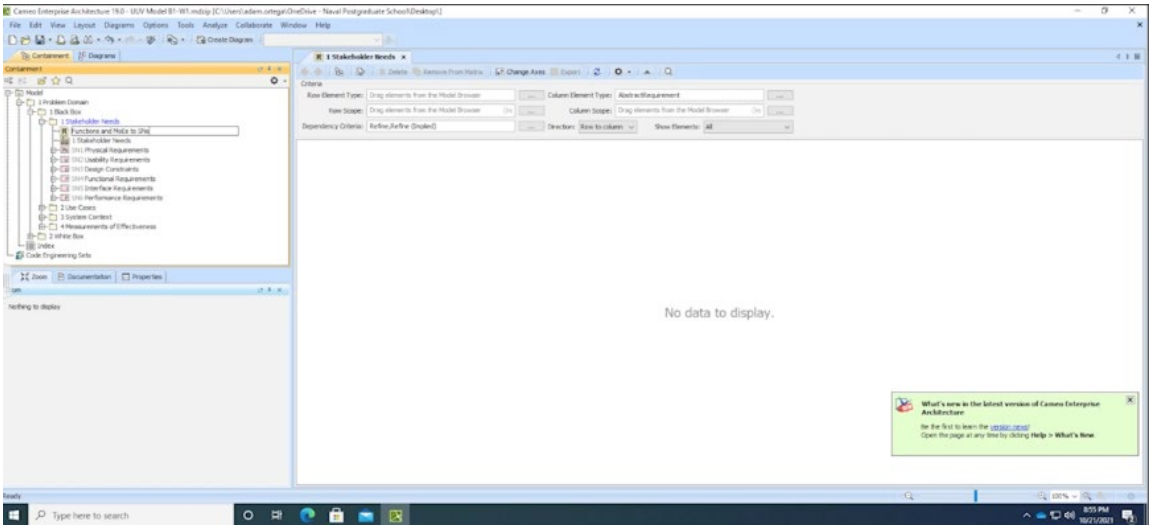
This will then bring up the *Create Diagram* popup menu and in the search text box we will type *rrm* and if nothing comes up then click on the *Expert* button located on the bottom left corner of the *Create Diagram* popup menu. The menu will then change from standard to expert and we are now able to see and select *Refine Requirement Matrix* option.



Then we will label the Requirement Matrix as *Functions and MoEs to SNs* and hit enter on the keyboard.

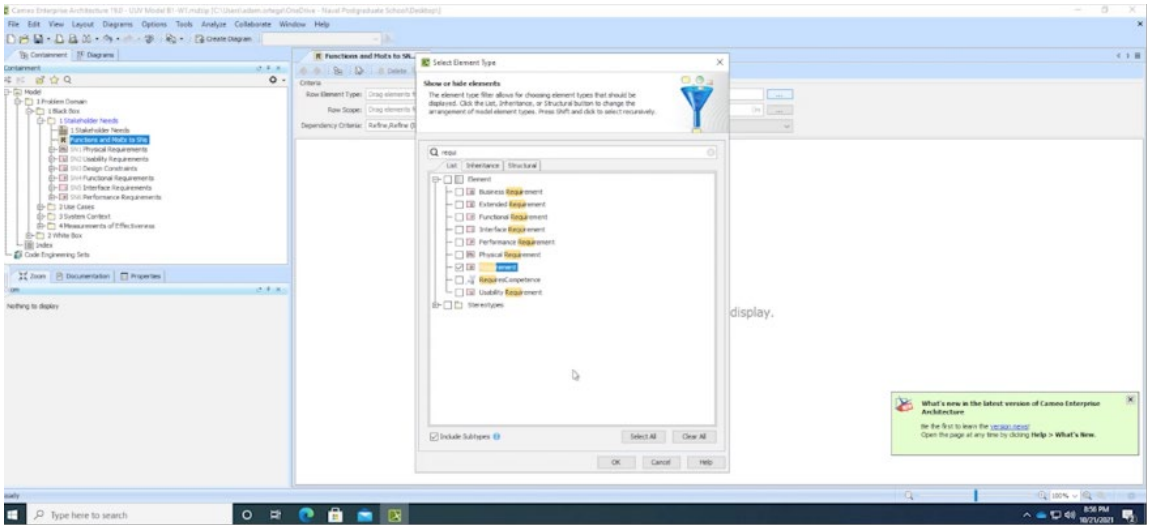


Now we can specify the *Column Element Type* by clicking on the button showing (…)

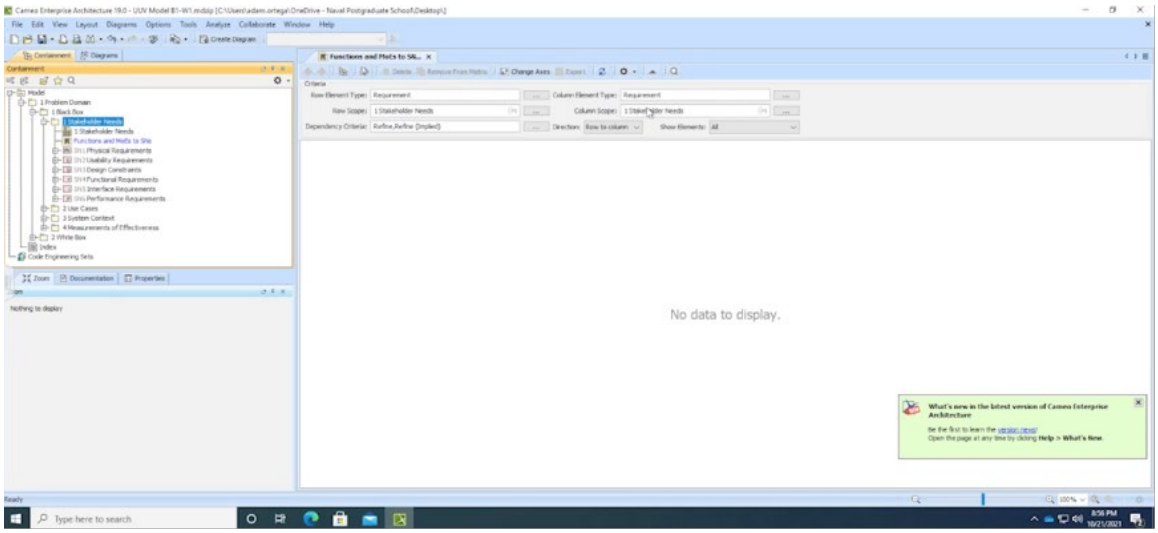


Then the *Select Element Type* popup window will appear. On the bottom of the *Select Element Type* popup window, we will click on the *Clear All* button. Then in the

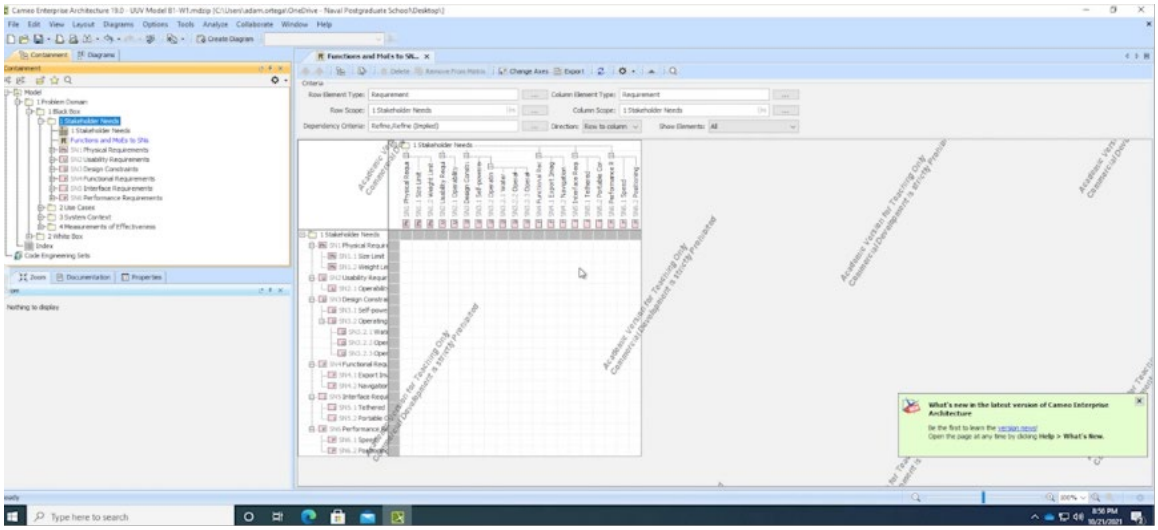
search text box we will type *requi* and select *Requirement* from the search menu and click the *Ok* button.



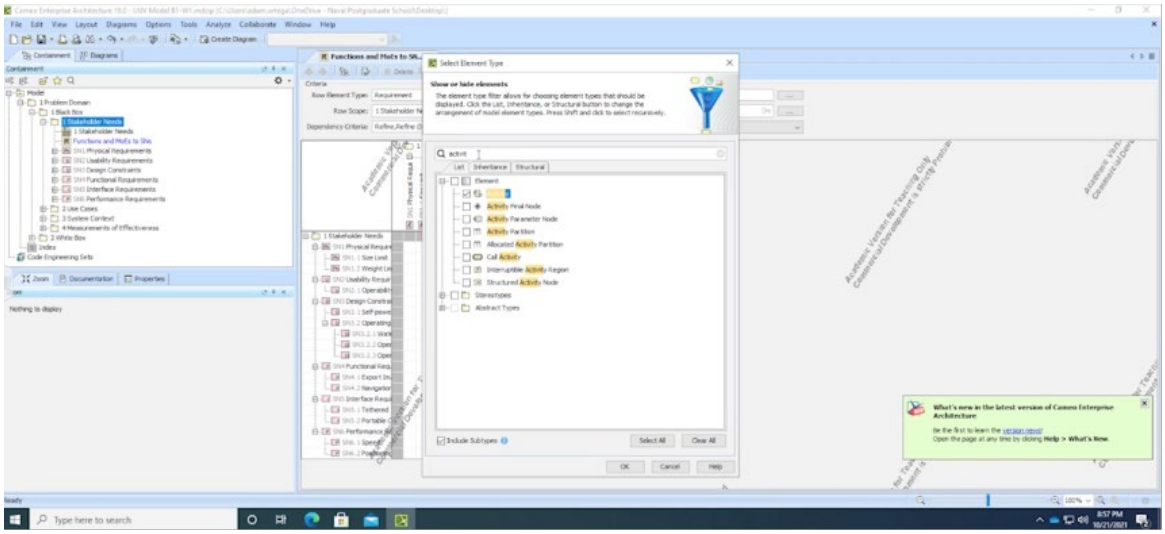
Now in the model browser we will select the *Stakeholder Needs* package and we will drag it to the *Column Scope*.



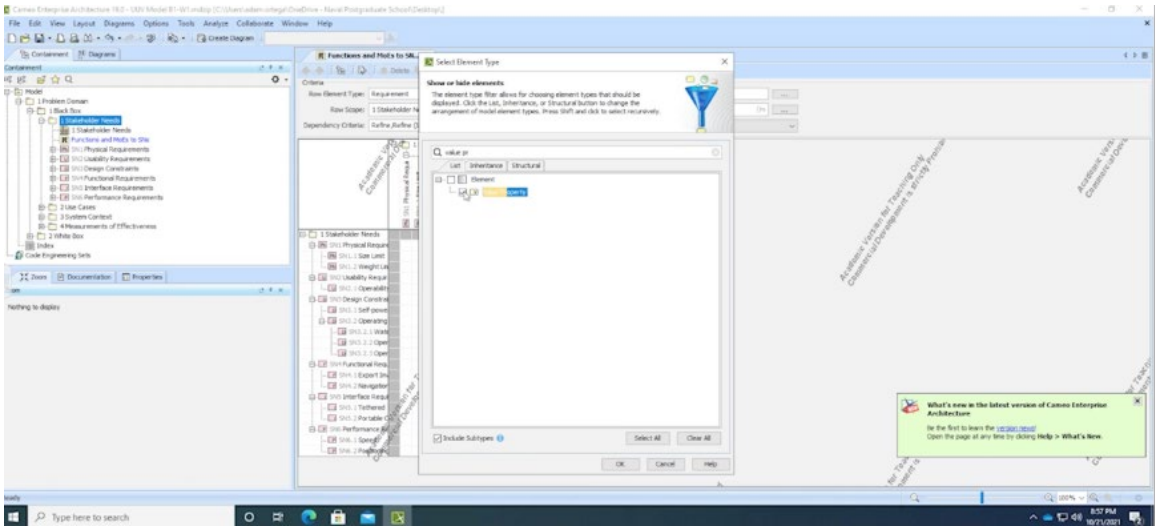
After click on the empty white space to generate the *Refine Requirements Matrix* and we can see that it is there.



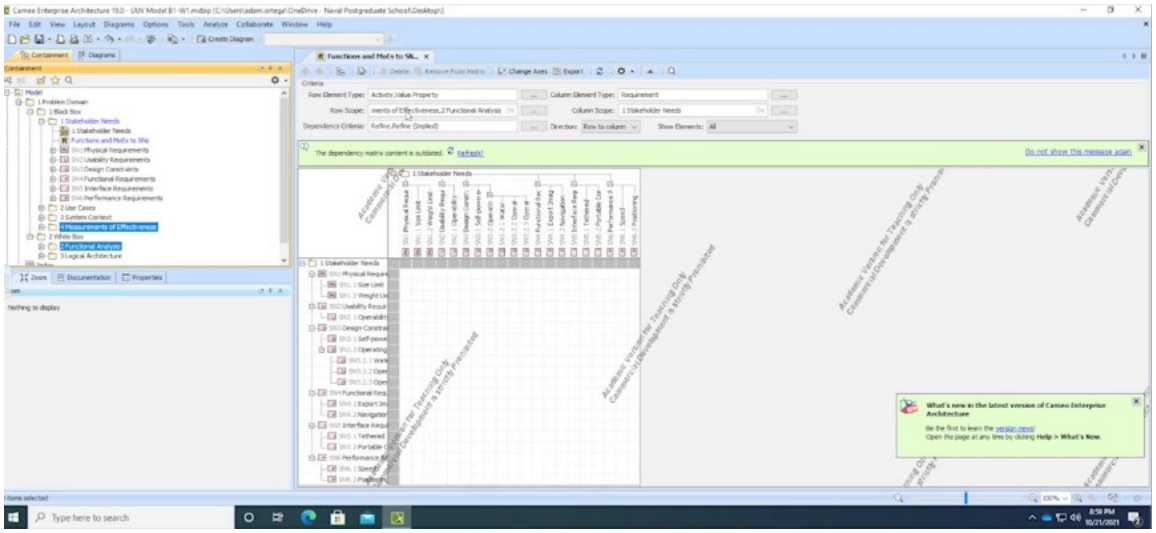
Next, we will specify the *Row Element Type*. Now again we will click on the (…) button and the *Select Element Type* popup window will appear again. Click on the *Clear All* button. Then in the search text box we will type *activity* and select *Activity* from the search menu.



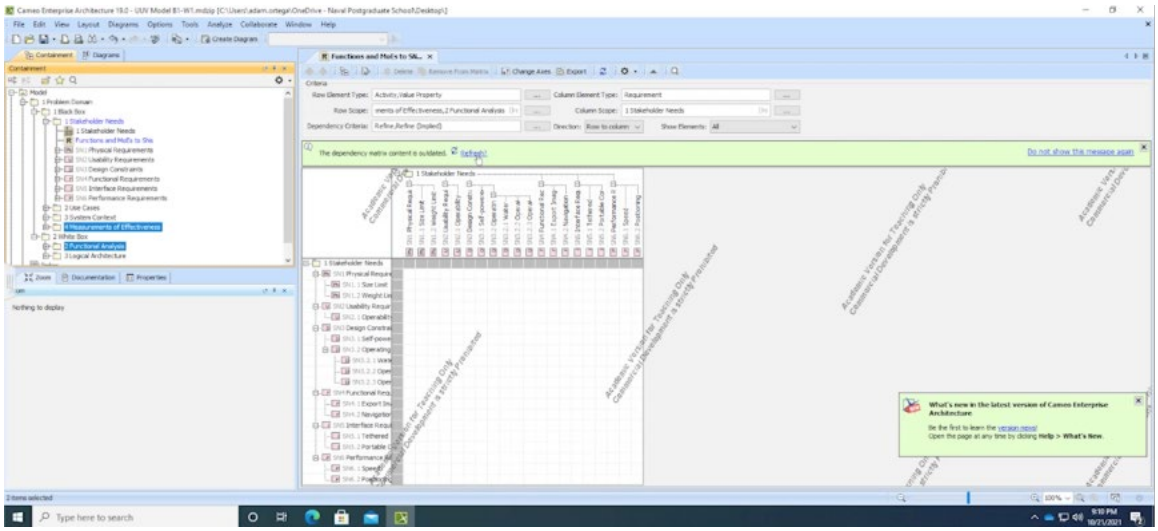
Then in the search text box we will type *value pr* and select *Value Property* from the search menu and hit on the *Ok* button.



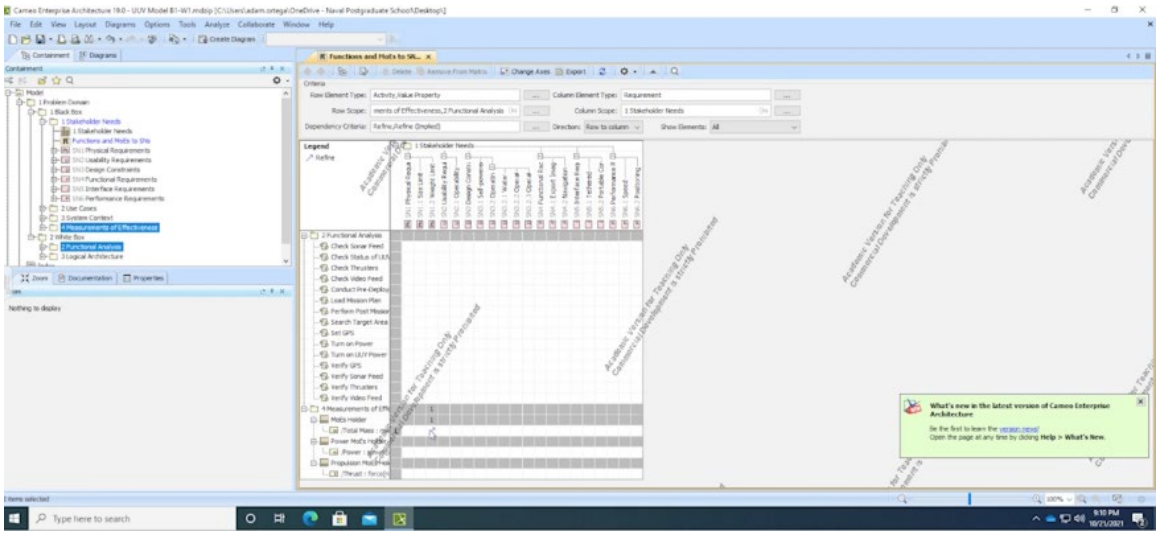
Now we will be selecting two packages from the model browser. The first one will be *4 Measurements of Effectiveness* and hold and *CTRL* and expand the *2 White Box* to select *2 Functional Analysis*. We will then drag both of them to the *Row Scope*.



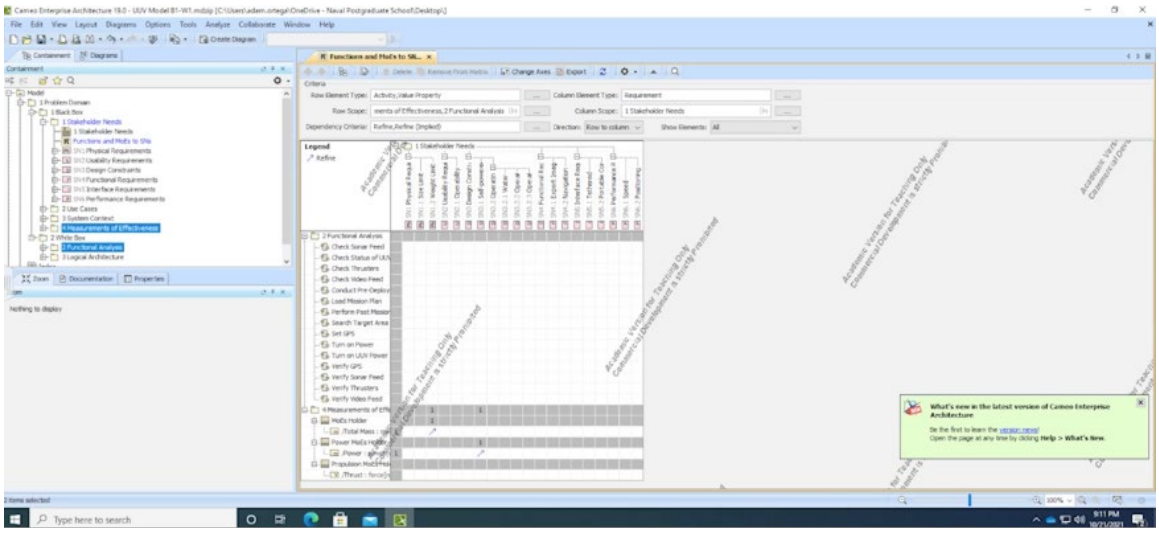
Then we will click on the *Refresh* hyperlink to update the matrix



And from here we can start to capture the refine relationship by double clicking any of the cells at the intersection of the rows and columns that are appropriate. For example, we can click on the intersection between *Total Mass* and *Weight Limit*.



And we can click on the intersection of *Power* and *Self-powered*.



We can continue to establish the relationships between the two that applicable depending on the Stakeholder Needs are, the activities, and MOEs of the system. This concludes the tutorial on the final stakeholder needs. Thank you.

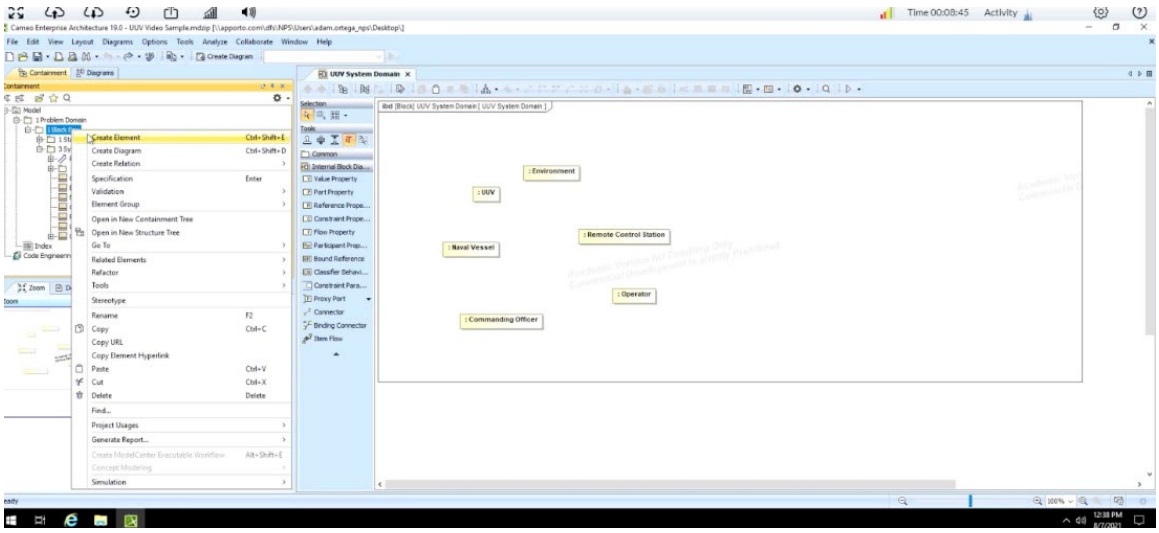
B2: Use Cases Part I Tutorial

Background:

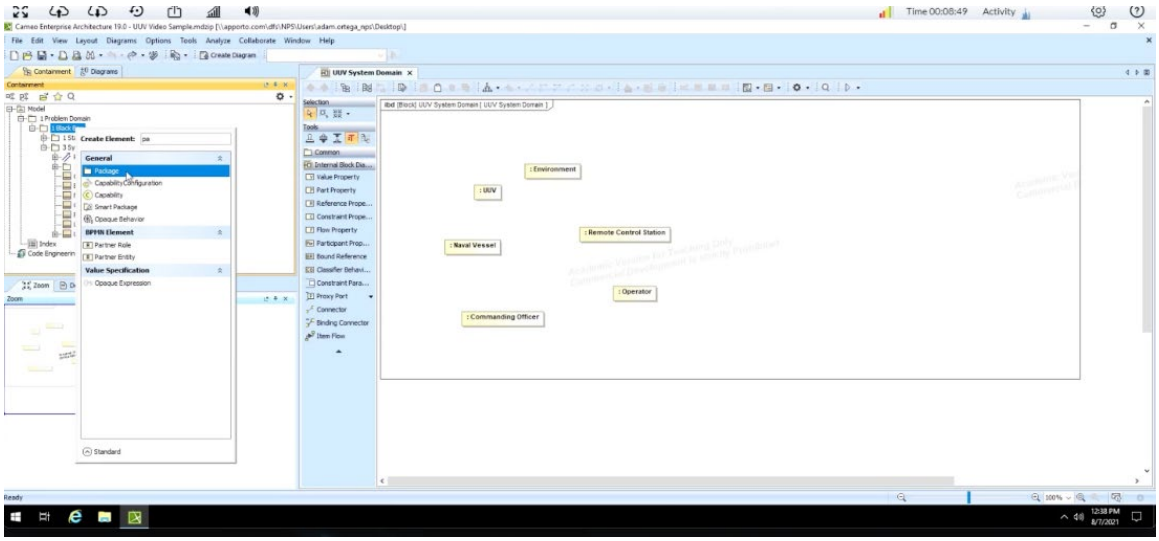
In this section of the tutorial, use cases and use case scenarios for the system are developed to redefine functional stakeholder needs. The use cases and use case scenarios are more accurate in presenting what to expect from the system and what is wanted to be achieved. It is the responsibility of the Requirements Team, to specify and define the different use cases of the system.

Step-By-Step Guide:

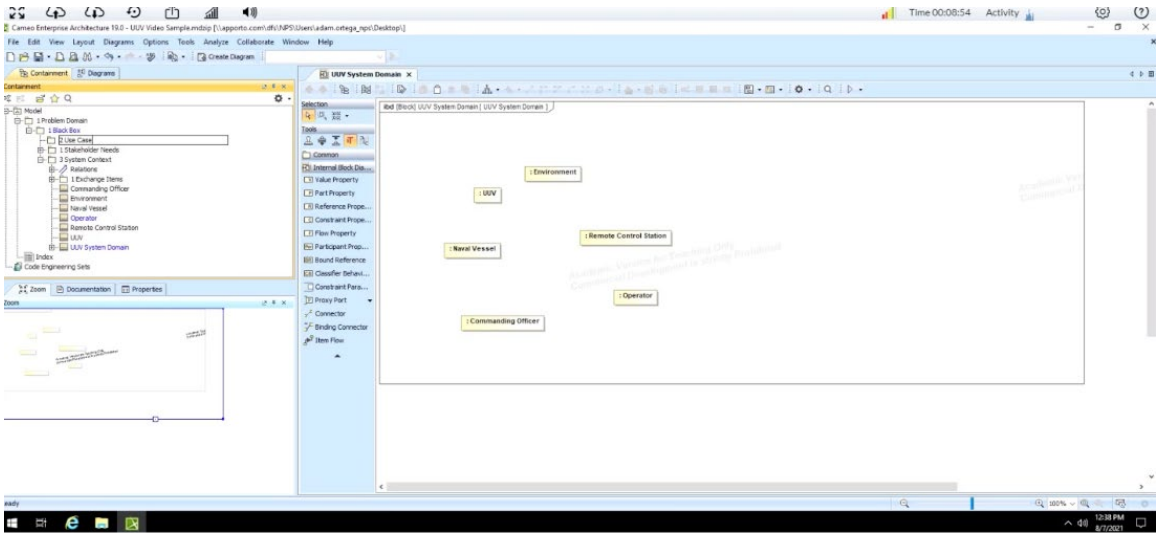
This tutorial is Part I of creating a use case and their scenario for the system. The first step will be creating a separate package for the use cases to keep the containment tree organized. Therefore, we will create another package within the black box. Let's right click the *black box* package and select *Create Element* from the drop-down menu.



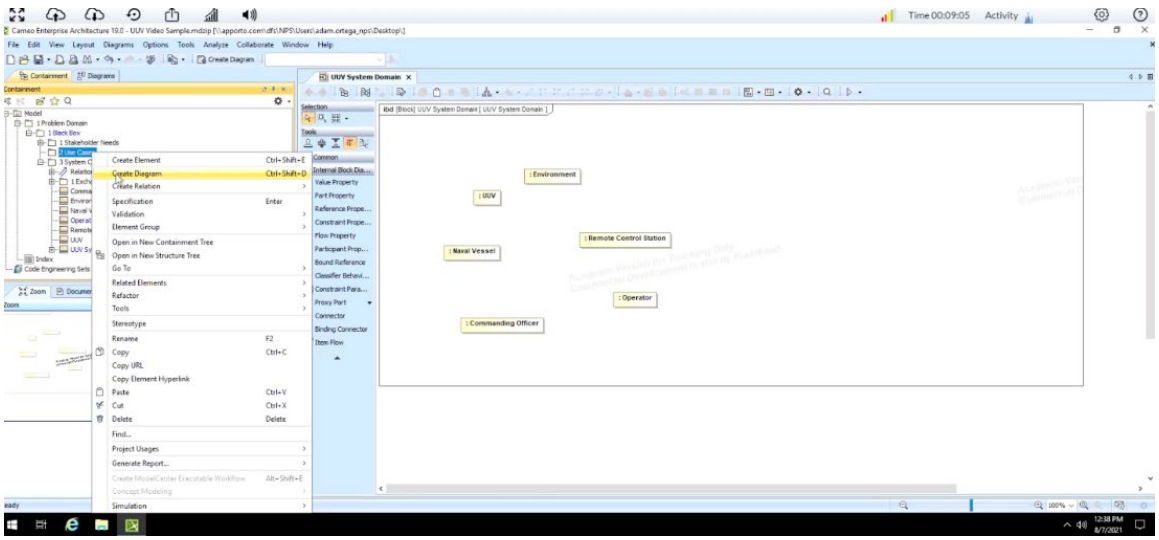
Then the *Create Element* menu window will appear and from the search field type in *Pa* to select *Package*.



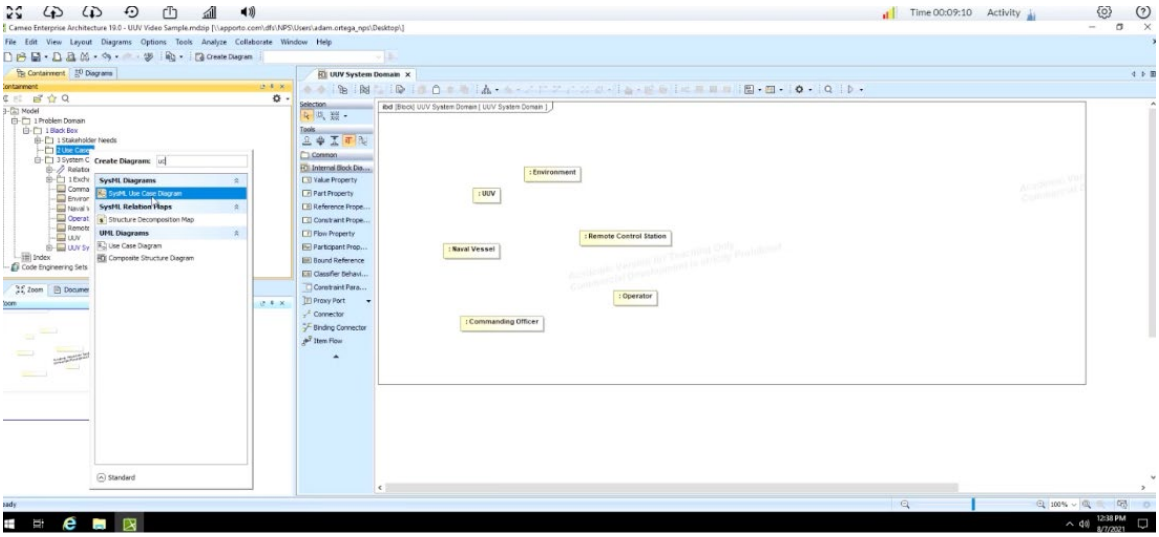
And we will name this *2 Use Cases* and then we will hit the enter key on the keyboard.



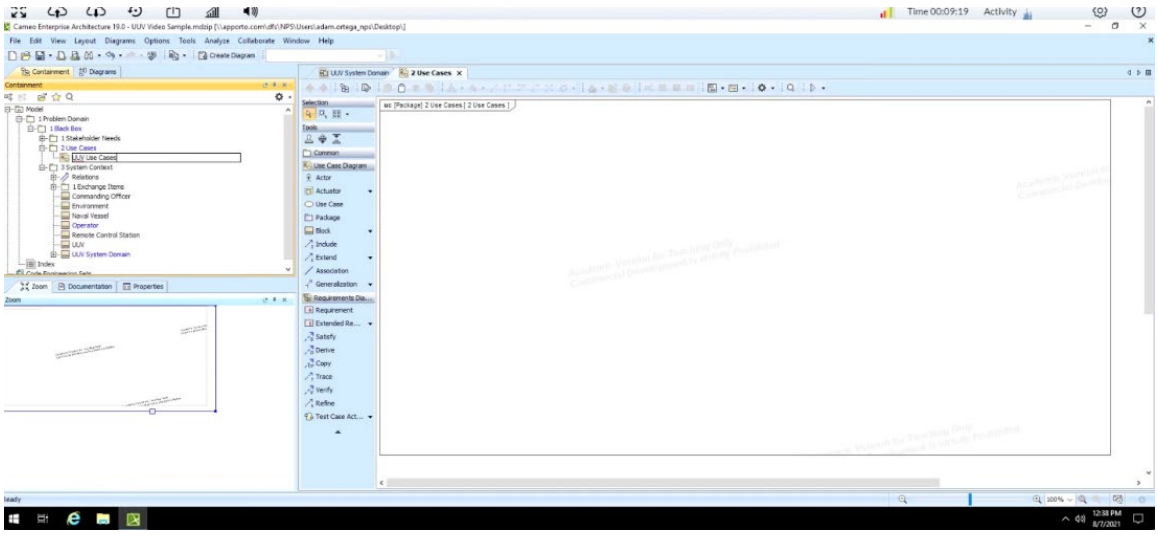
Now we will go ahead and start creating the Use Case Diagram. We will right click the *Use Cases* package and from the drop-down menu we will select *Create Diagram*.



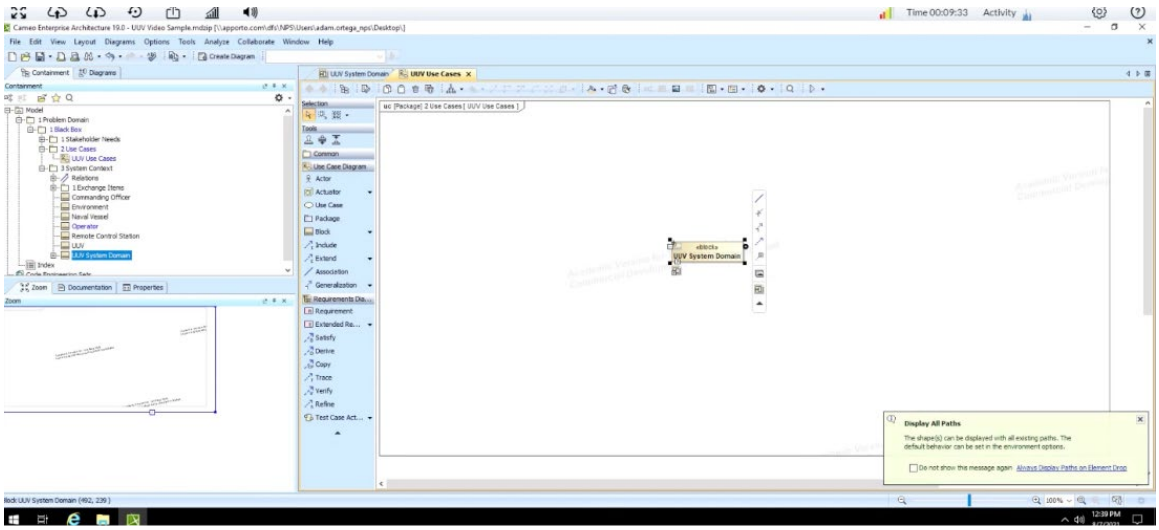
Then the *Create Diagram* menu window will appear and from the search field type in *Uc* in order to find and select *SysML Use Case Diagram*.



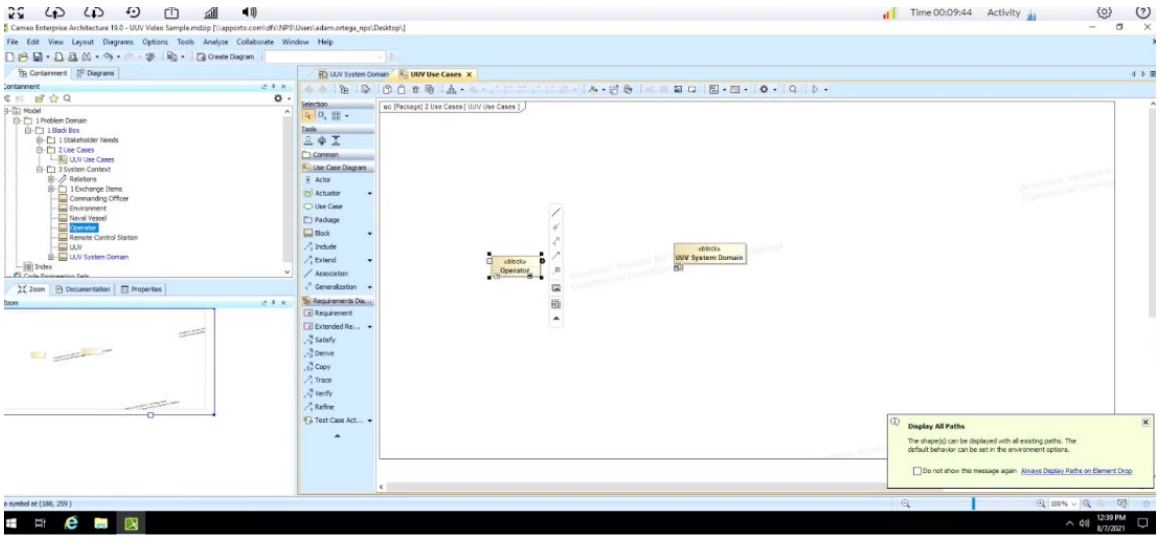
And we will name this *UUV Uses Cases* by typing in the text field.



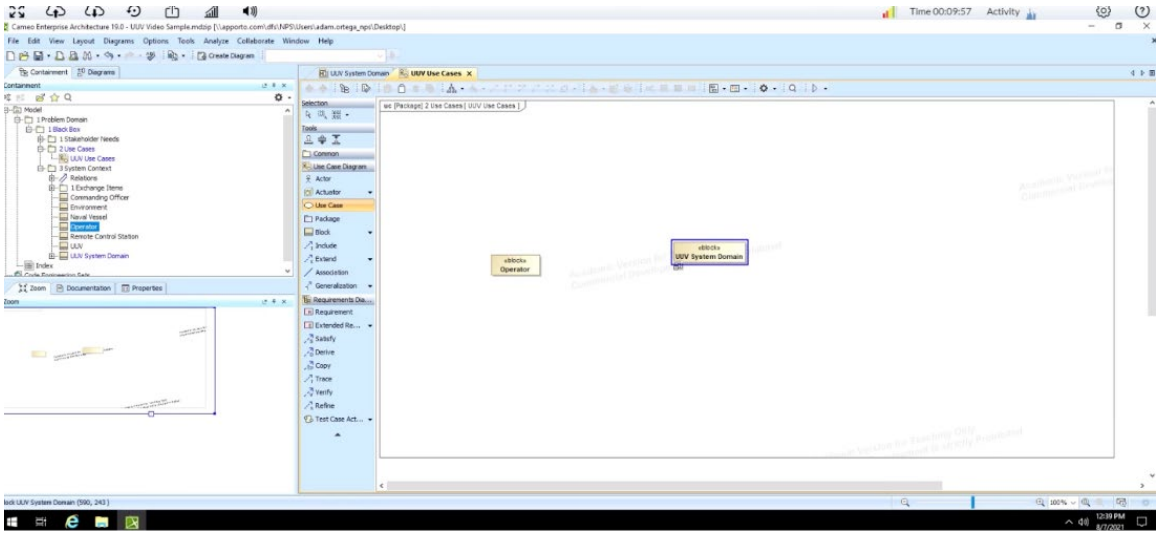
Now we can select the *UUV System Domain block* from the system domain package in the model browser. We are going to drag this onto the *Use Cases* pane.



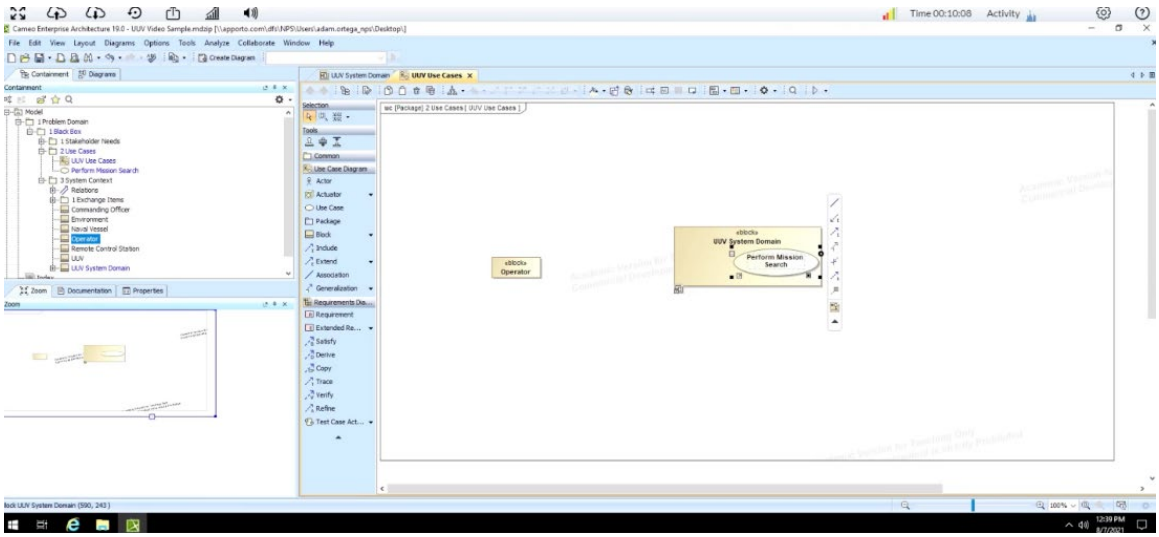
We will also select the *Operator* block and it onto the *Use Case* pane. The reason is that the *Operator* is the primary user of the system.



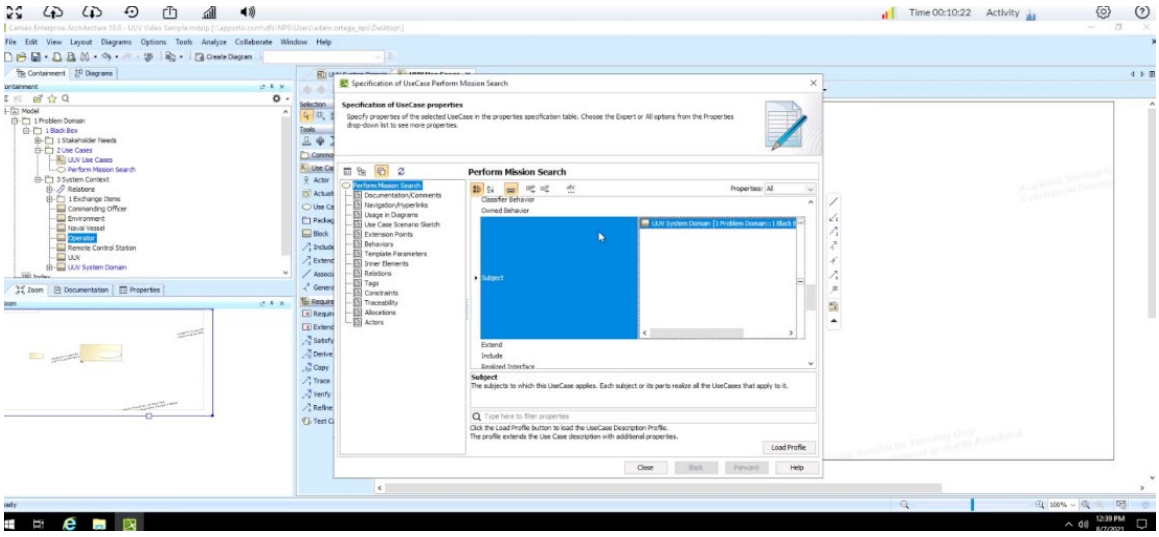
Now we will start capturing the use cases by clicking the *Use Case* button on the diagram pallet and drag into the *Use Case* pane. And we will go over the *UUV System Domain block*. A blue border will appear around the *UUV System Domain block* when hovering over it.



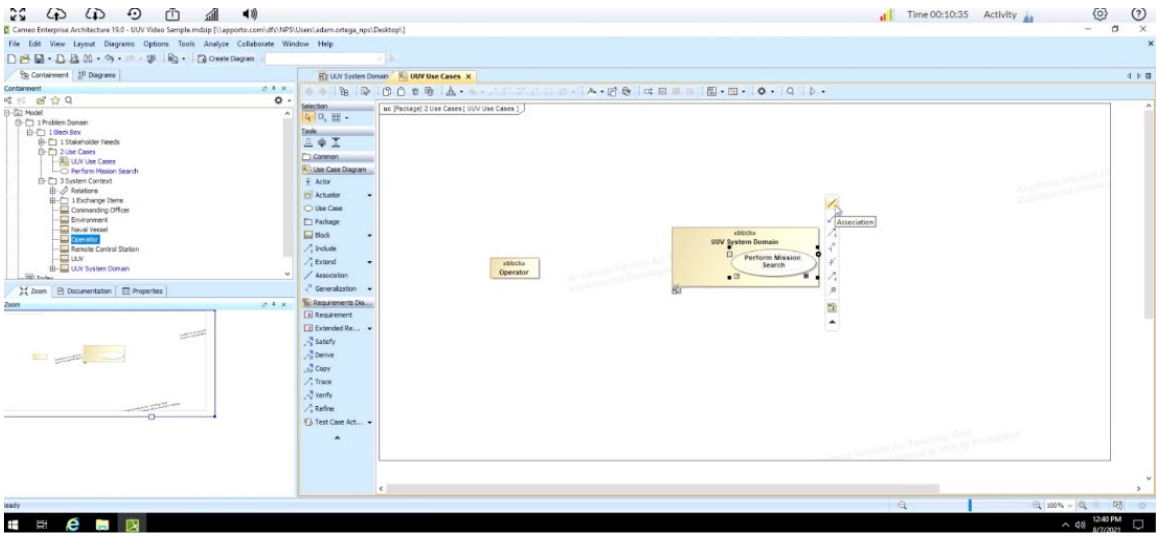
We then click the *UUV System Domain block* and it will create a use case within the block. Then we will start to type *Perform Mission Search* and then hit the enter key on the keyboard.



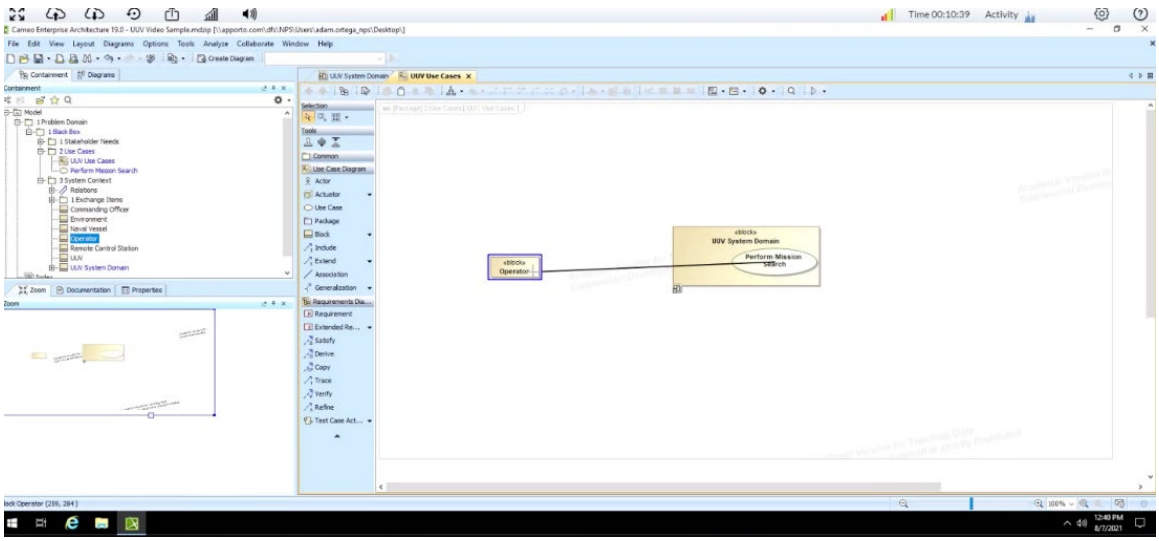
Then we will go ahead and double click on the use case, which then a pop-up window will appear to modify the specifications of the Use Case *Perform Mission Search*. We will scroll down to verify that UUV System Domain block is set as the subject.



Now we can close the specifications of the Use Case *Perform Mission Search* pop-up window. Now that we have done that, we can set an association. We will click on the *association* button on the smart movability tool bar.

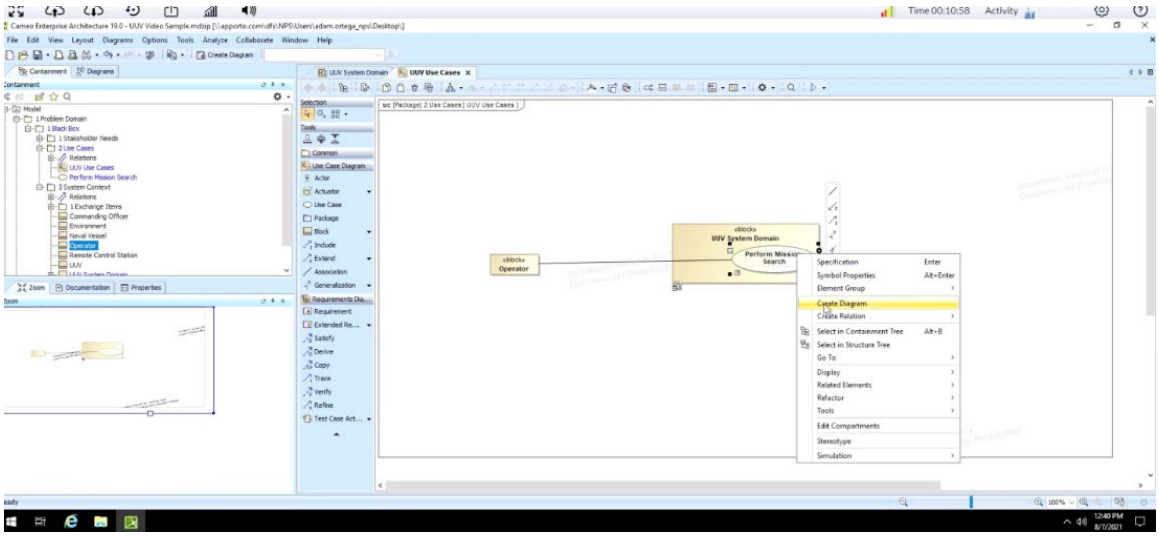


Once we have clicked on the association button, we will bring it over to the *Operator* block. This will cause the *Operator* block to be highlighted in blue. And then we will click on the *Operator* block to finish making an association.

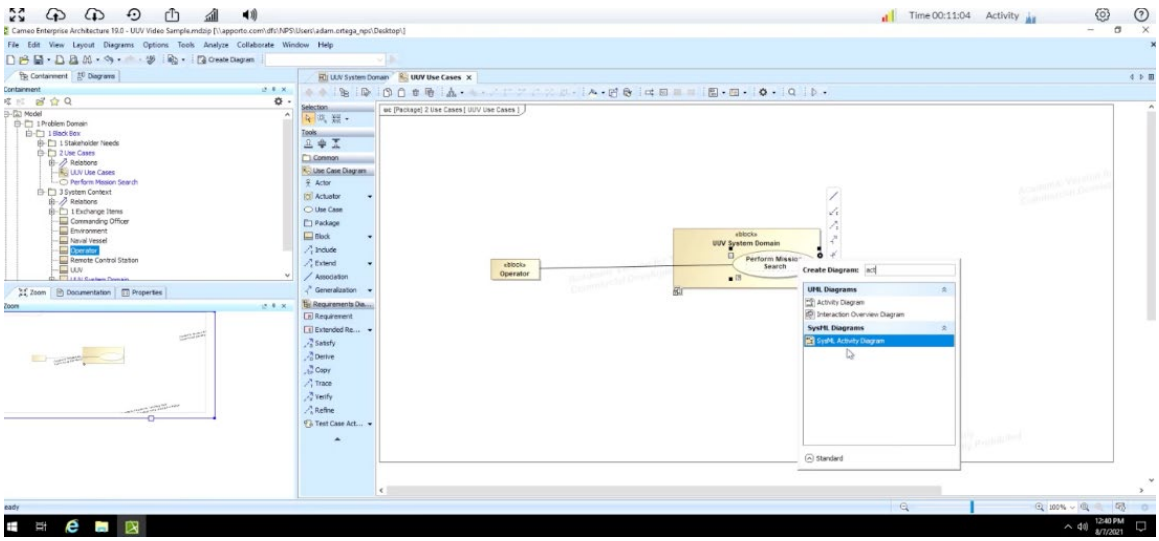


Now we can start creating the diagram for specifying the use case scenario. We will be creating a SysML Activity Diagram. So, we can right click on the use case shape of the *Perform Mission Search*. A drop-down menu will appear and we will select the *Create Diagram* option.

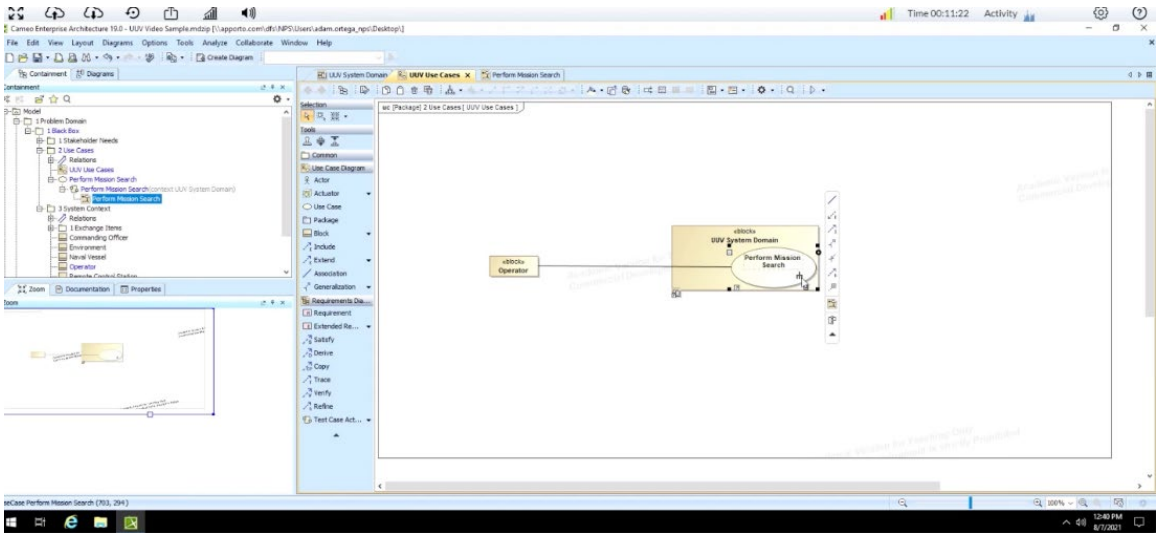
Then the *Create Diagram* menu window will appear and from the search field type



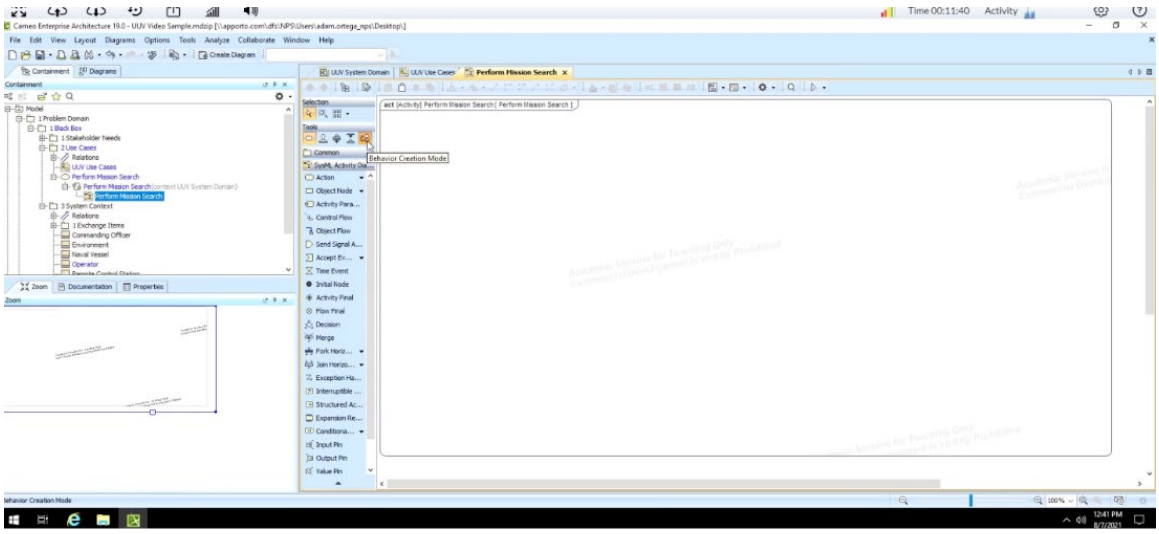
in *act* and select *SysML Activity Diagram* from the menu window, which it will automatically open.



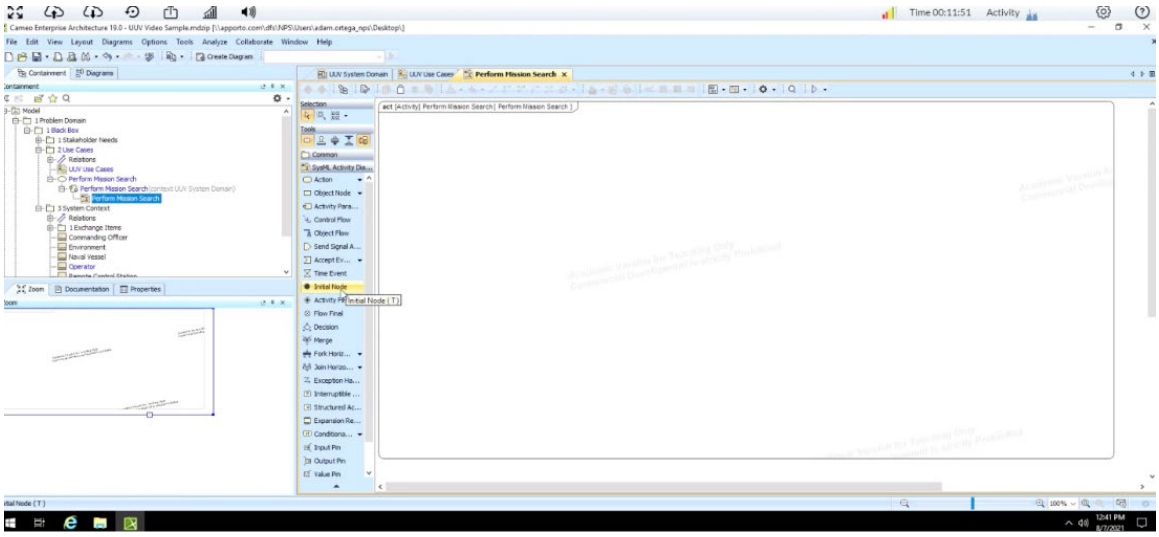
We can go back to the *UUV Use Case Diagram* and verify that there is an internal diagram with the *Perform Mission Search* as denoted by the shape of the rake.



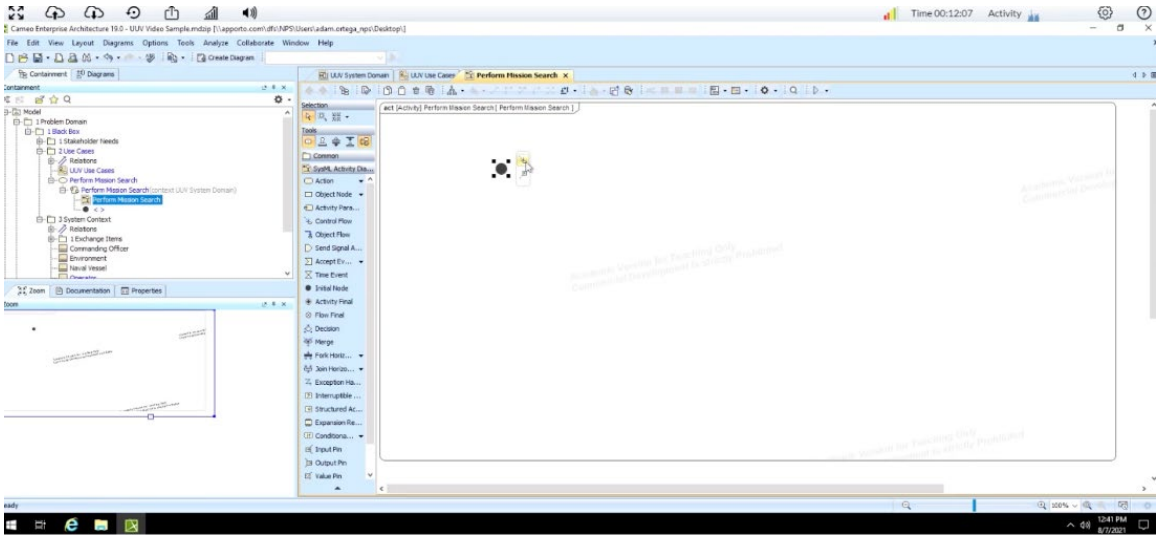
So now we can start specifying the use cases scenario. Make sure we go back to the *Perform Mission Search Activity Diagram*. We will make sure that the automatic *behavior creation mode* is on. This is to make sure that the actions created are typed by activities.



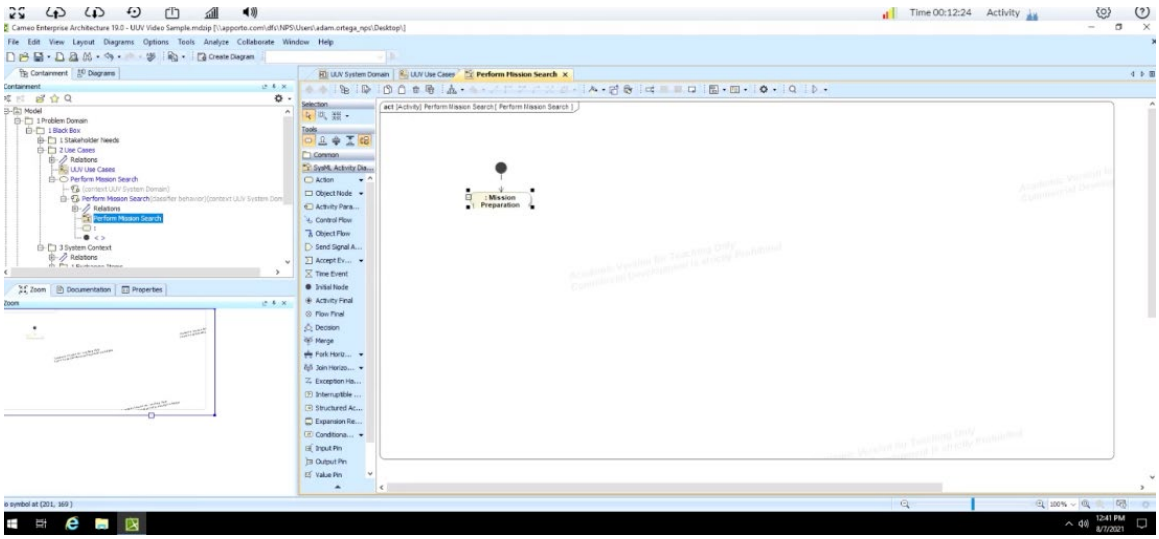
Now we can start with the *Initial Node* button on the diagram pallet and click an empty space on the *Perform Mission Search Activity Diagram*.



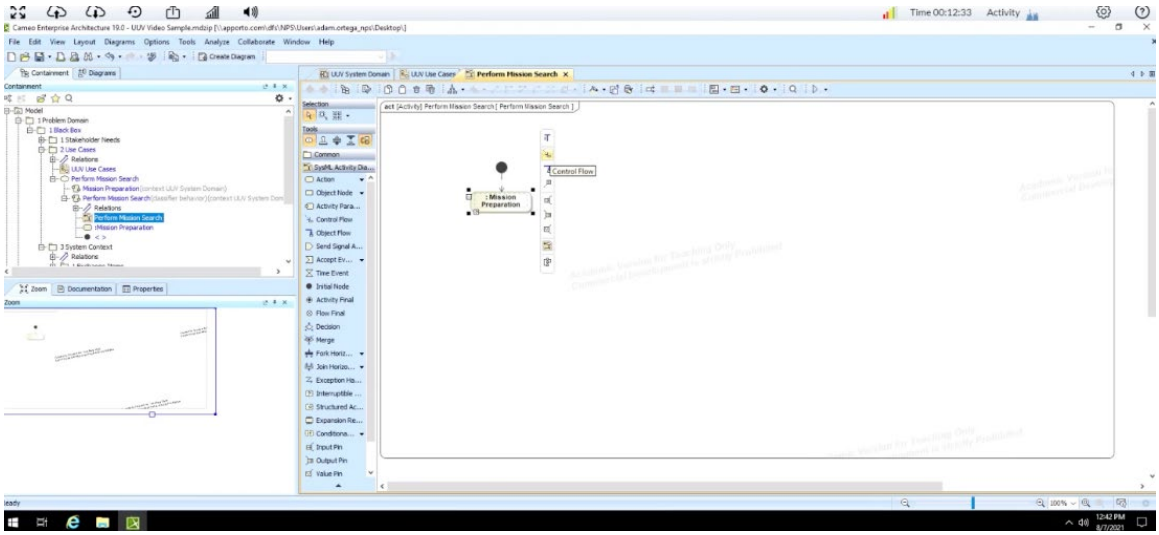
Now that we have the *Initial Node* placed on the *Perform Mission Search Activity Diagram*, from here we can select the *Control Flow* button on the smart manipulator tool bar.



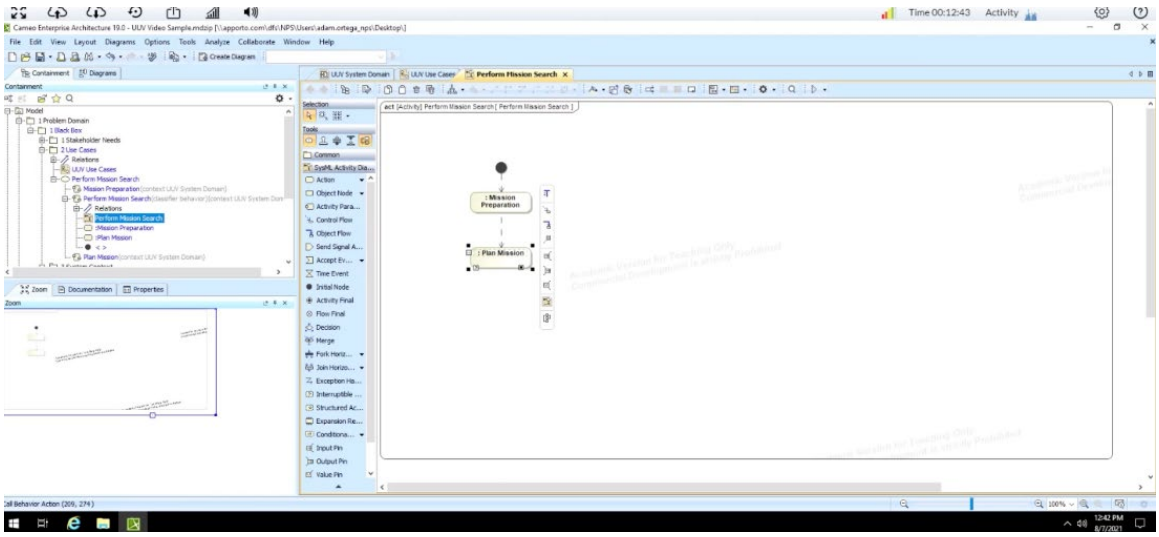
Then we click on an empty space below to create a new activity. For this UUV case study, we will start with the first activity by typing *Mission Preparation* and hit the enter key on the keyboard.



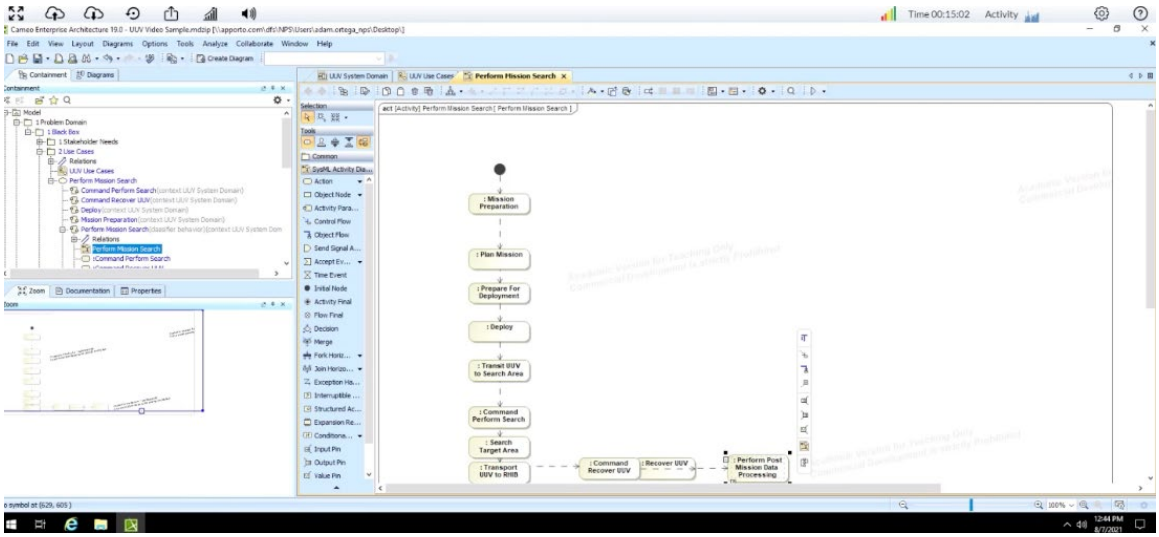
We will repeat these steps to build out this case scenario. Therefore, select the *Control Flow* button from the smart manipulator tool bar.



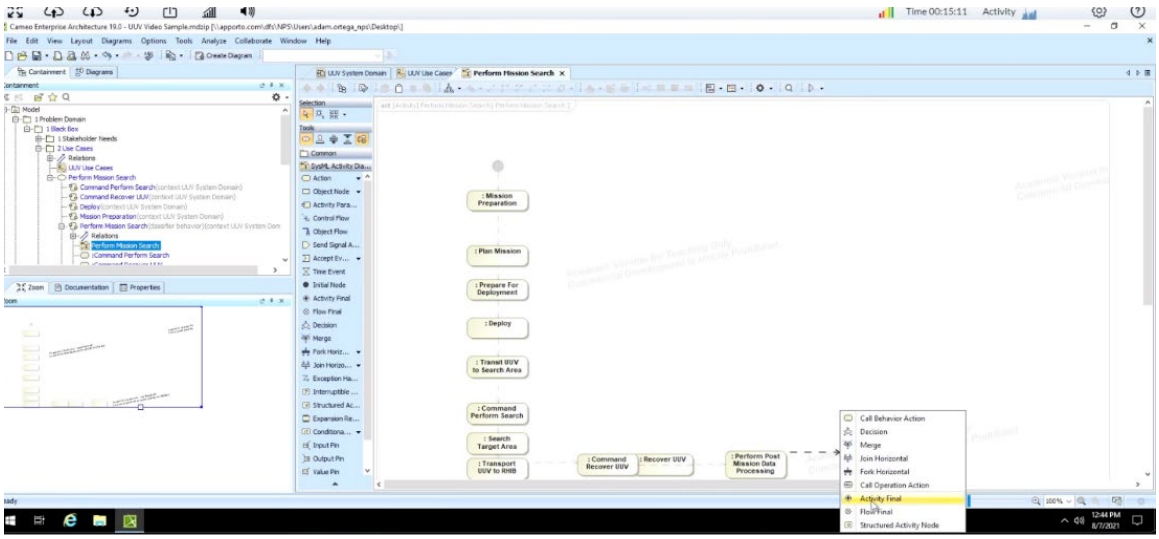
And then again click an empty space to create a new activity, which will be *Plan Mission* for the second activity. After typing *Plan Mission*, we will then hit enter key on the keyboard.



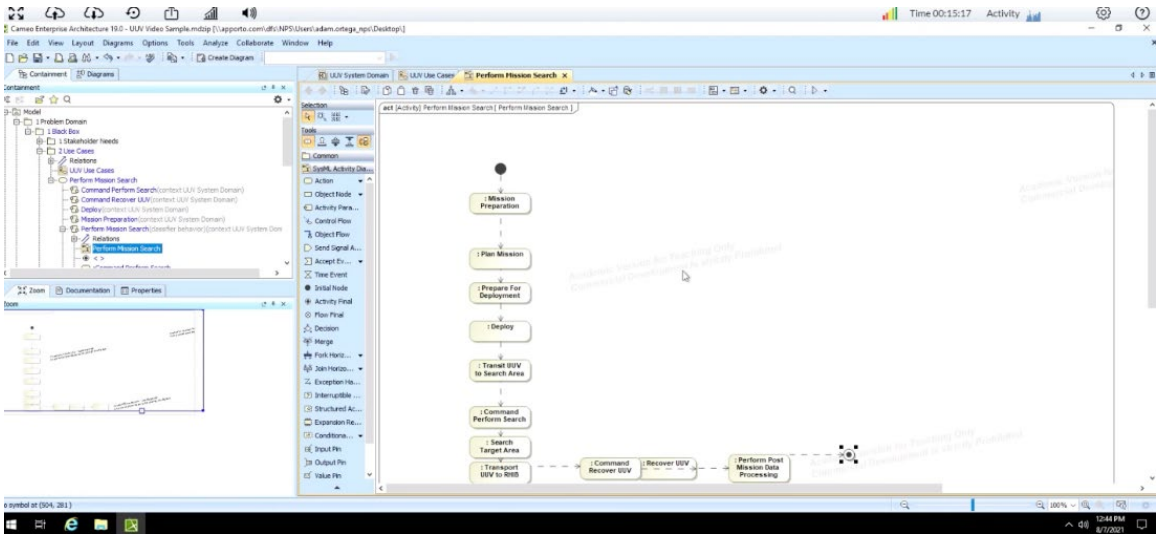
We will repeat the steps again for the following new activities. The Next activities will be *Prepare For Deployment*, *Deploy*, *Transit UUV to Search Area*, *Command Preform Search*, *Search Target Area*, *Transport UUV to RHIB*, Command *Recover UUV*, *Recover UUV*, and *Perform Post Mission Data Processing*.



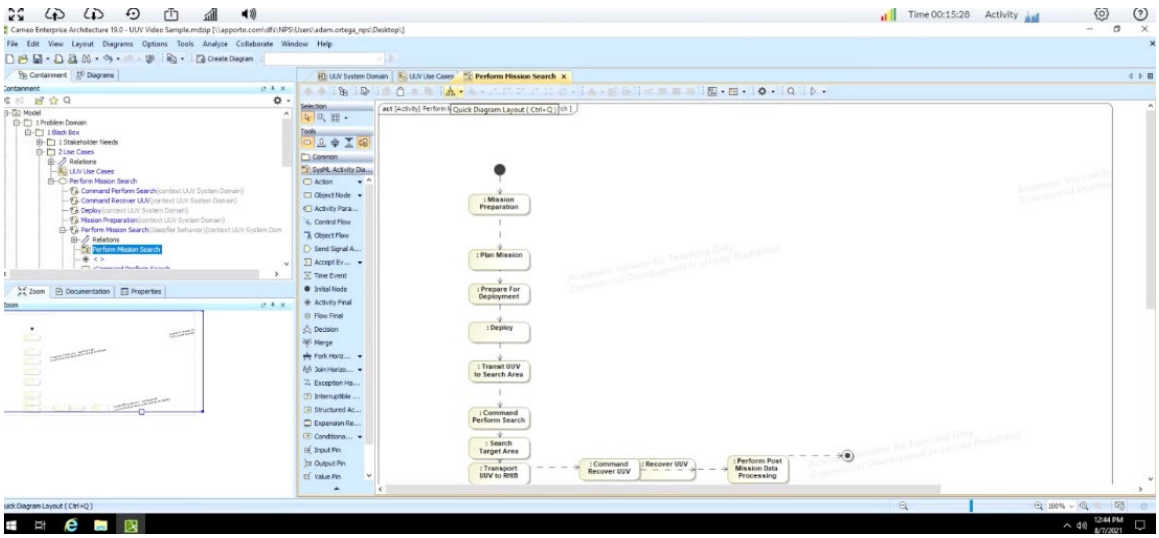
For the last activity which is the *Perform Post Mission Data Processing*, we will click on the *Control Flow* button and drag the dashed arrow to the right and then right click to select *Activity Final* node from the drop-down menu.



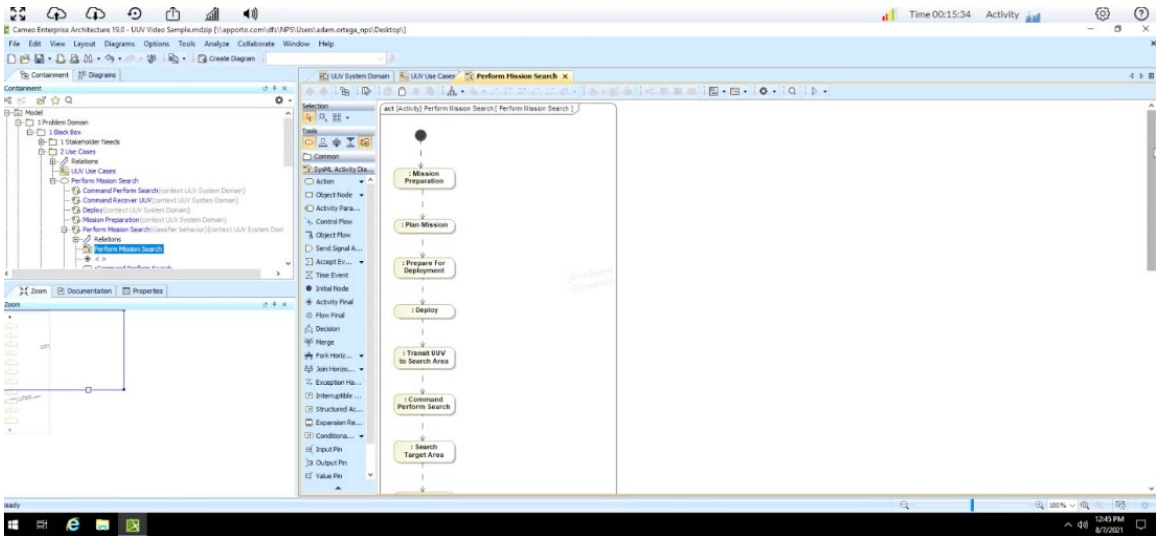
This will then create an End Node on the *Perform Mission Search Activity Diagram*.



Then we will click on the empty space of the *Perform Mission Search Activity Diagram* to unselect the end node. Then we will then click on the *Quick Diagram Layout* button to format the activity diagram to a much better and evenly spaced-out view.



Following the steps provided in this tutorial will get an outcome that is similar to the activity diagram shown below.



This concludes Part I of creating a Use Case. Thank you.

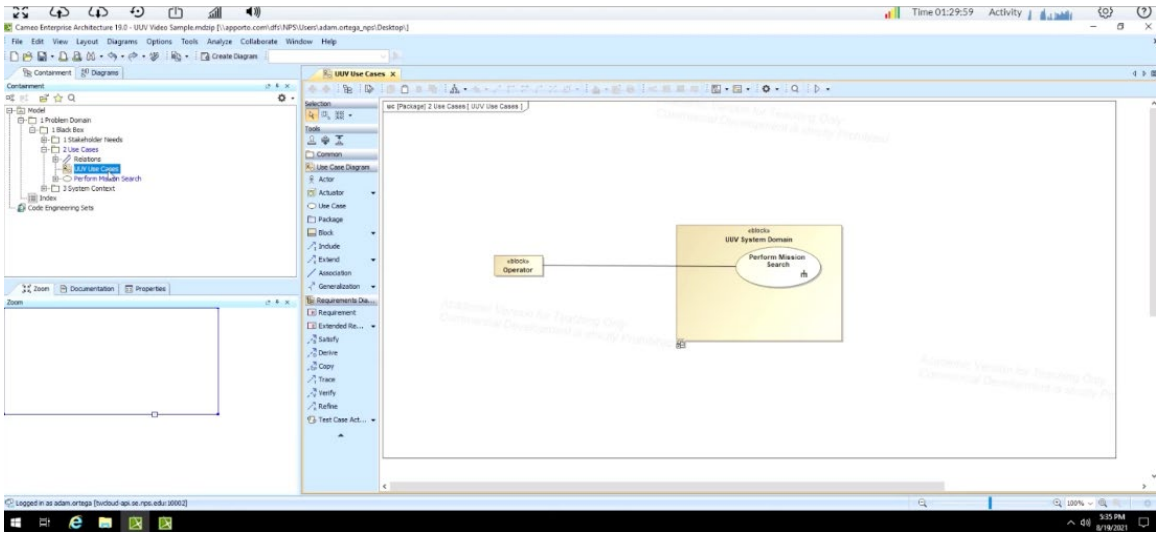
B2: Use Cases Part II Tutorial

Background:

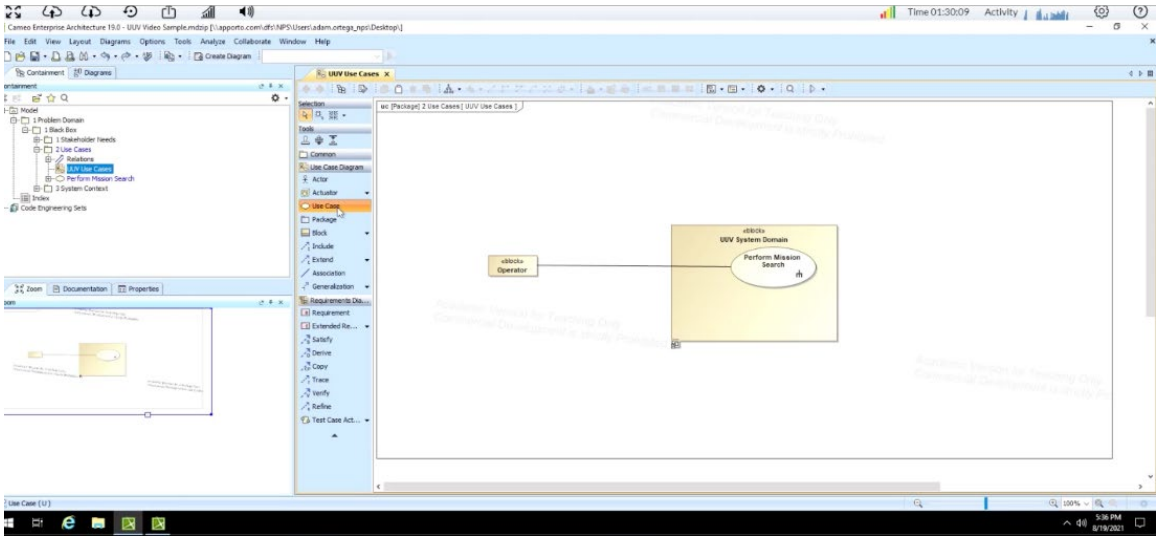
This part of the model utilizes a SysML use case diagram in which entities, represented as blocks, perform one or more use cases of the system context. Using blocks, instead of actors, is recommended because blocks are treated as a part of the system context, can be marked as external, and can have a defined behavior. The use case scenarios, on the other hand, are captured using SysML activity or sequence diagrams. The users of the system will also be identified.

Step-By-Step Guide:

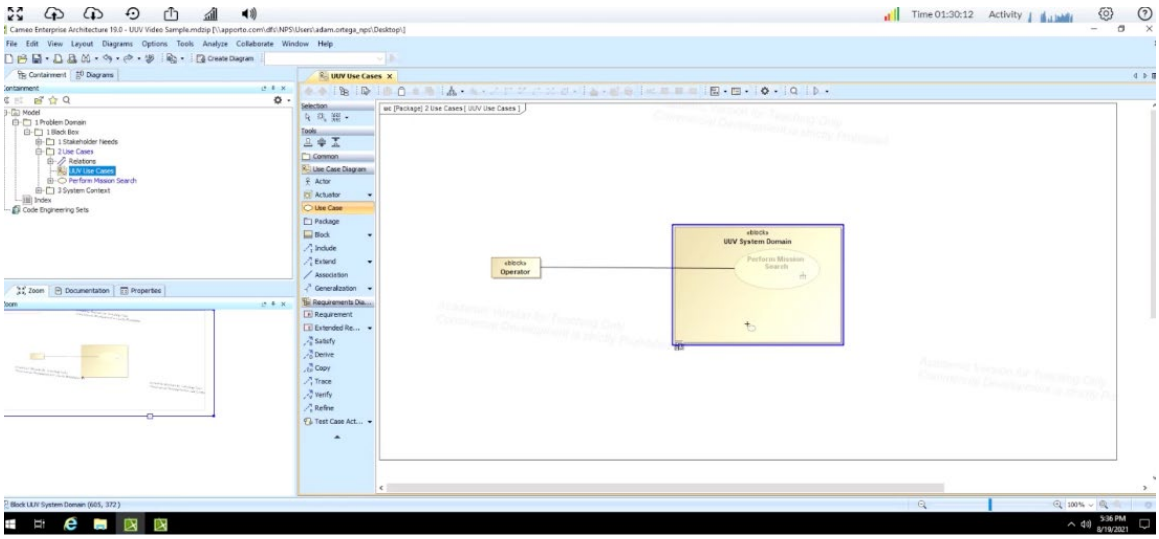
This is part 2 of the tutorial of Use Cases. We will look at adding alternative flow of actions and as well as allocating behavior to the structure, which specifies which participants of the related system context is responsible for performing which actions. So let's go ahead and open our Use Case diagram. First, we will start by expanding our *Problem Domain* package and then locate and expand the *Use Cases* package. Once the *Use Cases* package has been expanded locate the *UUV Use Cases Diagram* and double click it. This will bring up the diagram that was created on the Use Case Part I tutorial video.



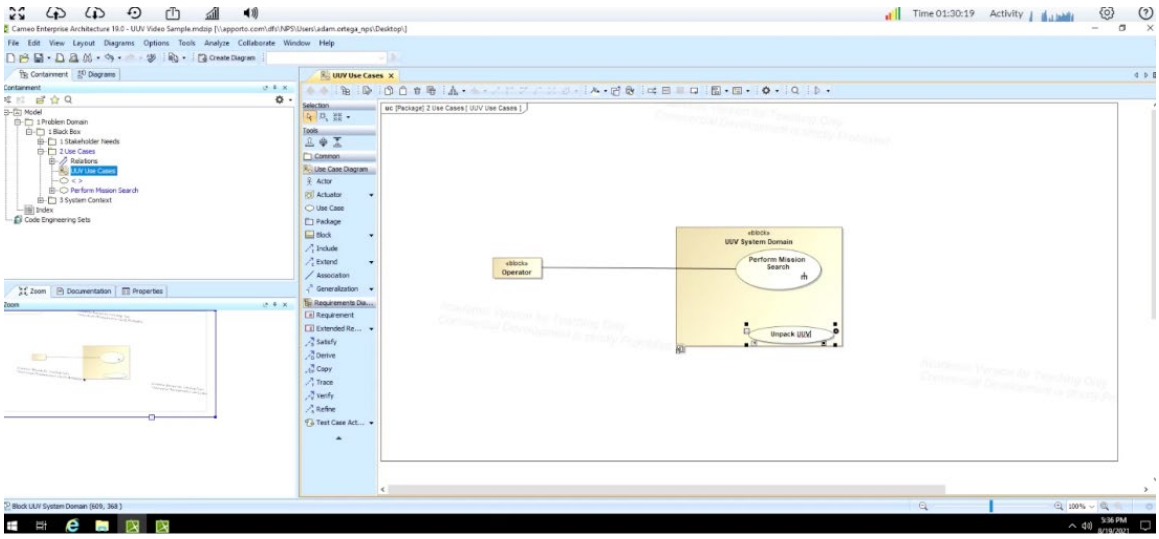
For this example, we will go ahead and create another use case. So we will click on the *Use Case* button in the diagram pallet.



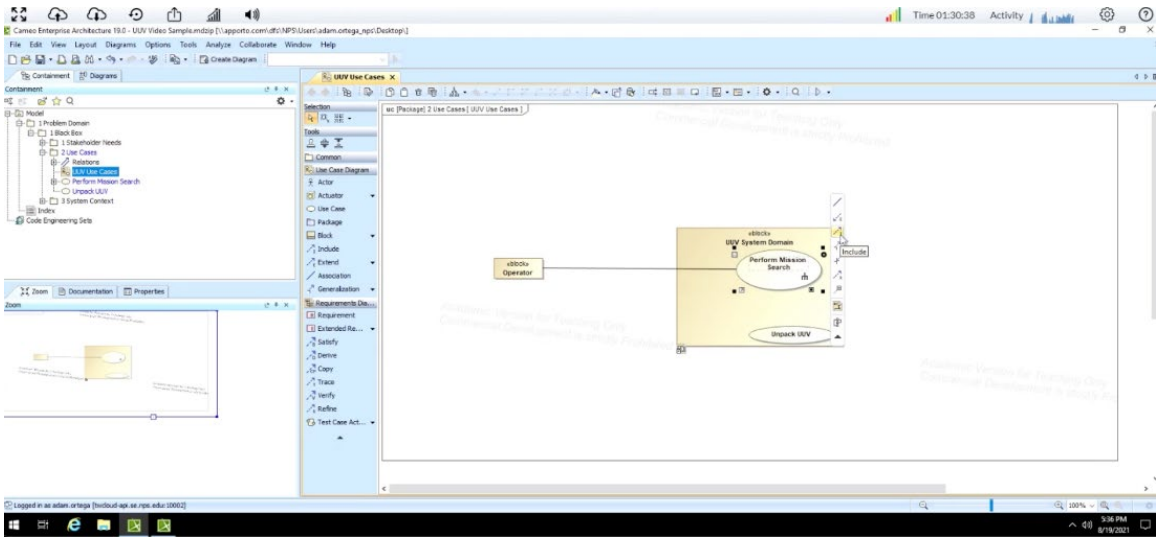
Then we will drag the new use case to the *UUV System Domain* block. This will cause a blue highlight around the *UUV System Domain* block when the mouse pointer is hovering over the *UUV System Domain* block.



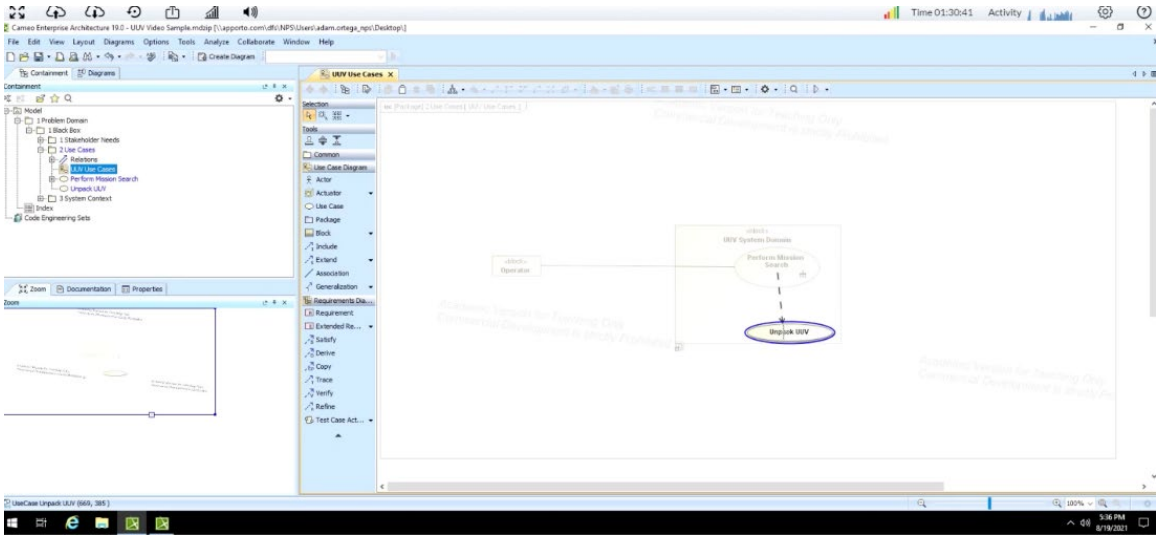
Click on the *UUV System Domain* block to create the new use case inside the block. For this use case, we will name it by typing *Unpack UUV* and hit the enter key on the keyboard.



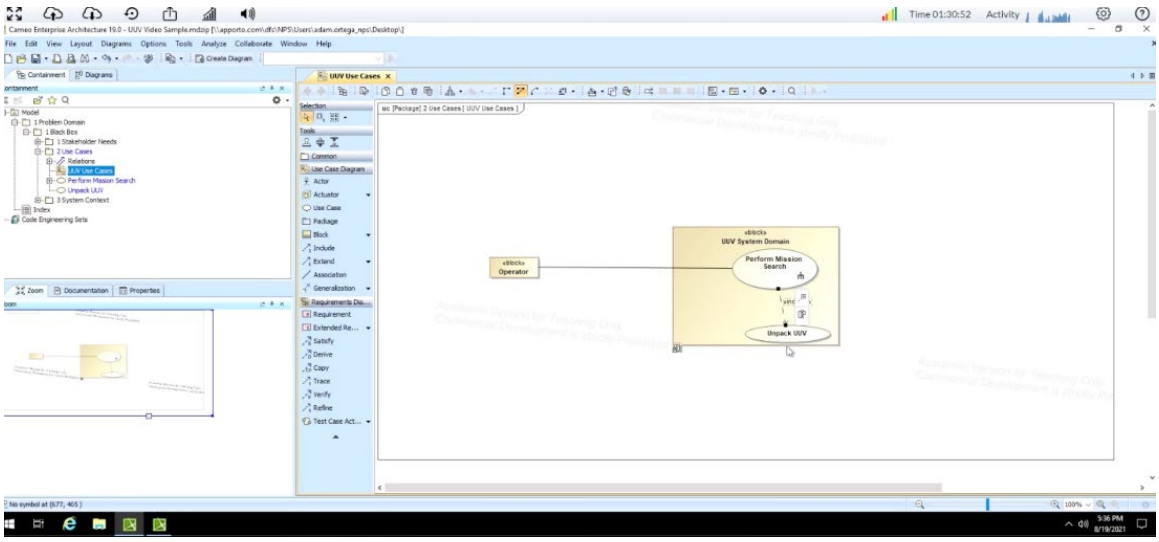
Before we create an activity diagram, we know that the scenario of unpacking the UUV is part of the *Perform Mission Search* scenario. Therefore, we can click on the *Perform Mission Search* scenario and then click on the *include* button on the smart manipulator tool bar.



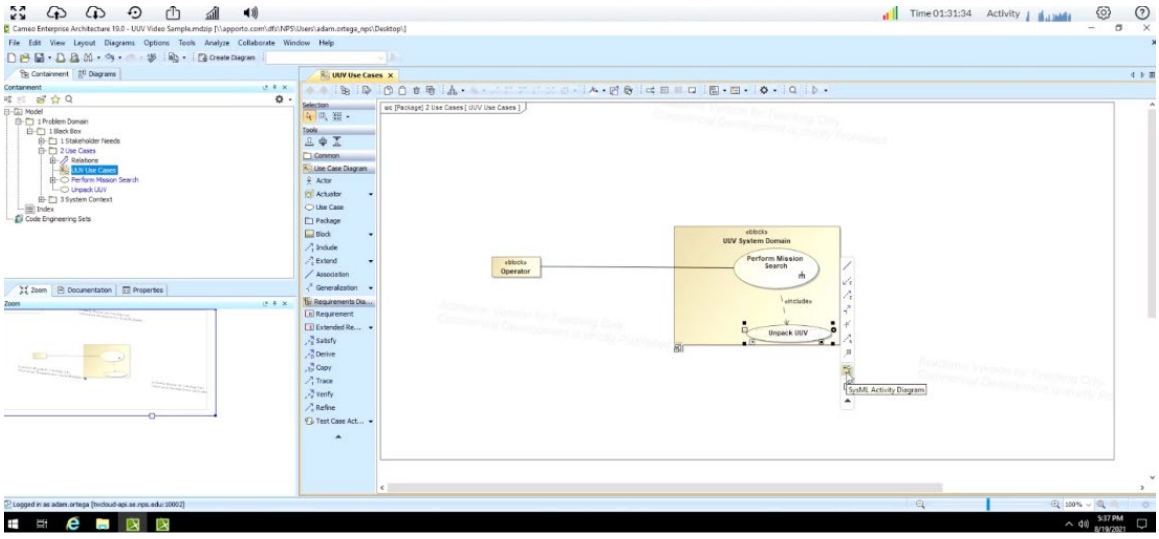
Next, bring it over to the *Unpack UUV* use case. This will cause a blue highlight circle around the use case scenario *Unpack UUV*.



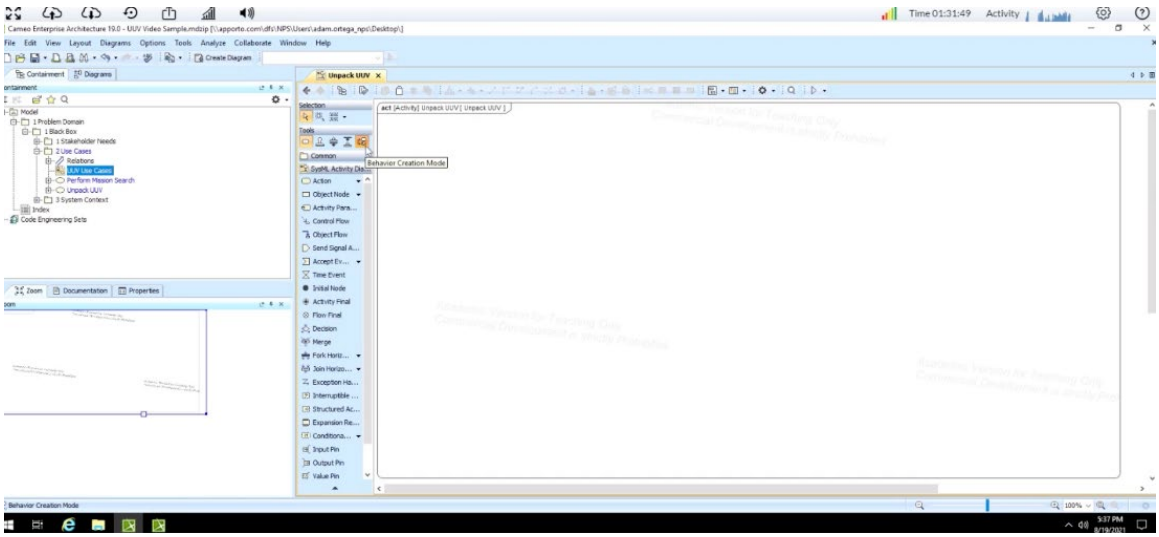
Now, we know that the *Unpack UUV* scenario is a part of the *Perform Mission Search* scenario.



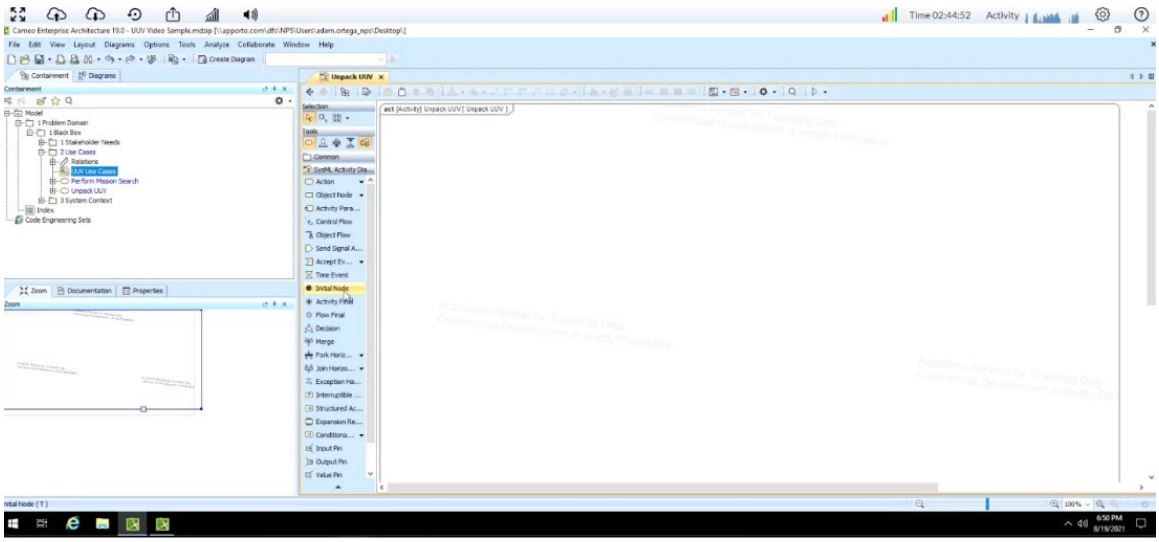
We will now create the activity diagram of the for the *Unpack UUV* scenario. Therefore, we will click on the *Unpack UUV* scenario and from the smart manipulator tool bar we will select the *SysML Activity Diagram* button.



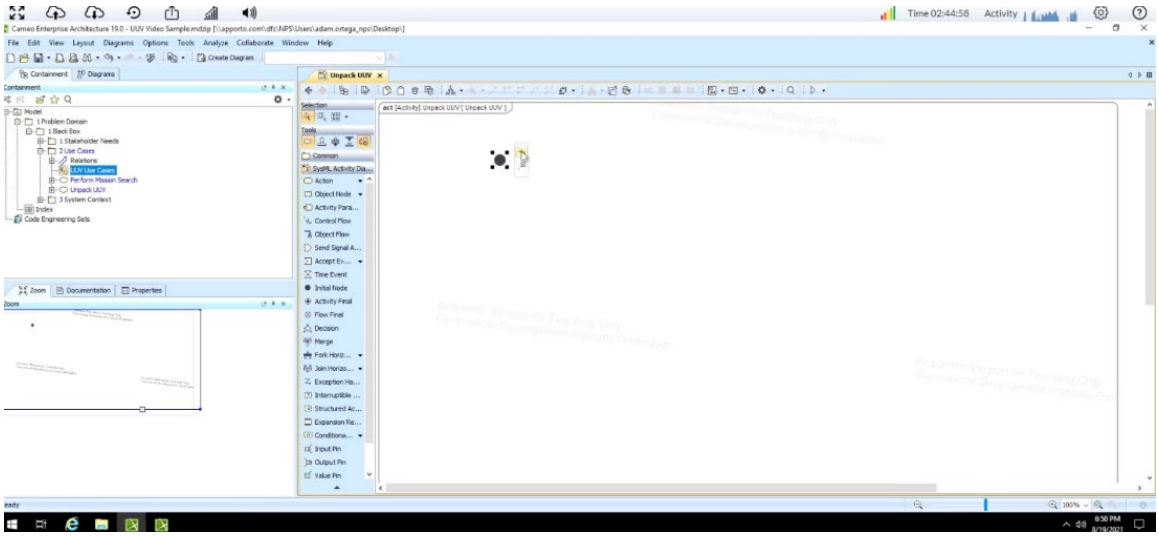
This will then open a new activity diagram window. So we start in the same steps as the pervious Use Case Part I tutorial video. We will make sure that the *Automatic Behavior Creation Mode* button is selected so that the new items will be created as activities.



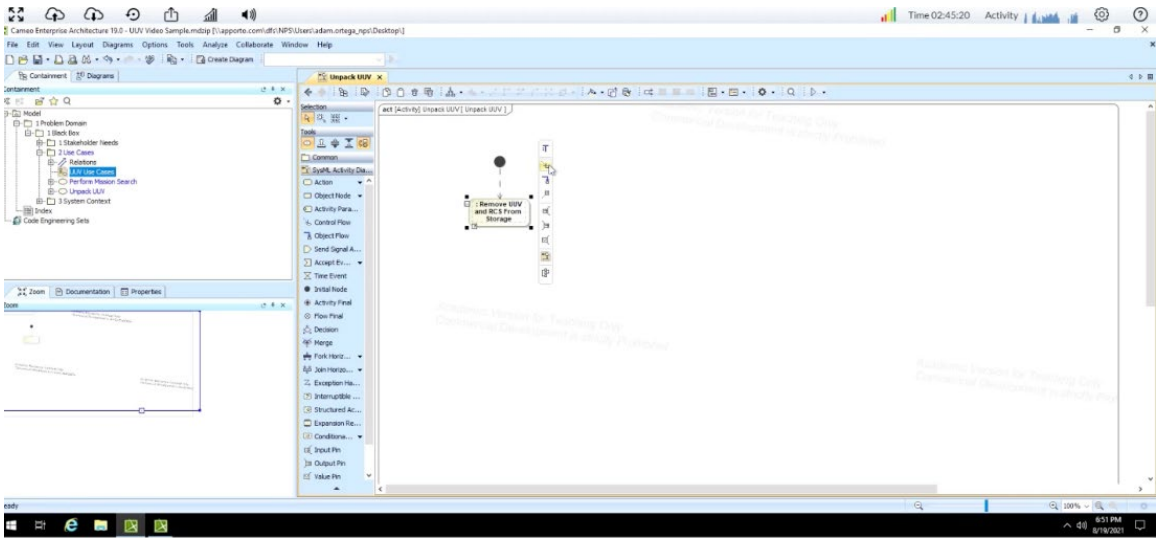
So to start we will click on the *Initial Node* button in the diagram pallet and click on the diagram pane to setup of the initial starting node. In the smart manipulator tool bar we will select the *Control Flow* button.



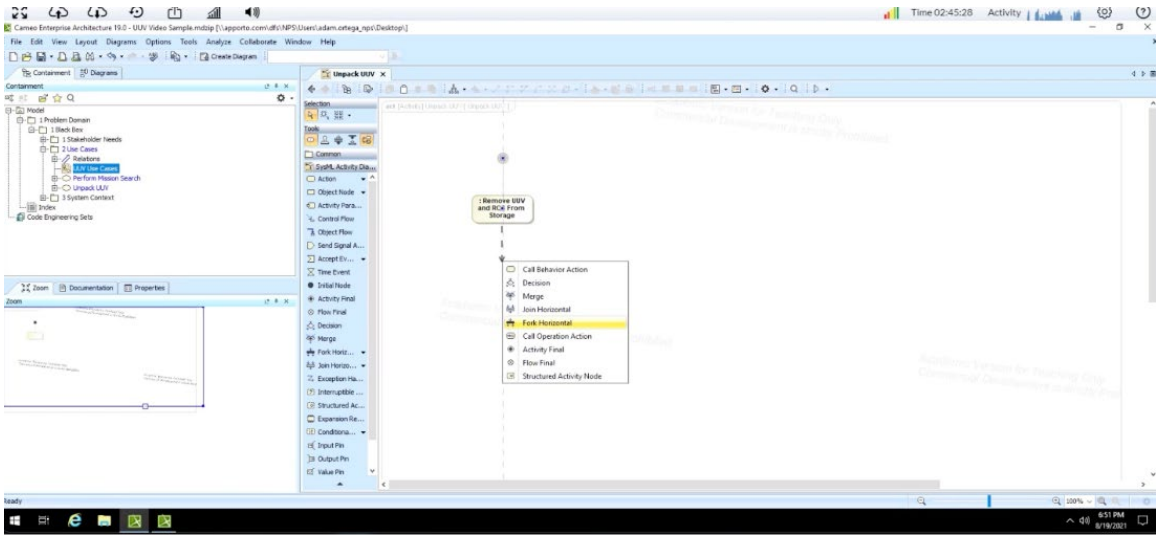
Then we will drag down and click on the empty space of the diagram plane.



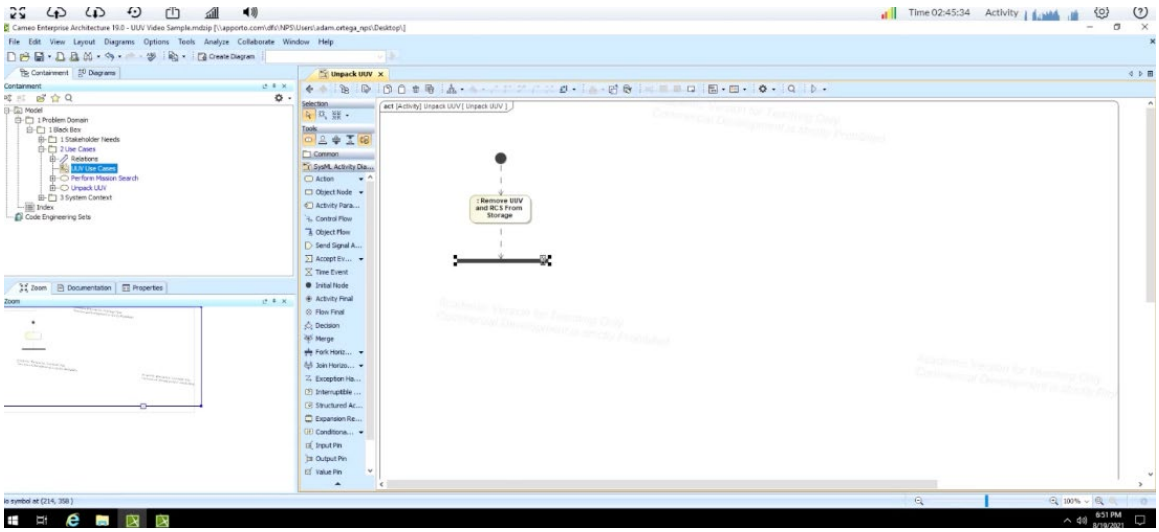
The first action we will type *Remove UUV and RCS from Storage*. Then we will click on the *Control Flow* button in the smart manipulator tool bar to create another action.



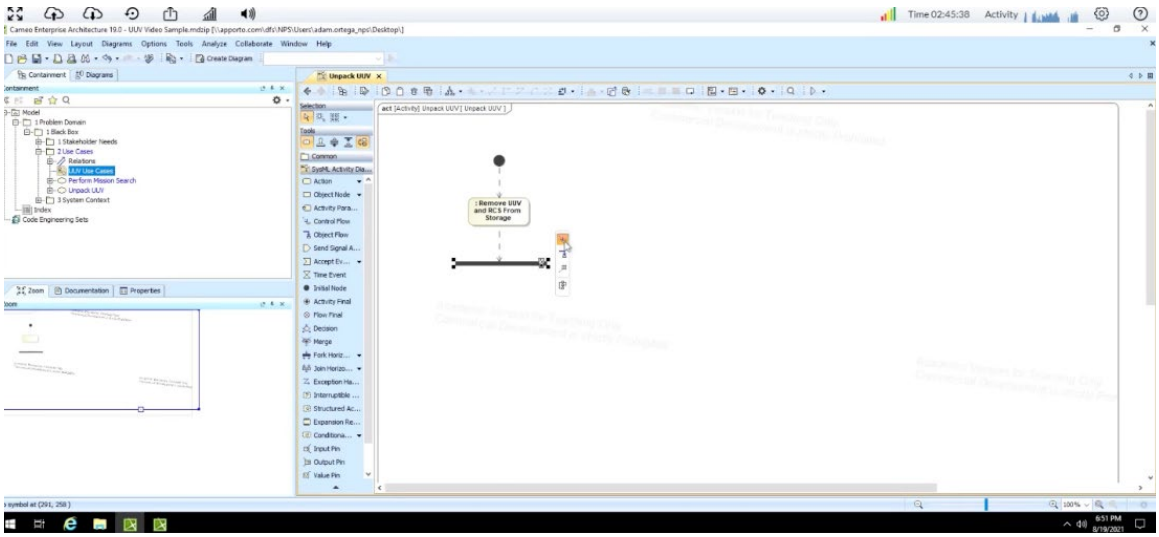
This time we will right click on the empty space of the diagram plane. A drop-down menu will appear and we will select *Fork Horizontal*.



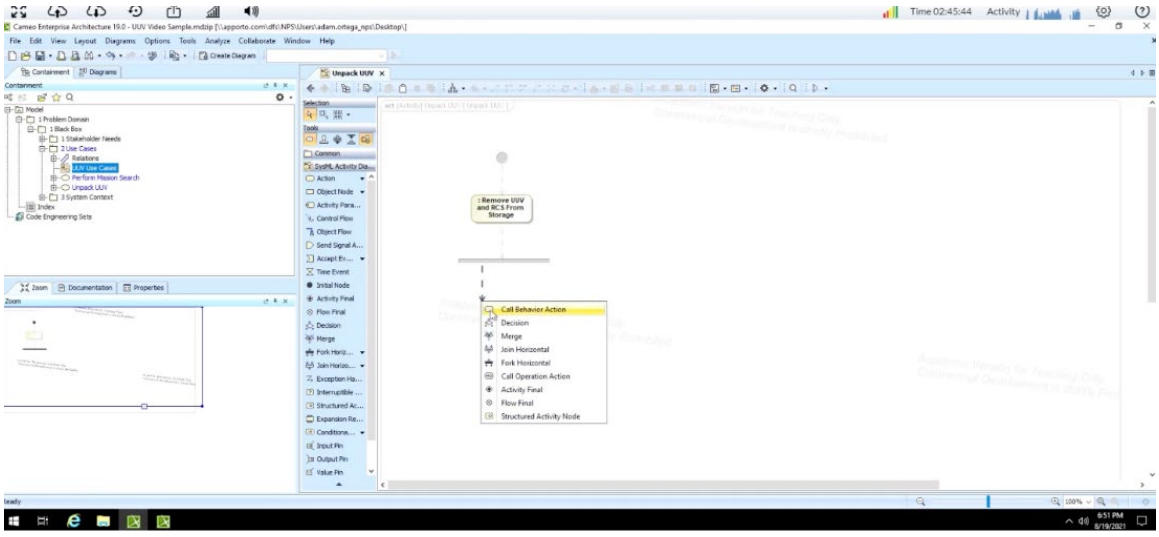
A black bold bar will appear on the diagram plane. This is to show the simultaneous flow of actions.



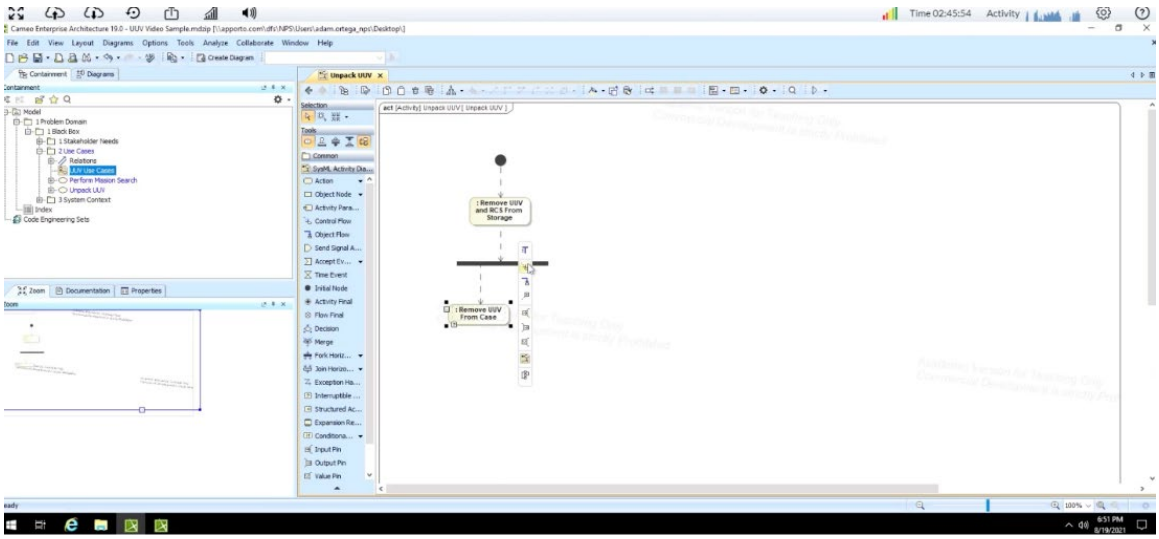
So we will click the *Control Flow* button in the smart manipulator tool bar of the horizontal fork bar to start the first path.



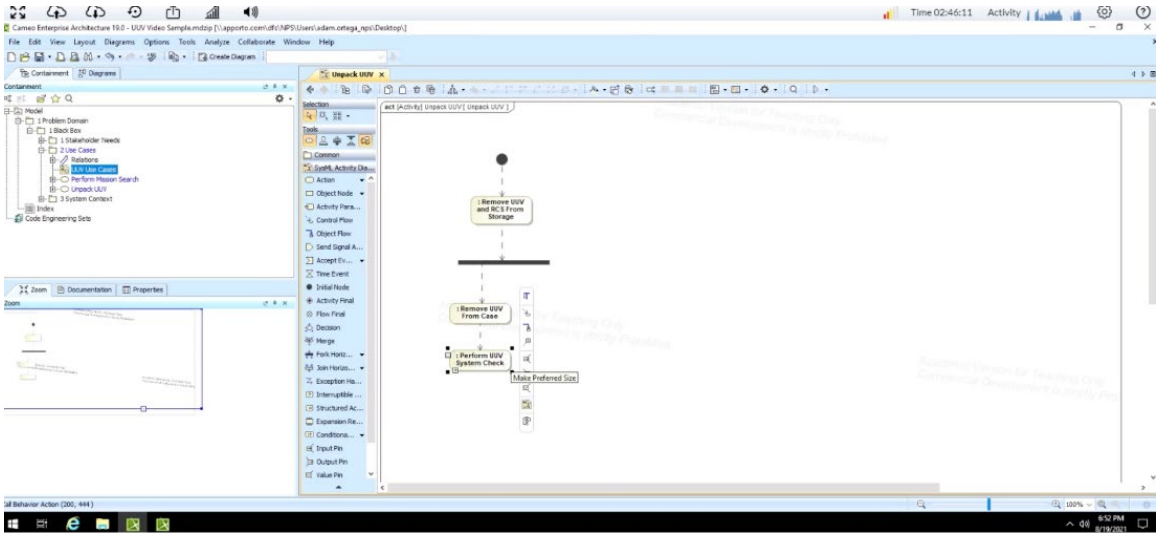
Again, we will need to right click on the diagram and select *Call Behavior Action* from the drop-down menu.



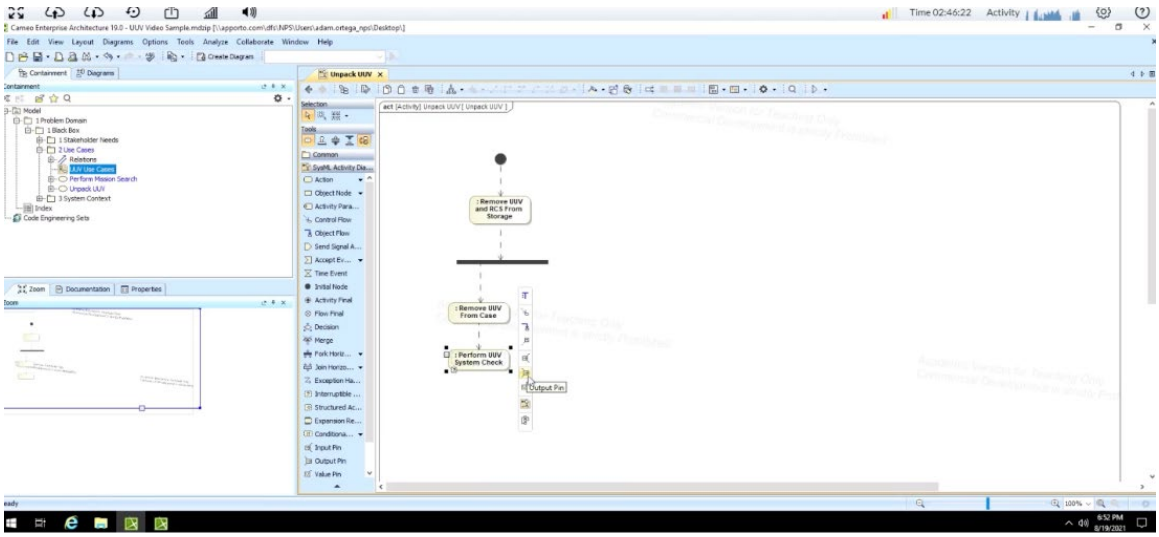
In selecting the *Call Behavior Action* from the drop-down menu, it will create an action in the diagram plane. We will start by typing the name of the new action *Remove UUV From Case* and then hitting the enter key on the keyboard. And again we will click the *Control Flow* button in the smart manipulator tool bar to create another action.



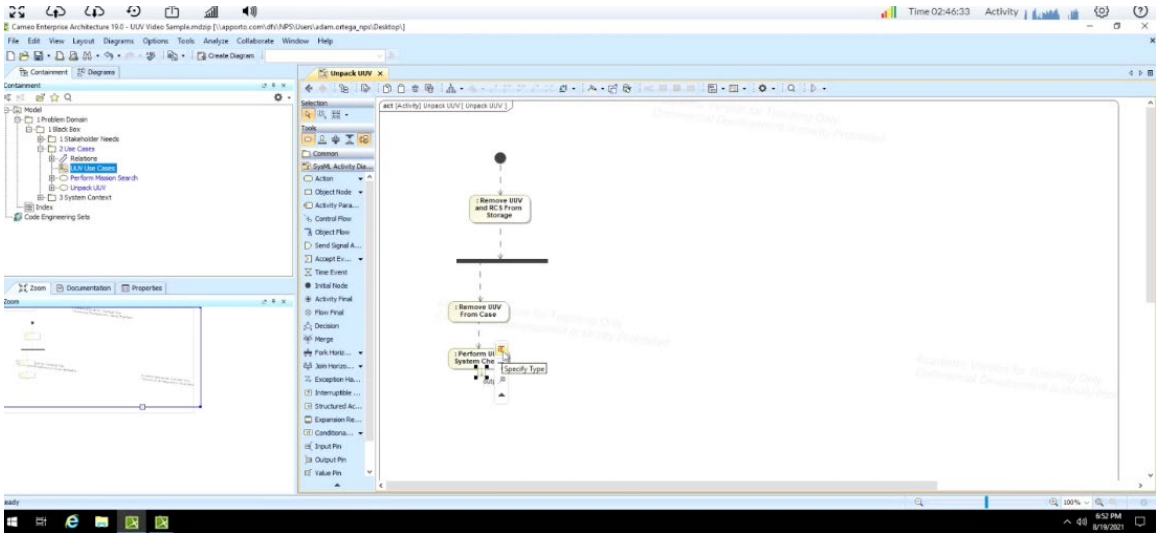
Then right click and select *Call Behavior Action* from the drop-down menu to create the new action. The new action will be named *Perform UUV System Check*.



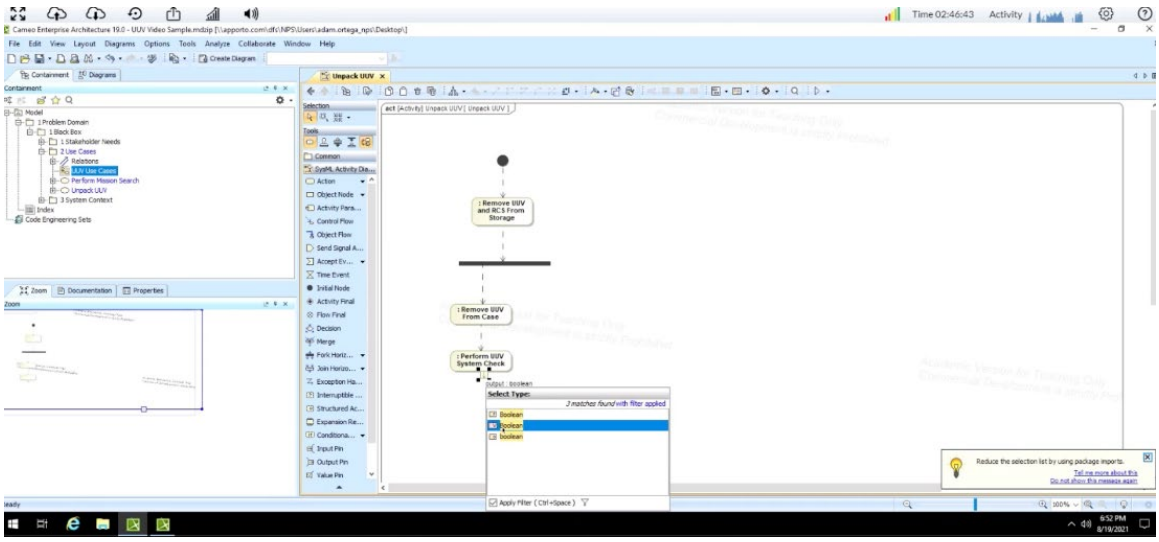
Now for this action we will click on the *Output Pin* on the smart manipulator tool bar to illustrate that actual information is being passed. We can go ahead and bring the *Output Pin* to an empty space on the diagram plane.



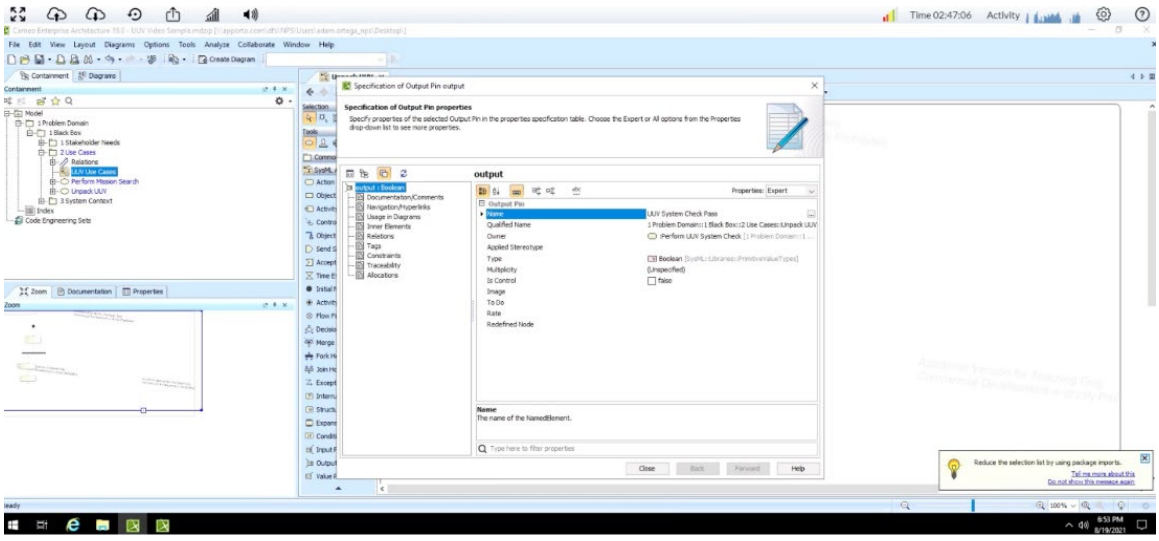
And now we can specify a type by clicking on the *Specific Type* button on the smart manipulator tool bar denoted as a *:T* symbol.



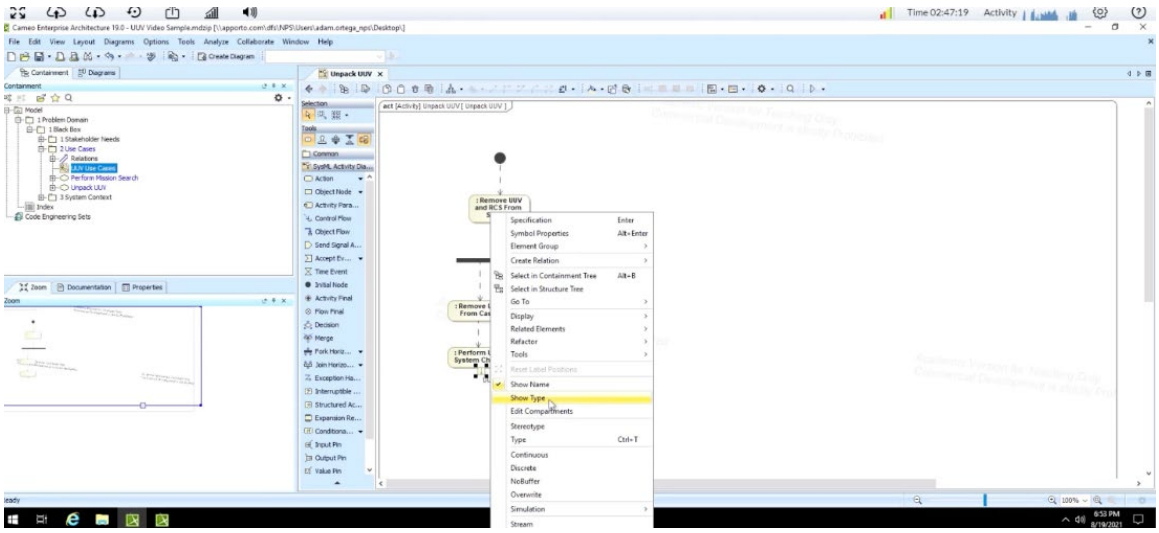
A popup window will appear for specific type and we will start typing *Boolean* and hit the enter key on the keyboard. And we will select the value option for Boolean, which is denoted as a *v* inside a rectangular box next to the word Boolean.



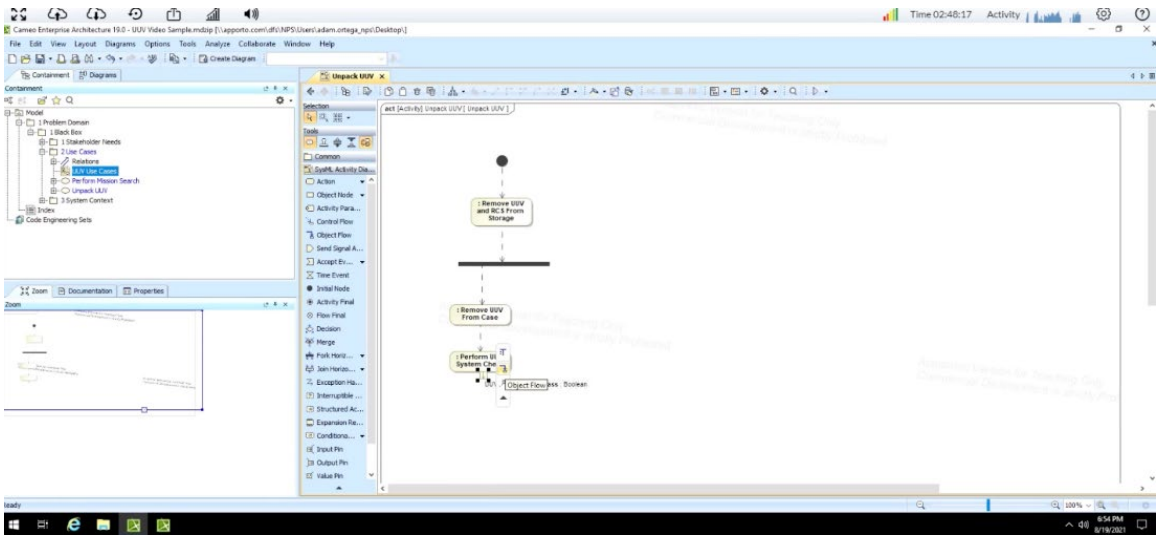
You can also double click to bring up the *Specification of Output Pin output* popup window. Therefore, we will change the output name to *UUV System Check Pass* and then we will close the popup window.



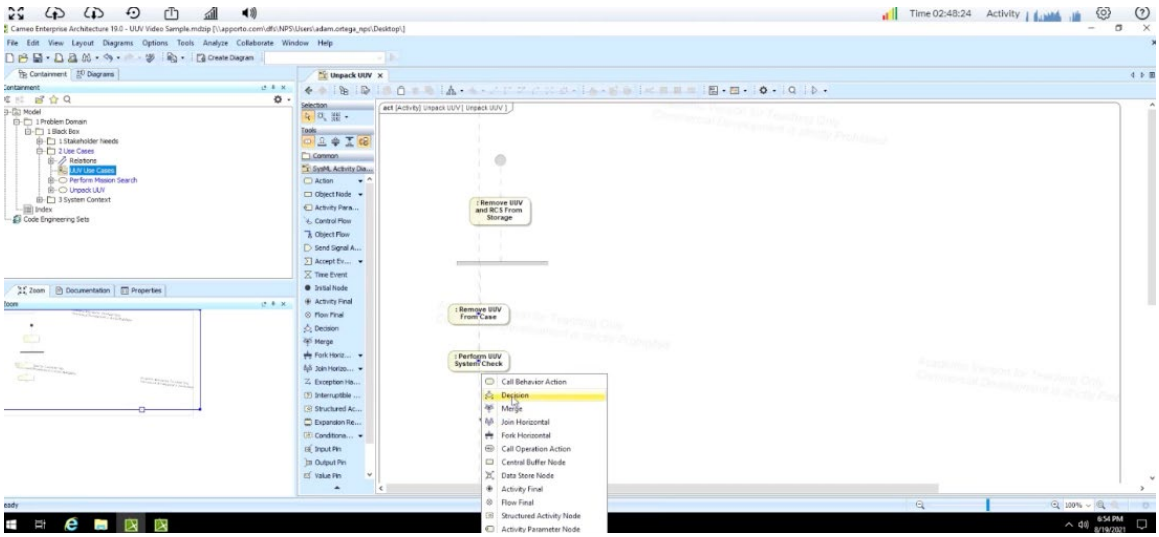
We can also right click on the *Output Pin* to bring a drop-down menu where we will be selecting *Show Type*.



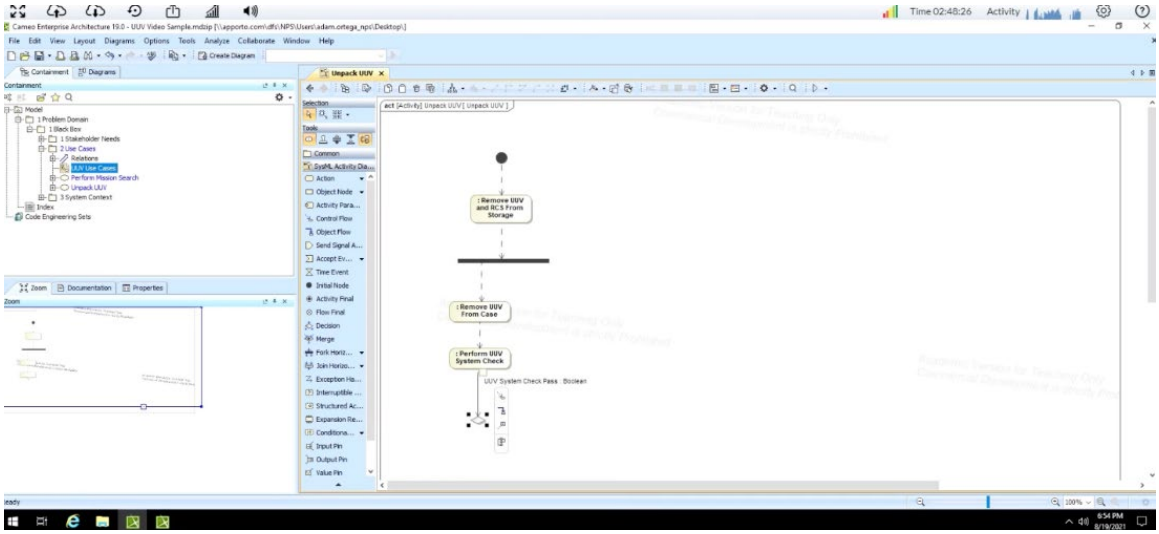
And we can see there is a Boolean. And now from the *Output Pin* we can select the *Object Flow* button on the smart manipulator tool bar.



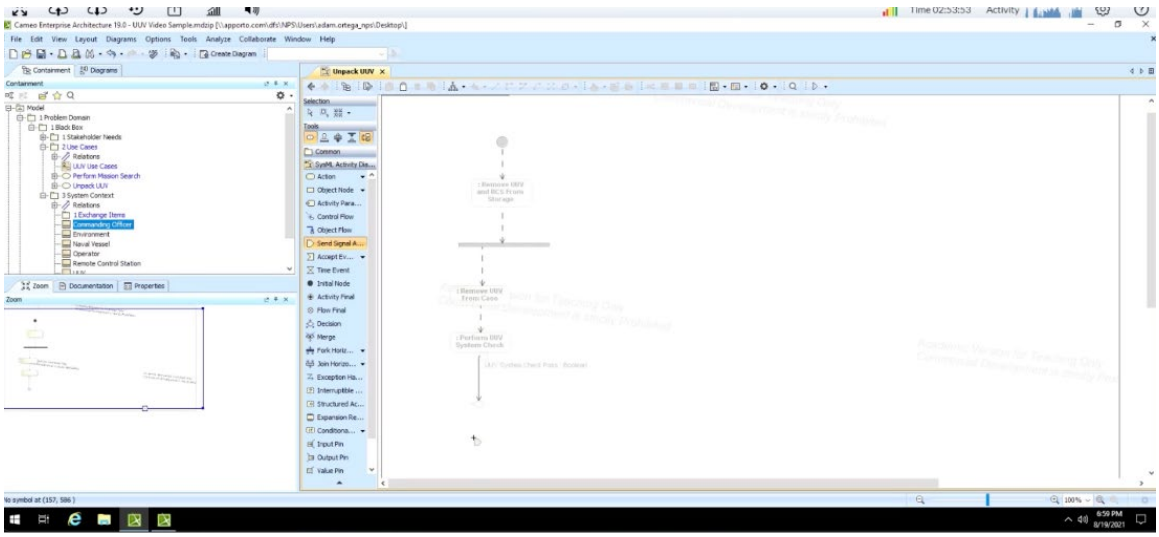
Now we will right click and select the *Decision* node from the drop-down menu.



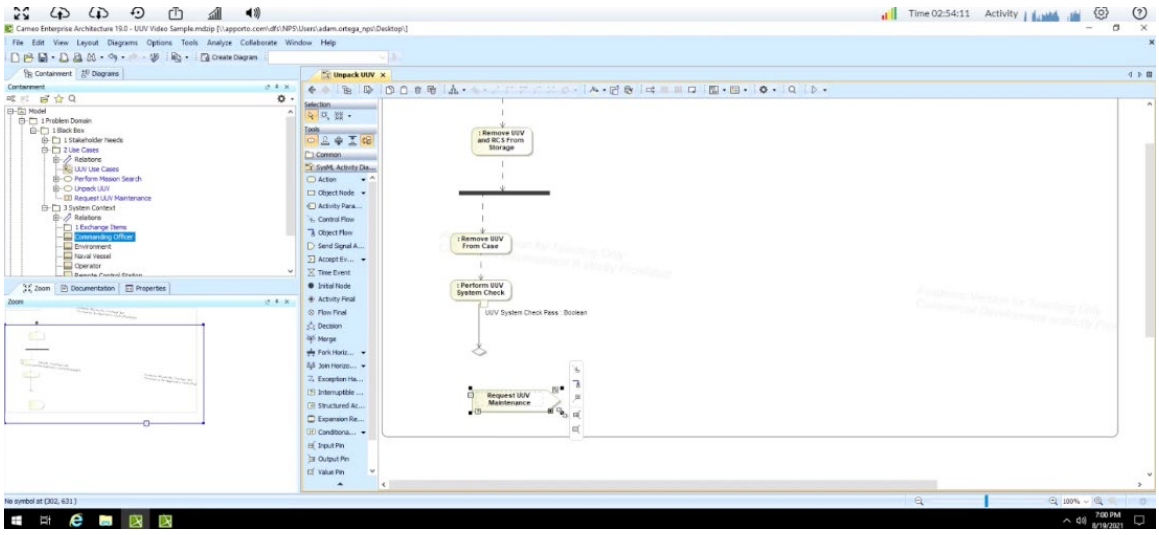
This will make a diamond shape *Decision Node* appear in the diagram pane.



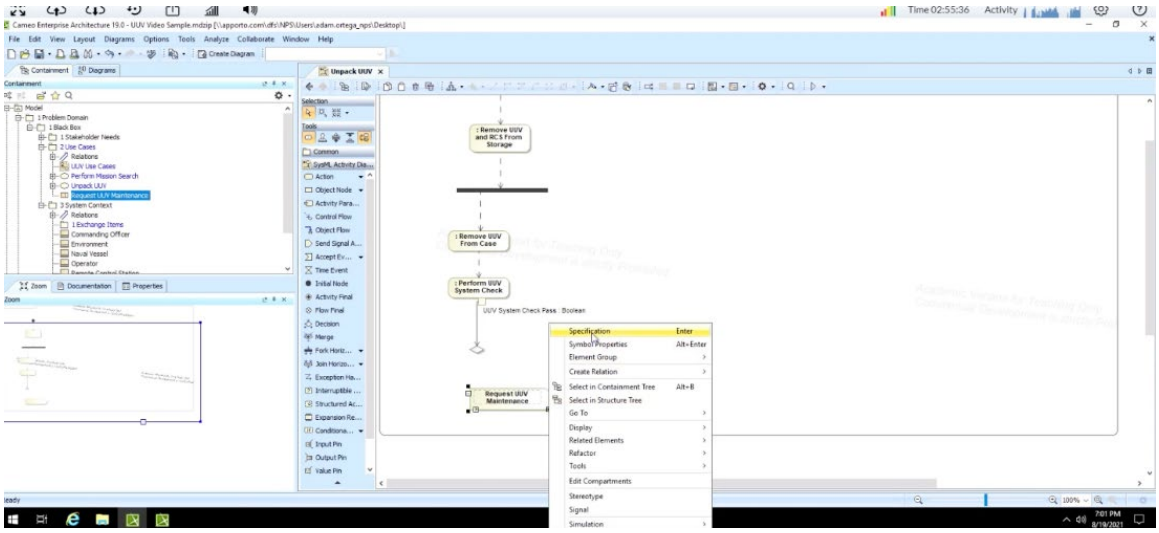
Now before we connect the decision node to alternate paths, we will create a send signal action by clicking on the *Send Signal Action* button in the diagram pallet. And dragging *Send Signal Action* onto the diagram plane and then placing it below the decision node.



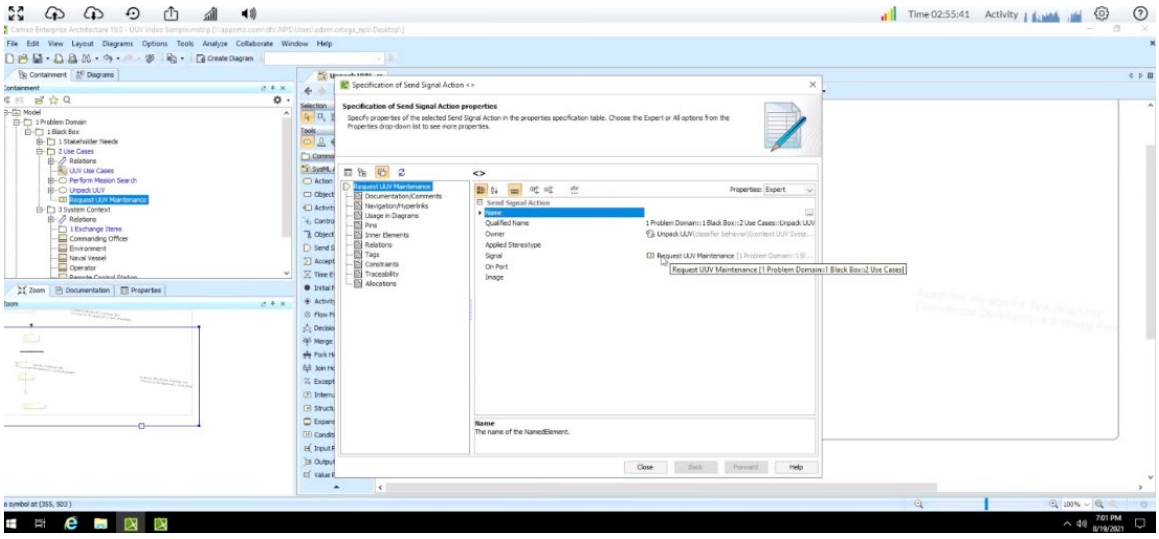
Then we will type *Request UUV Maintenance* for the name of the send signal action. We can go ahead and resize the send signal action to make it more legible and professional looking.



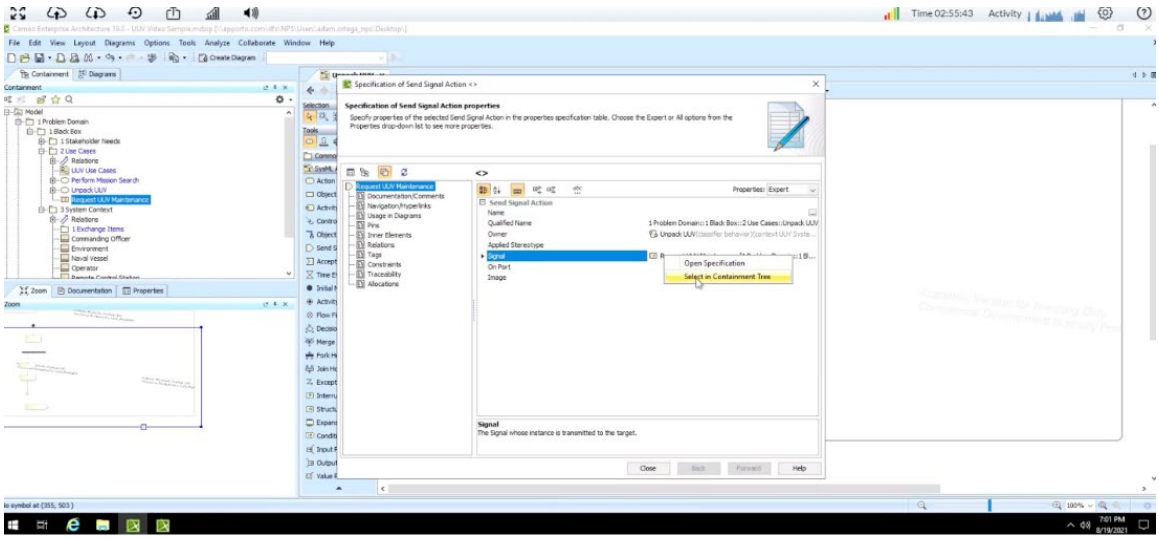
Then we can right click *Request UUV Maintenance* send signal action to bring up the drop-down menu where we will be selecting the *Specification* option.



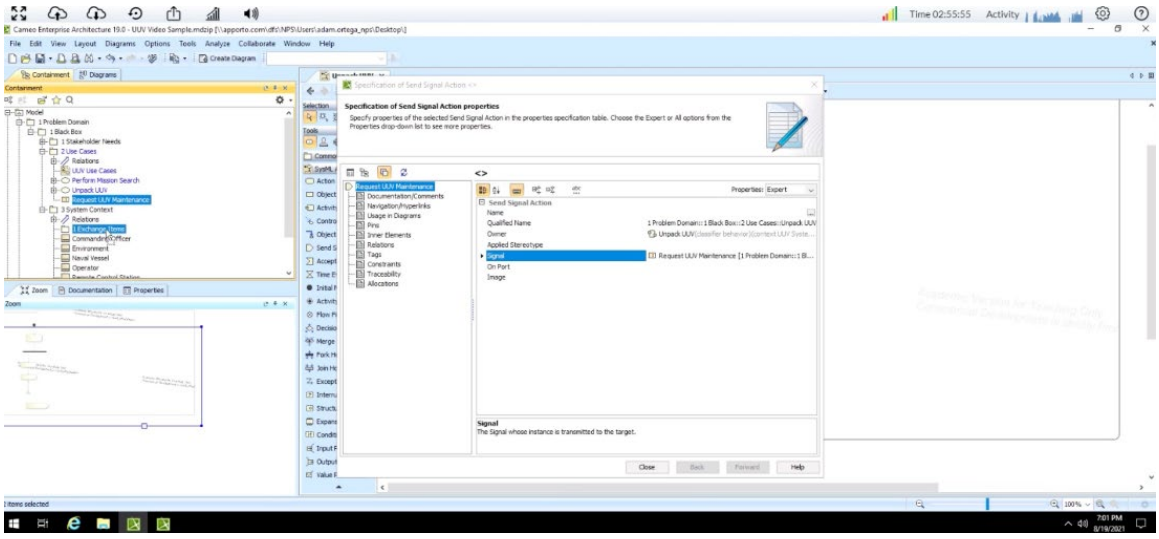
Then the *Specification of Send Signal Action* popup window will appear and we can see that a signal was created.



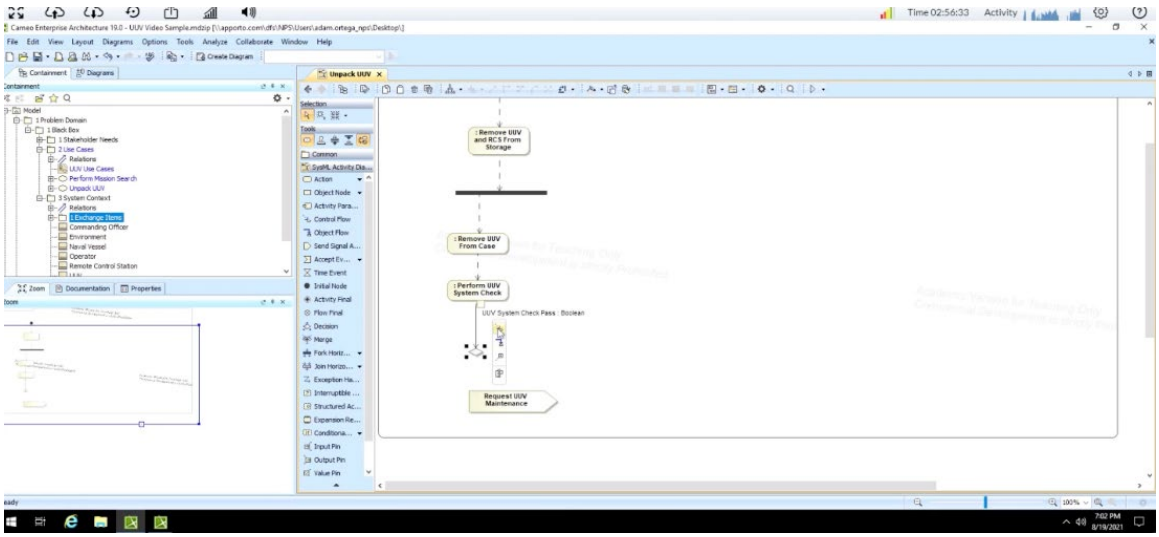
And we will right click on the signal and a drop-down menu will appear and we will select the *Select in Containment Tree* option.



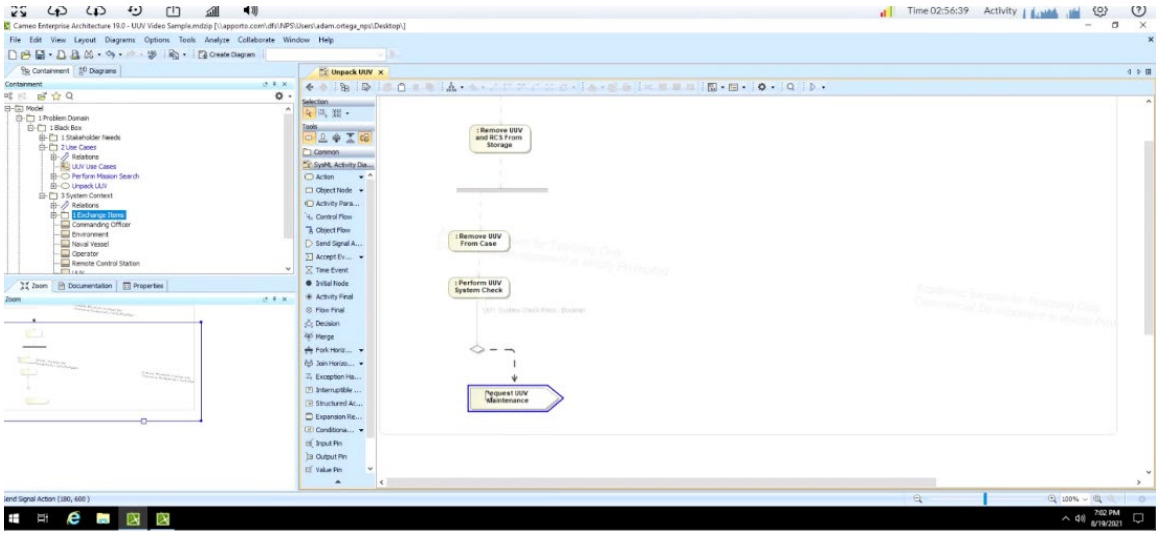
Then we see that is located in the *Unpack UUV* scenario, but we will drag this signal to the *Exchange Items* in the system context. This is to keep the containment tree organized.



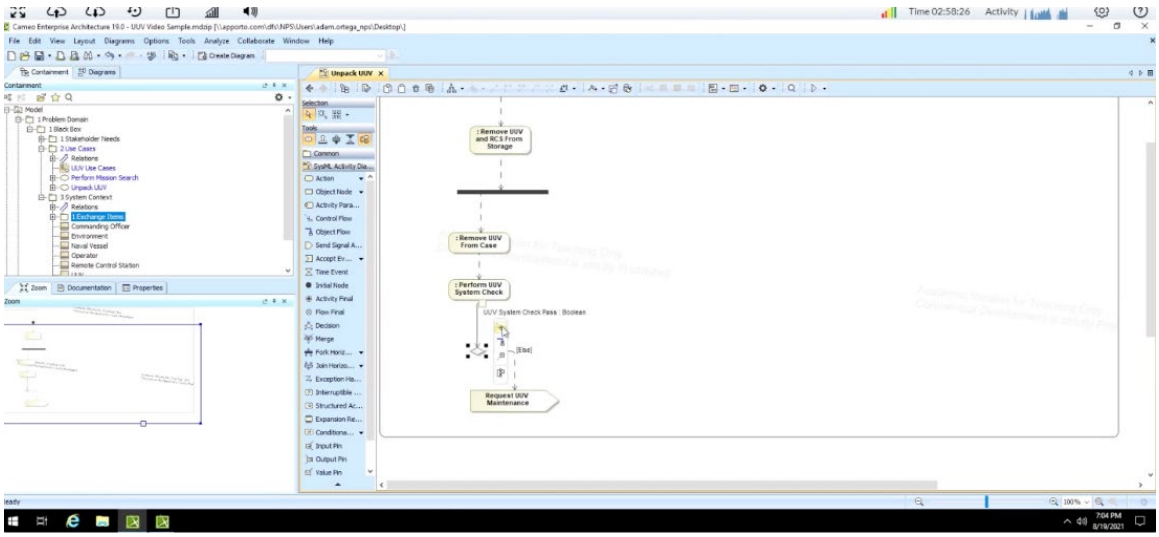
Once that is done, we can continue with the decision node so we can close the *Specification of Send Signal Action* popup window. We click on the *Decision Node* and we will click on the *Control Flow* button in the smart manipulator tool bar.



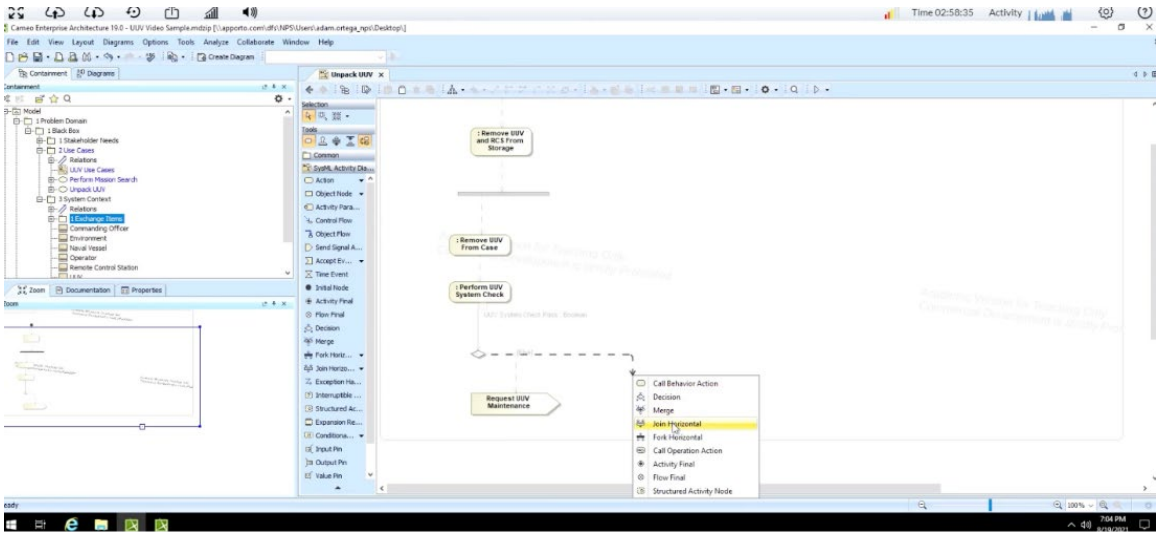
Then we will bring it to the *Request UUV Maintenance* send signal action, which will highlight in blue the send signal action.



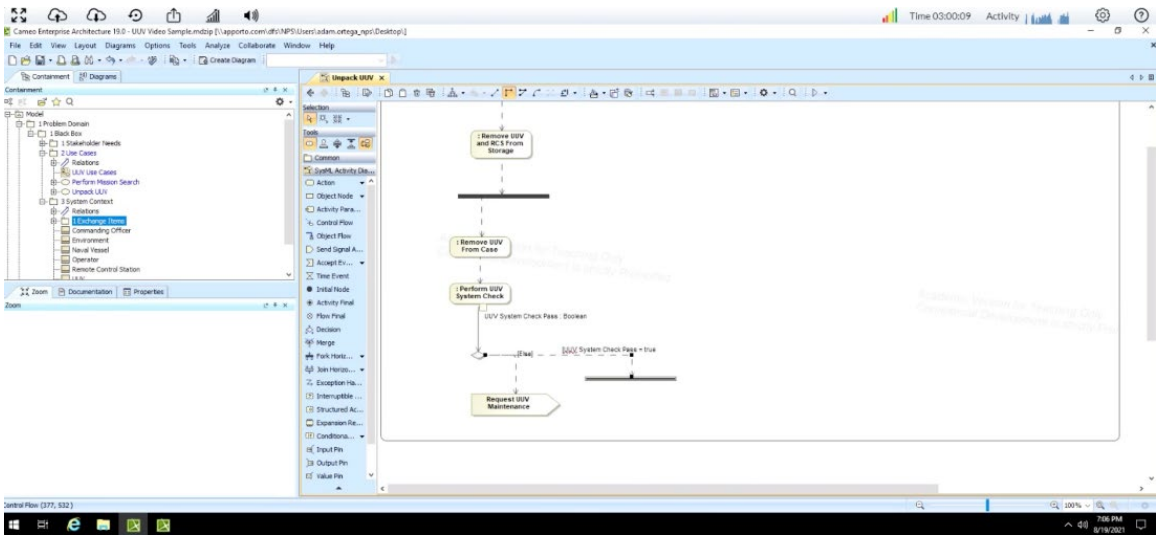
And while the connector between the two is selected we can type *[Else]* and hit the enter key on the keyboard. And this will create a guard for this path. Then we will click on the *Decision Node* and select the *Control Flow* button in the smart manipulator tool bar.



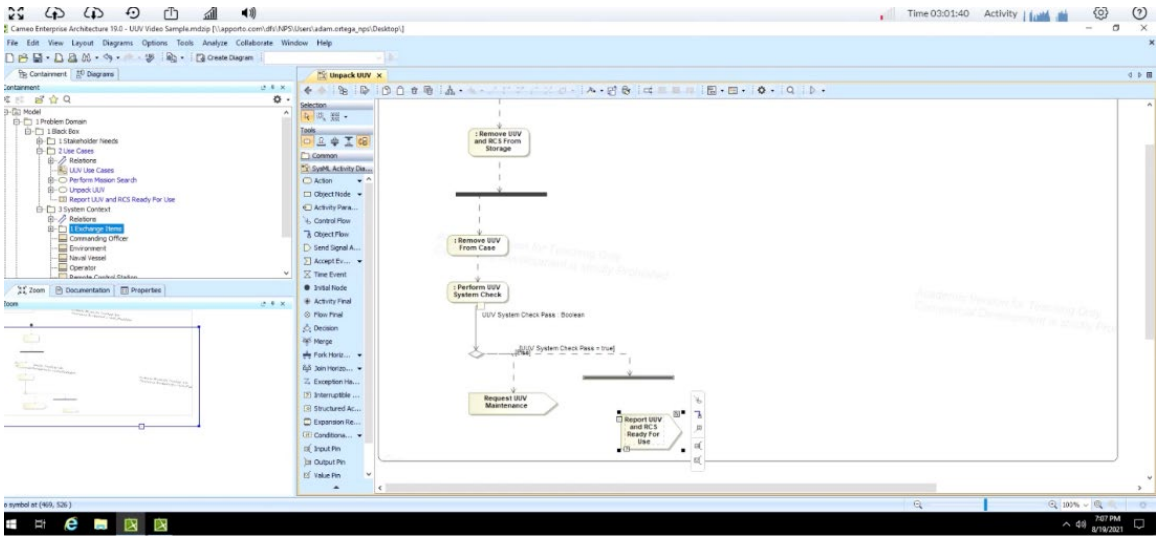
This time we will right click on the empty space of the diagram plane. The drop-down menu will appear and we will select the *Join Horizontal* option.



A black and bold horizontal line will appear on the diagram plane. And again we will select the connector between the two and for this one we will type *[UUV System Check Pass = true]* and then we will hit the enter key on the keyboard.

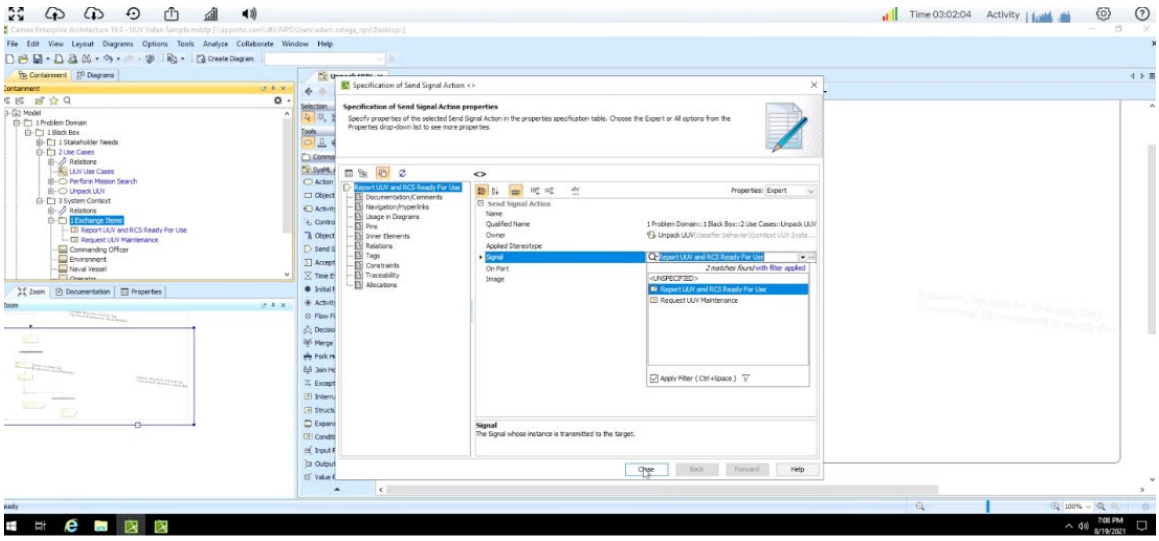


And then from here we will create another send signal action beneath the *Join Horizontal* bar. We will then type *Report UUV and RCS Ready For Use* for the new send signal action.

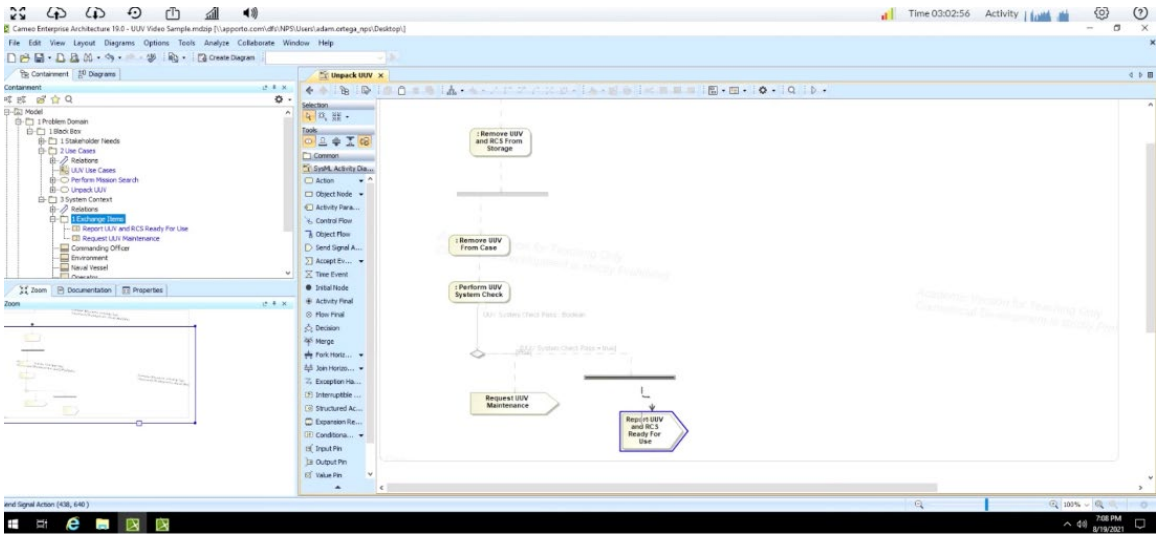


And again we can right click on the *Report UUV and RCS Ready For Use* send signal action to open the drop-down menu and select the *Specification* option. This will then open the *Specification of Send Signal Action* popup window where we can see that the signal was created. Then we will right click on the signal to select the *Select From Containment Tree* option from the drop-down menu. Then we will drag the *Report UUV*

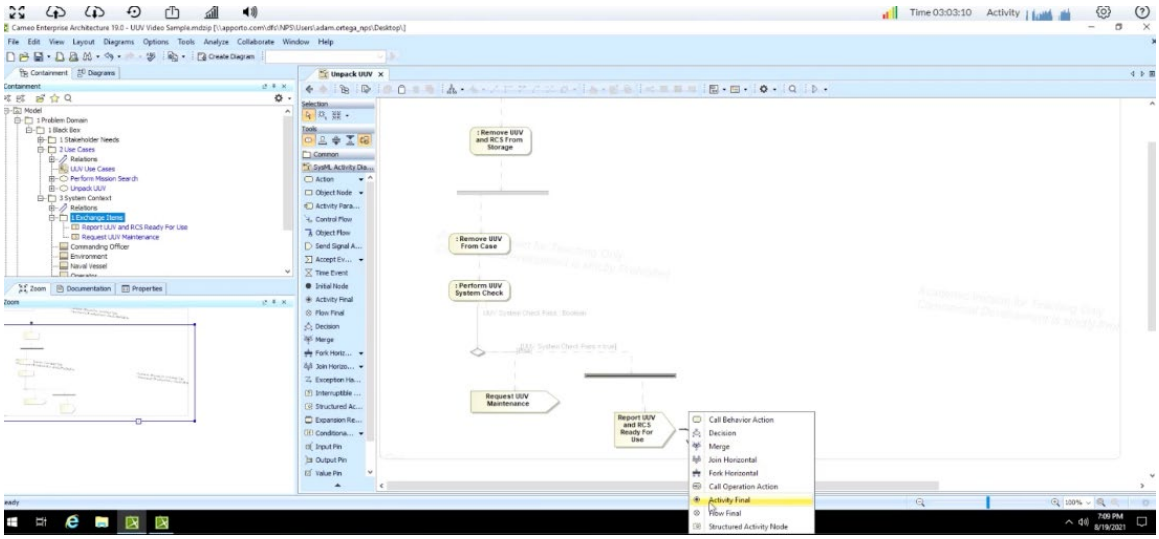
and RCS Ready For Use send signal action from the *Unpack UUV* scenario to the *Exchange Items* folder in the system context. Then we close the *Specification of Send Signal Action* popup window. Under the *Exchange Items* folder in the system context, you will see both signals *Request UUV Maintenance* send signal action and *Report UUV and RCS Ready For Use* send signal action.



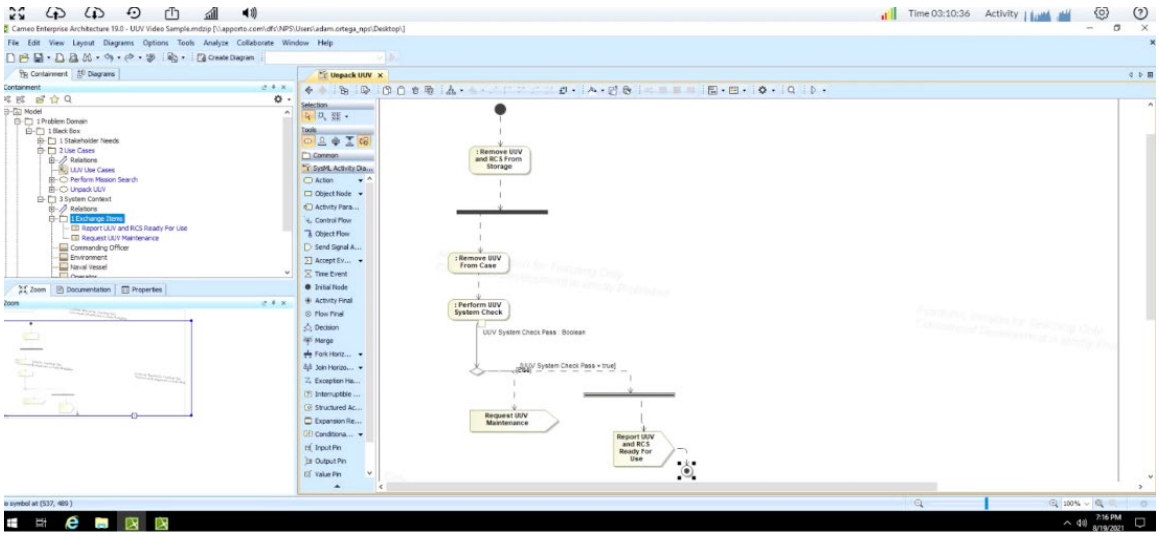
Then we click on the *Join Horizontal* bar and select the *Control Flow* button in the smart manipulator tool bar. Then bring it to the *Report UUV and RCS Ready For Use* send signal action, which will cause the send signal action to be highlighted in blue.



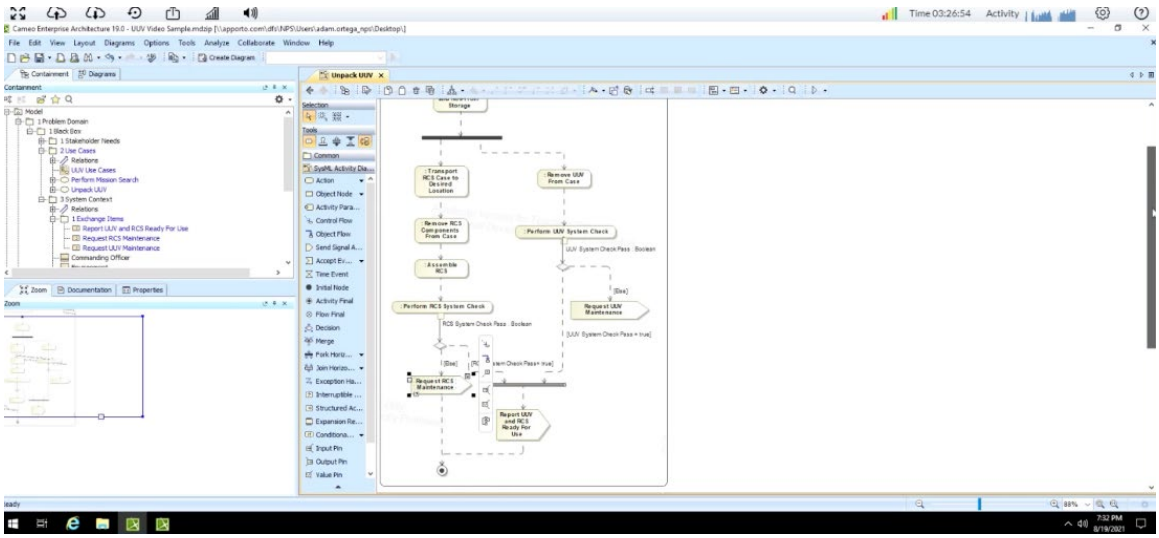
Now we will click on the *Control Flow* button in the smart manipulator tool bar and we will right click on the empty space on the diagram plane. A drop-down menu will appear and we will select the *Activity Final* option.



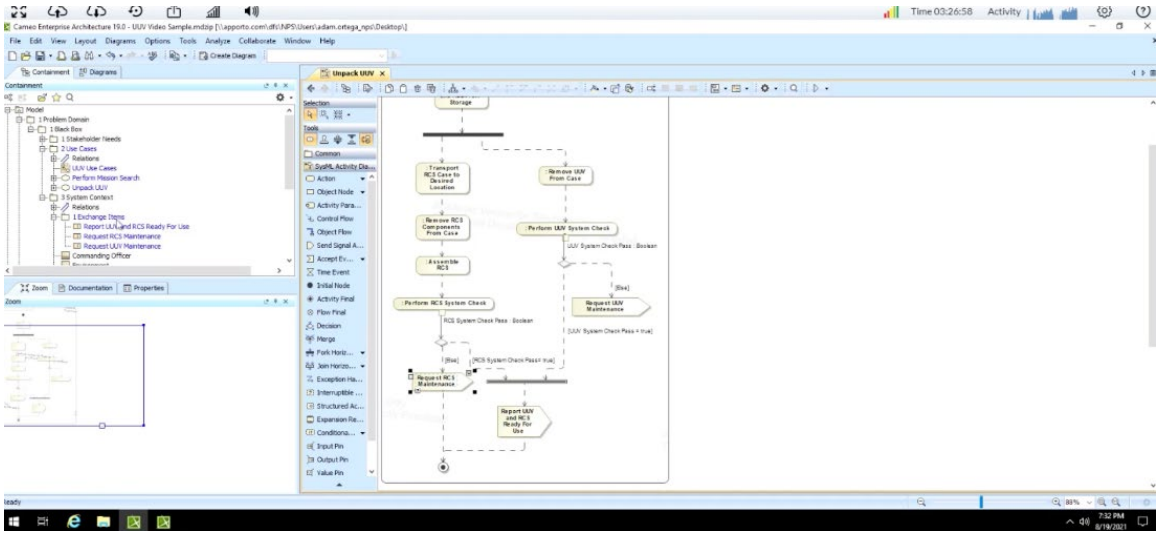
Then an *End Node* will appear on the diagram plane. And now we can go up to the *Fork Horizontal* to complete the alternate path.



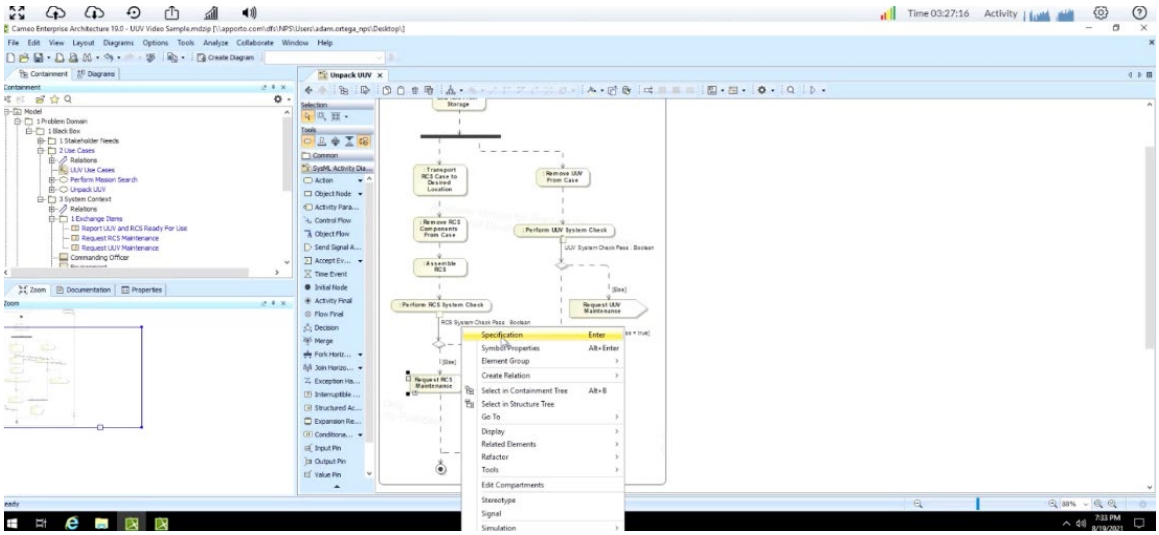
Now repeat the steps above to create the alternate path. Once the completion of the alternate path the diagram show look similar to this.



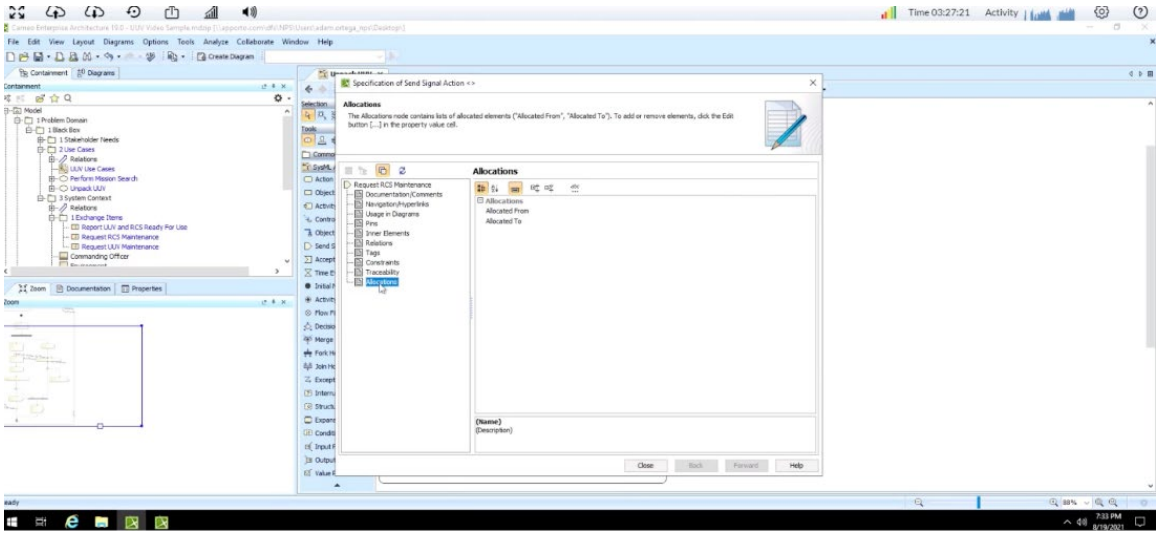
We have our set of signals under the *Exchange Items* folder in the system context.



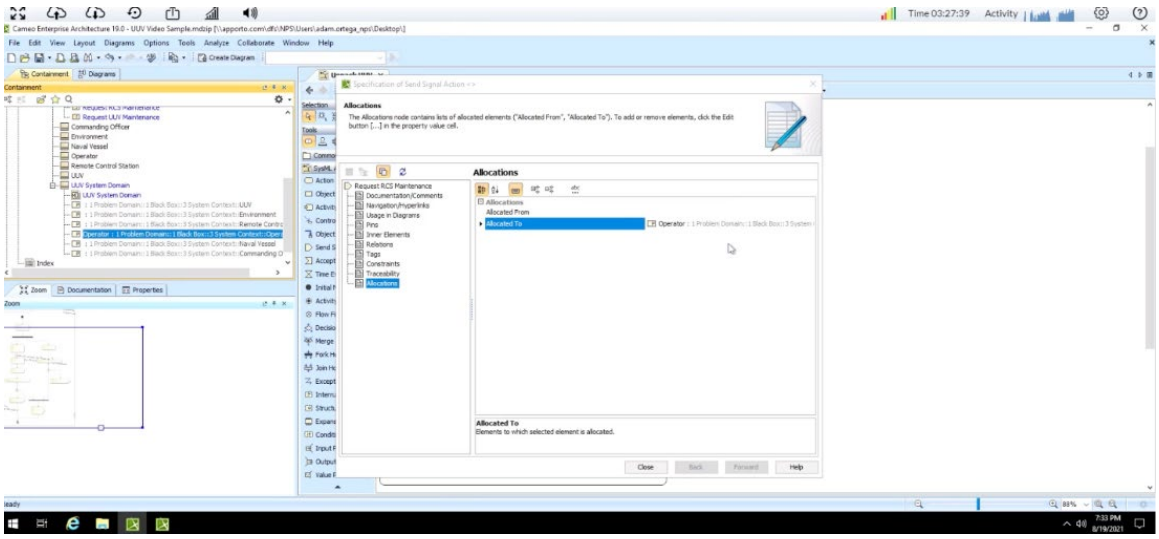
Now for the final steps, we will add an allocation to two of the send signal actions. So we will start with the *Request RCS Maintenance* send signal action and we will right click and select *Specifications* from the drop-down menu.



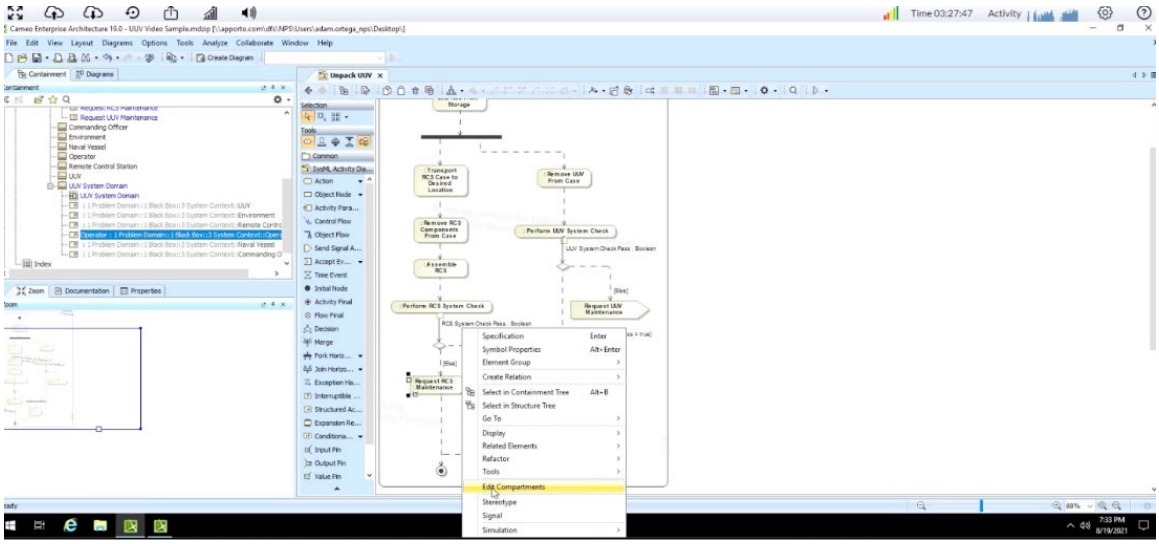
This will then open the *Specification of Send Signal Action* popup window and on the left side menu we will select *Allocation*. On the right side of the popup window, we are able to see the *Allocation To*.



And now we will go over to the containment tree and under the *UUV System Domain* block we will select the *Operator* part property and drag it to the *Allocation To*. And then we can close the *Specification of Send Signal Action* popup window.

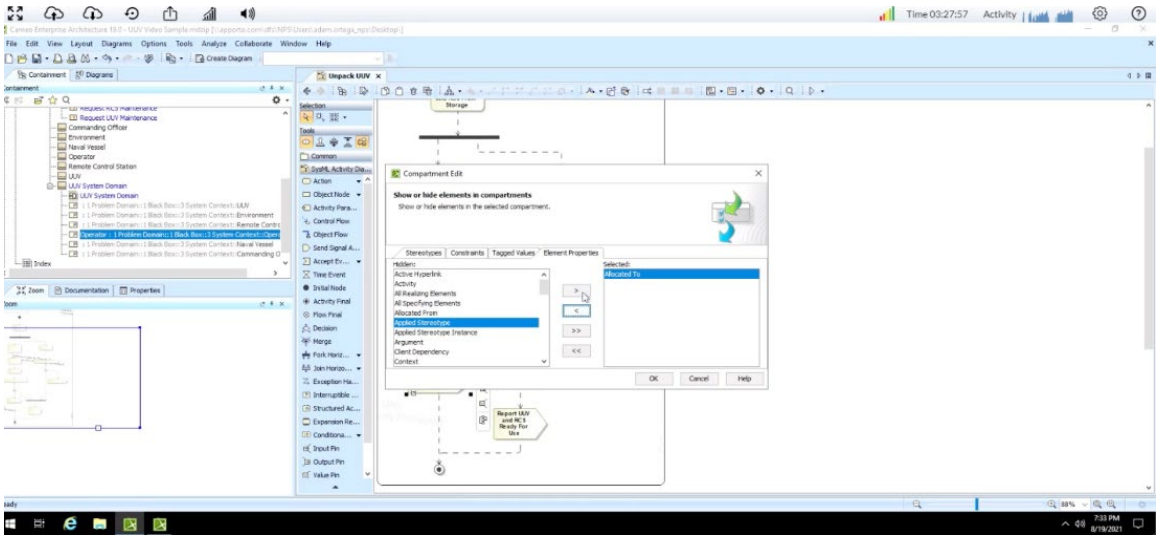


We will right click the *Request RCS Maintenance* send signal action and select *Edit Compartment* from the drop-down menu.

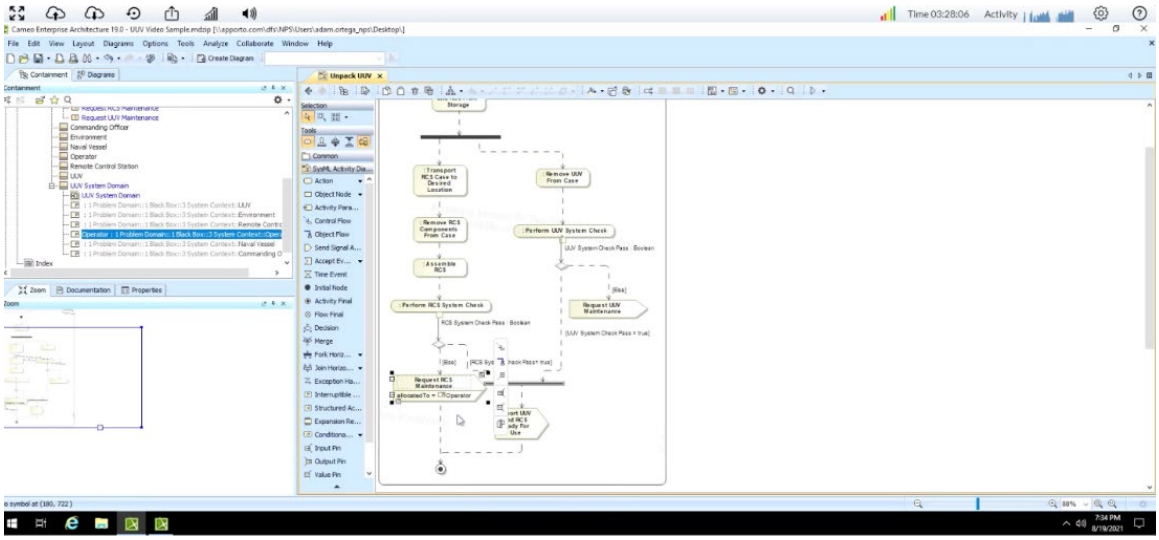


This will then make the *Edit Compartment* popup window appear. Then we click on the *Element Properties* tab and from the Hidden section list select *Allocation To* and

click on the > button to add to the select list. And then we select the *Ok* button to close down the *Edit Compartment* popup window.

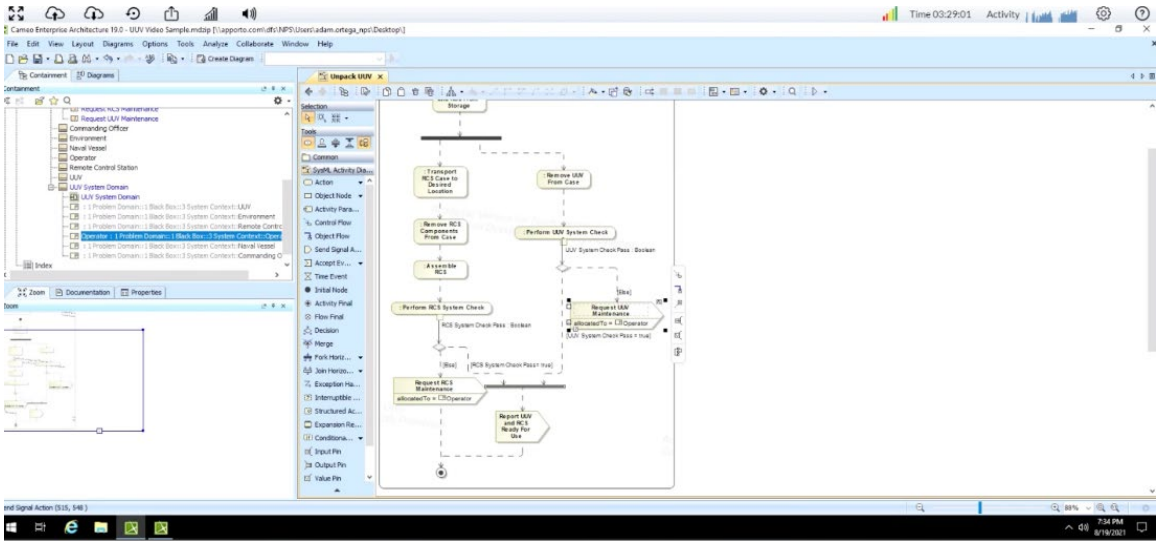


Now we can see that the *Request RCS Maintenance* send signal action is allocated to the *Operator*.



Now we will repeat the same process to the *Request UUV Maintenance* send signal action. So we will start first by right clicking and selecting *Specifications* from the drop-down menu. The *Specification of Send Signal Action* popup window will appear and on the

left side menu we will select *Allocation*. We will select *Allocation To*. And again we will go over to the containment tree and under the *UUV System Domain* block we will select the *Operator* part property and drag it to the *Allocation To*. And we will close the *Specification of Send Signal Action* popup window. Then we will right click *Request UUV Maintenance* send signal action and select *Edit Compartment* from the drop-down menu. Then the *Edit Compartment* popup window appear and we click on the *Element Properties* tab and from the Hidden section list select *Allocation To* and click on the > button to add to the select list. And then select the *Ok* button to close down the *Edit Compartment* popup window. Now we can see that both signals are allocated to the *Operator*. And the final diagram should look similar to this.



And this concludes Part II of the Use Cases tutorial. Thank you.

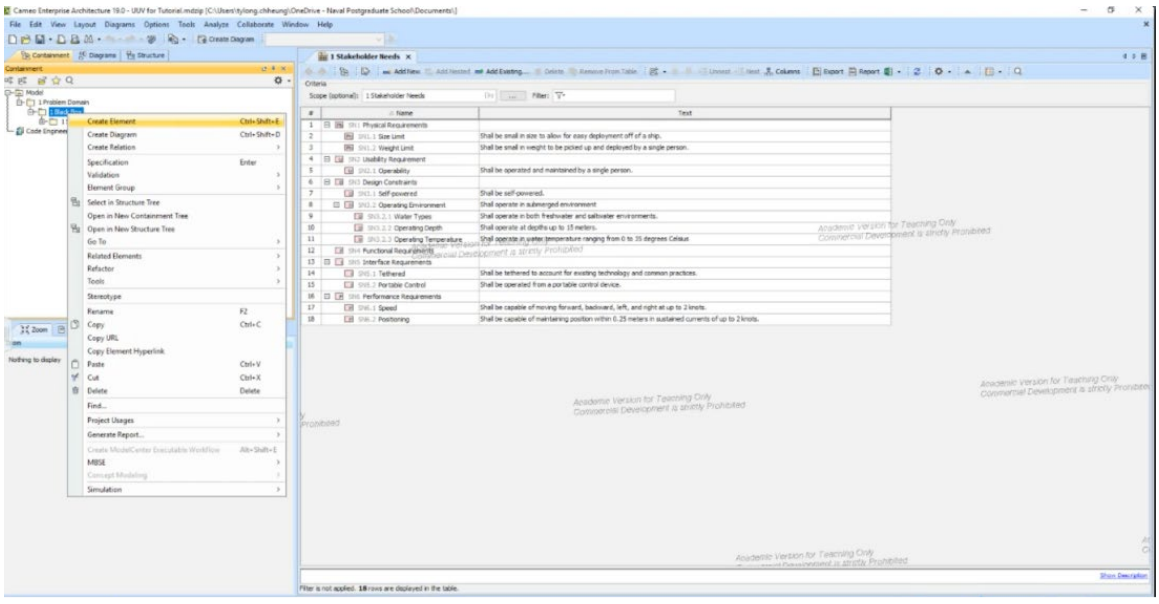
B3: Initial System Context Tutorial

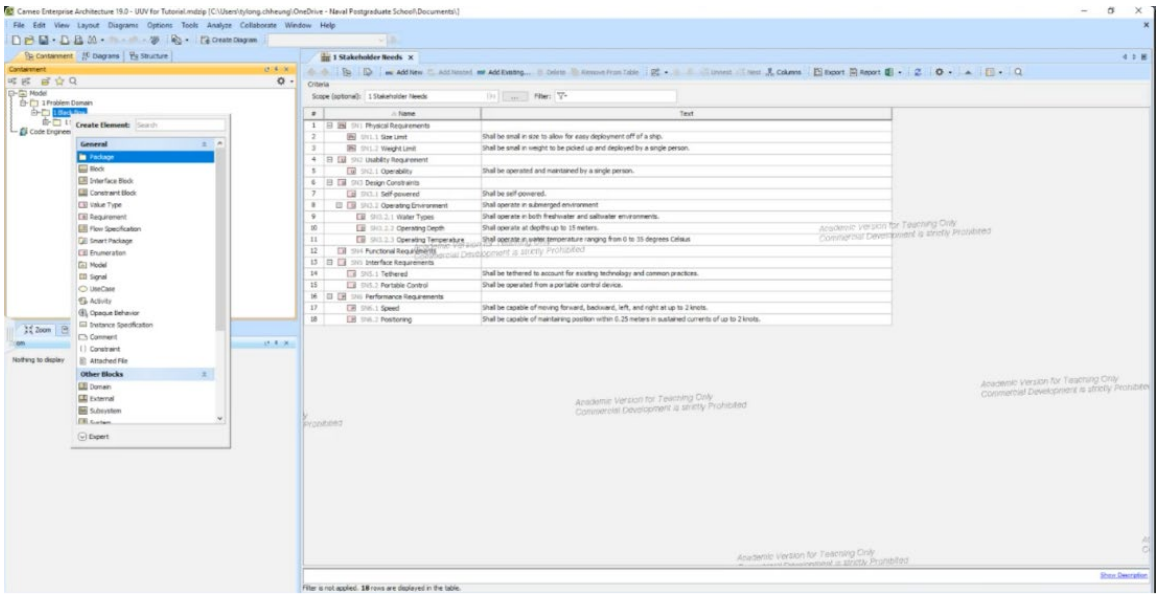
Background:

In this section, the system context is created to illustrate the overall domain of the system of interest. Internal and external boundaries can be defined, as well as other systems or users that interact with the system of context. In the initial phase, elements that partake in the system context are identified, and then specified in the final phase. Each part of the system context is represented in a SysML internal block diagram. Each part of the system context is typed by a block, which is used to define relationship or behavior of the system of interest. The system context can then be used to analyze behaviors of the system in various use cases.

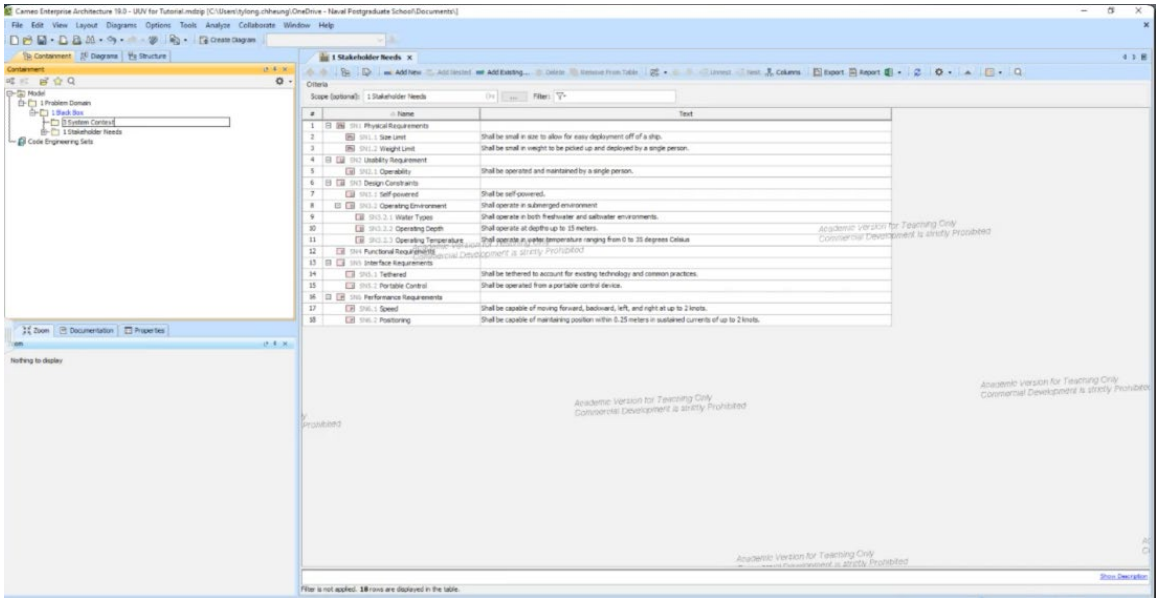
Step-By-Step Guide:

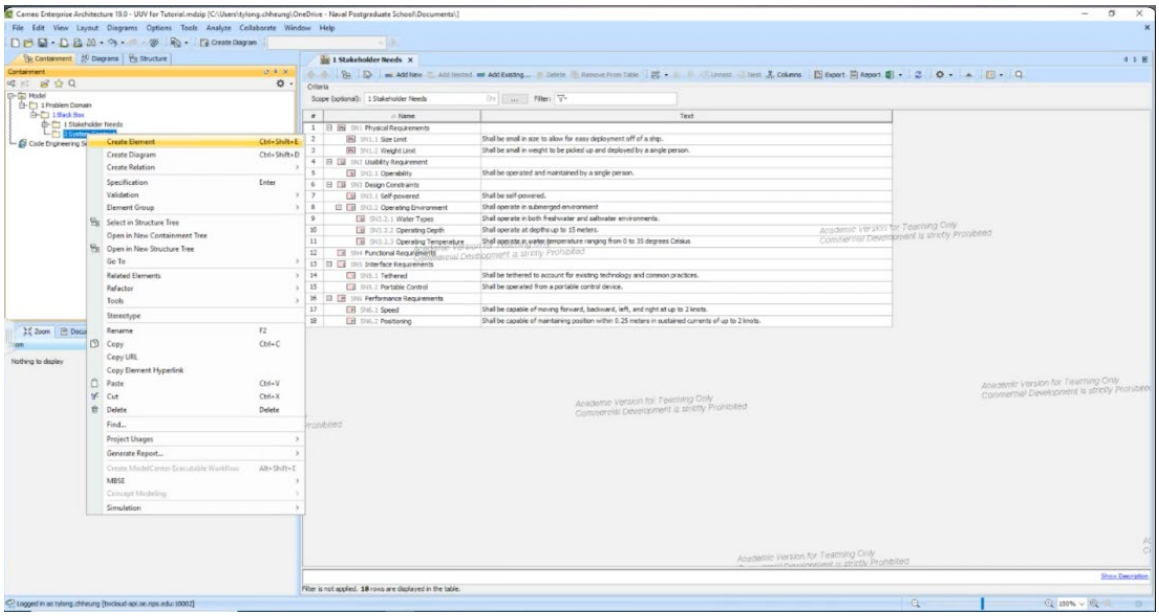
In this video, we are going to cover the initial system context for the UUV. We start up by clicking on *1 Black Box* and from the drop-down menu we select *Create Element*. Then another drop-down menu will appear for *Create Element* and we will select *Package* element for the element type.



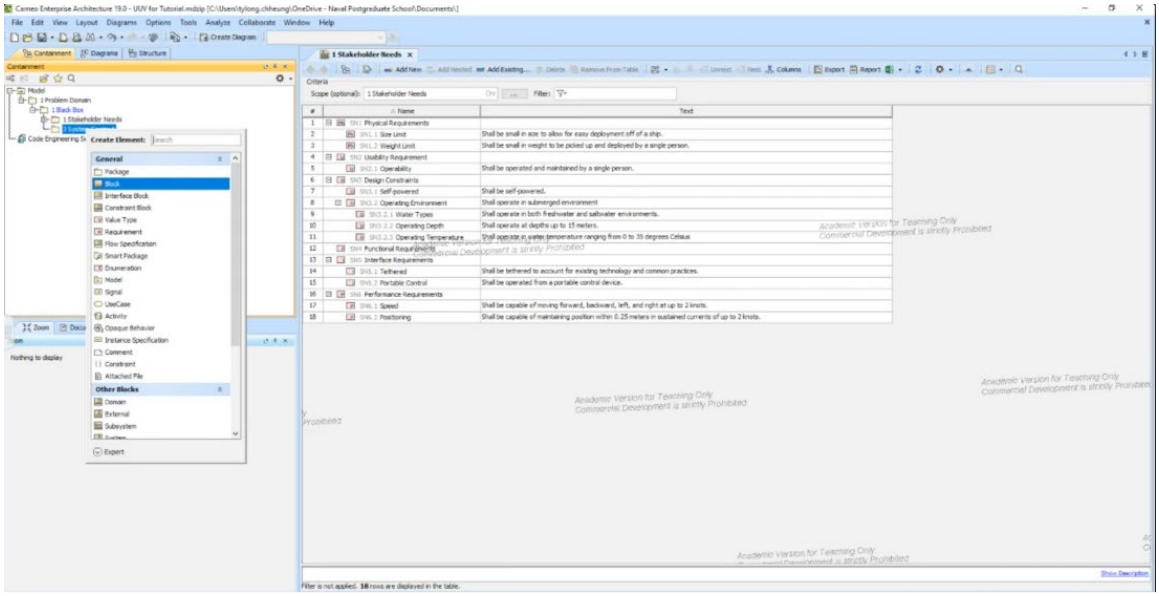


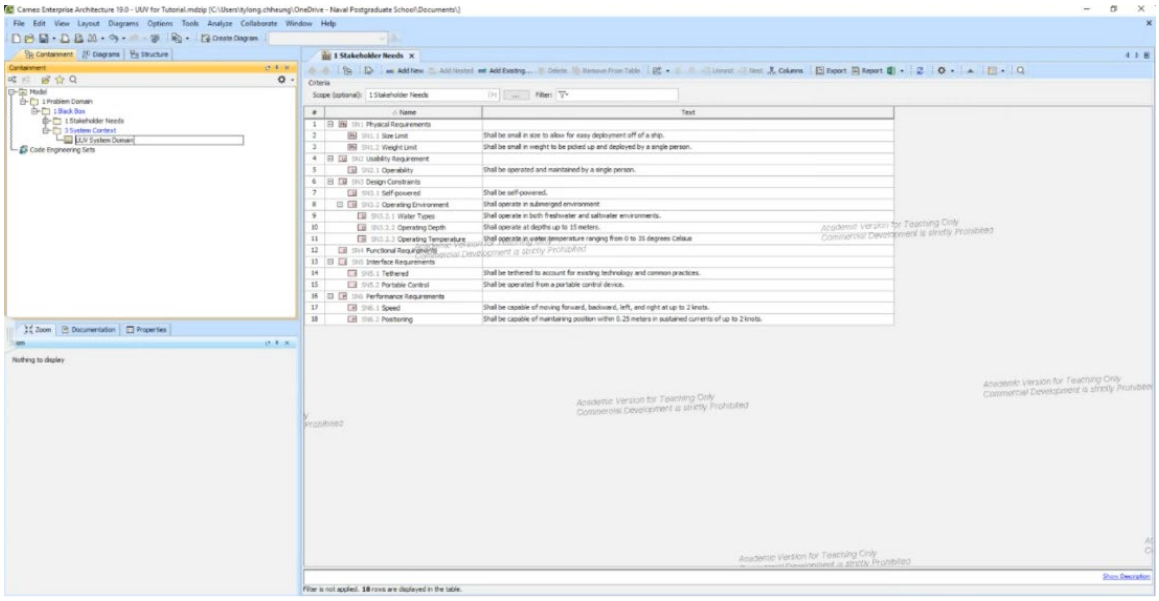
Label the package element as *3 System Context* and then hit enter to finish typing. From here we right click the *System Context* package and once again from the drop-down menu select *Create Element*.



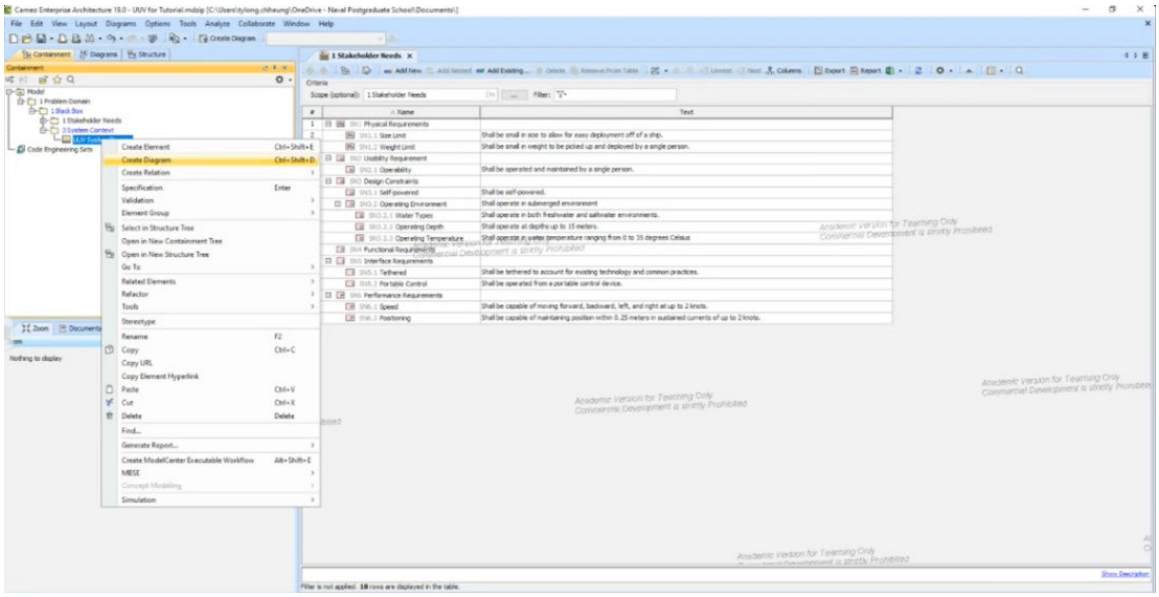


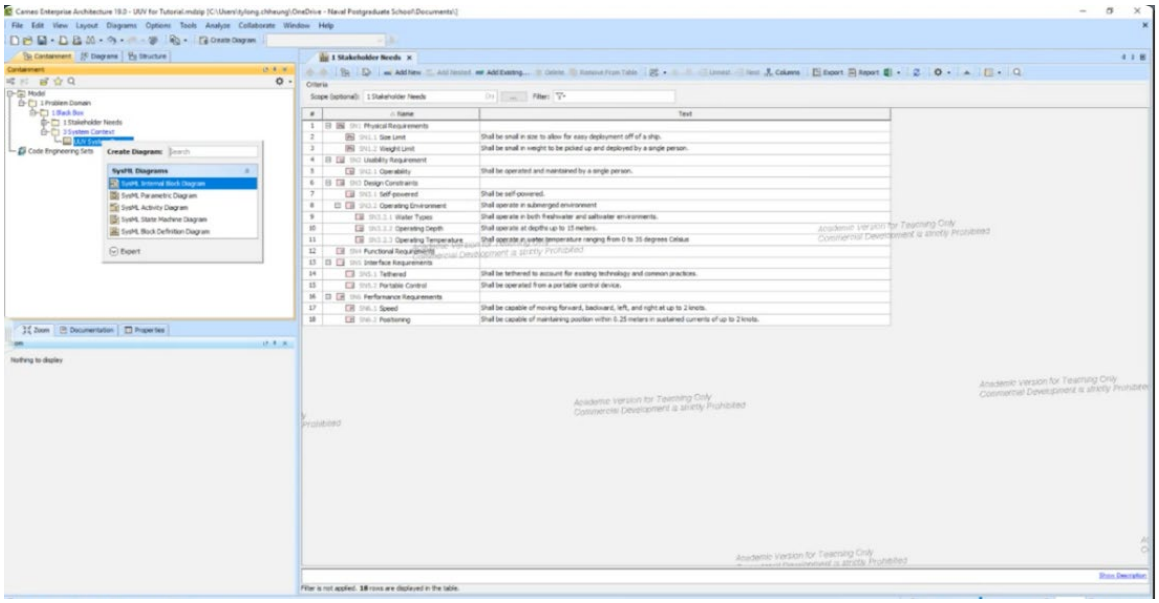
Then another drop-down menu will appear for *Create Element* and, in this case, we select the element type which is a *Block*. And for the *System Context* and label that *UUV System Domain*.



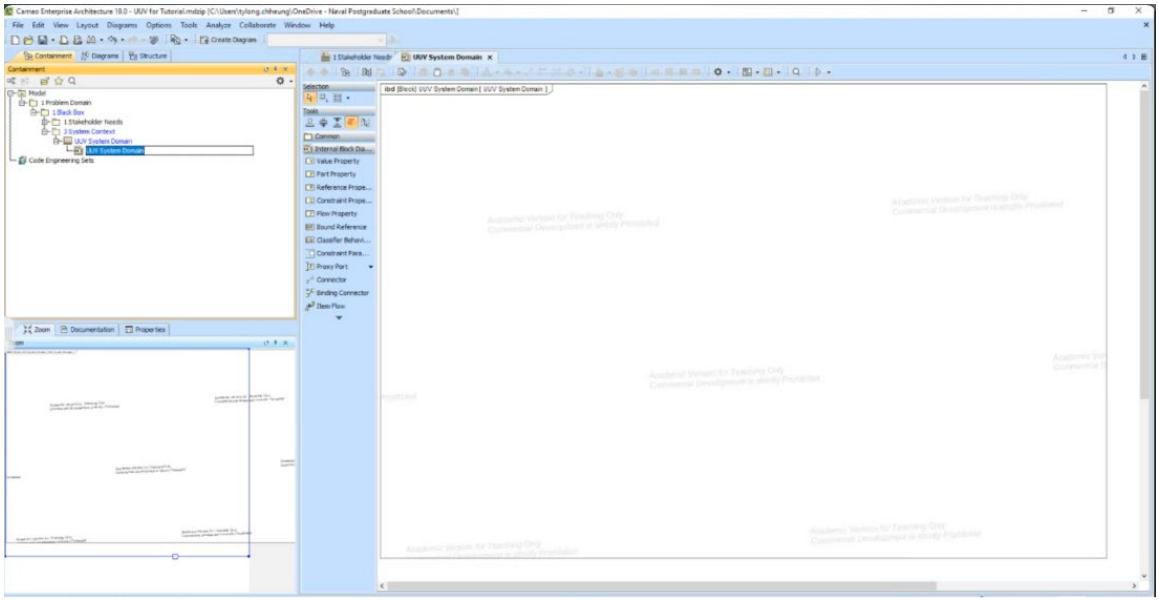


Then we right click the block *UUV System Domain* and from the drop-down menu this time select *Create Diagram*. Then another drop-down menu will appear for *Create Diagram* and we will select the *SysML Internal Block Diagram (IBD)*.

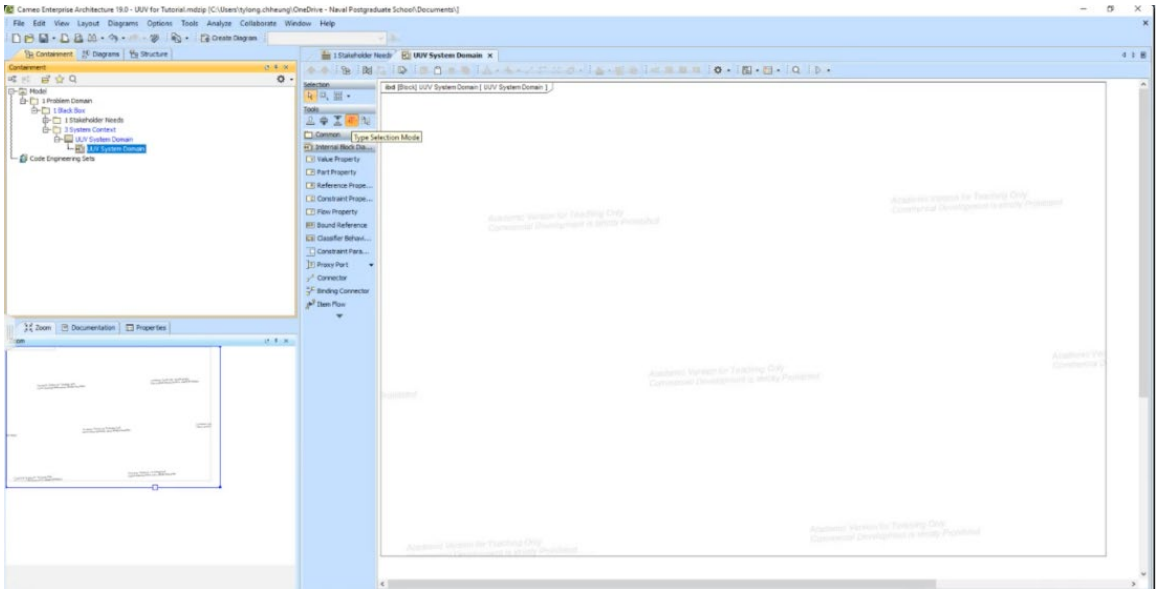


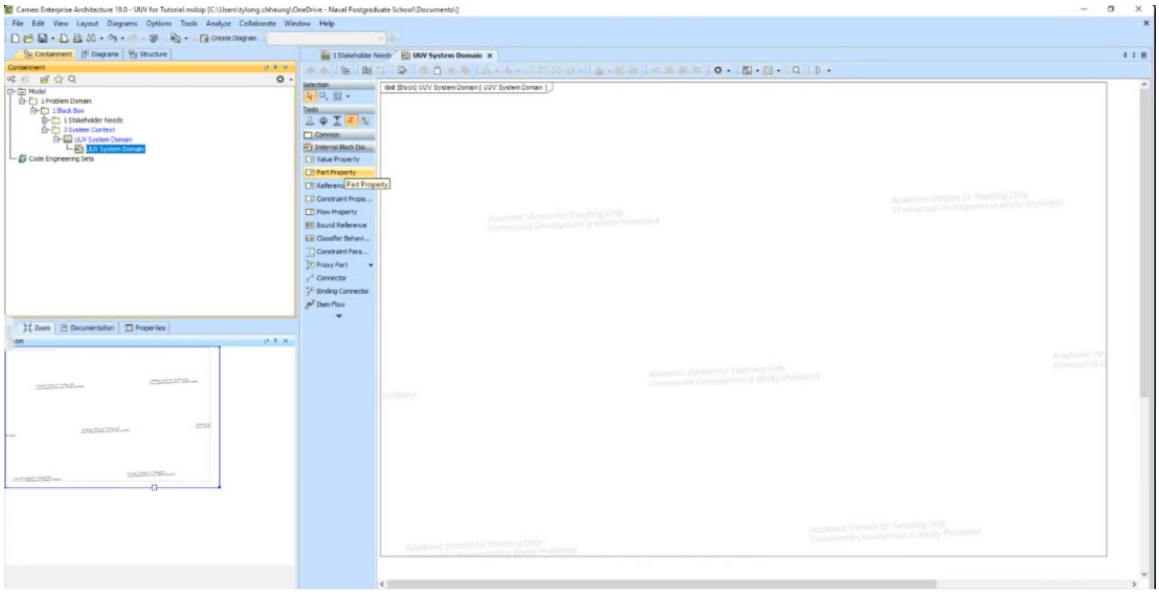


Double click that option and it will generate IBD for you for *UUV System Domain*.

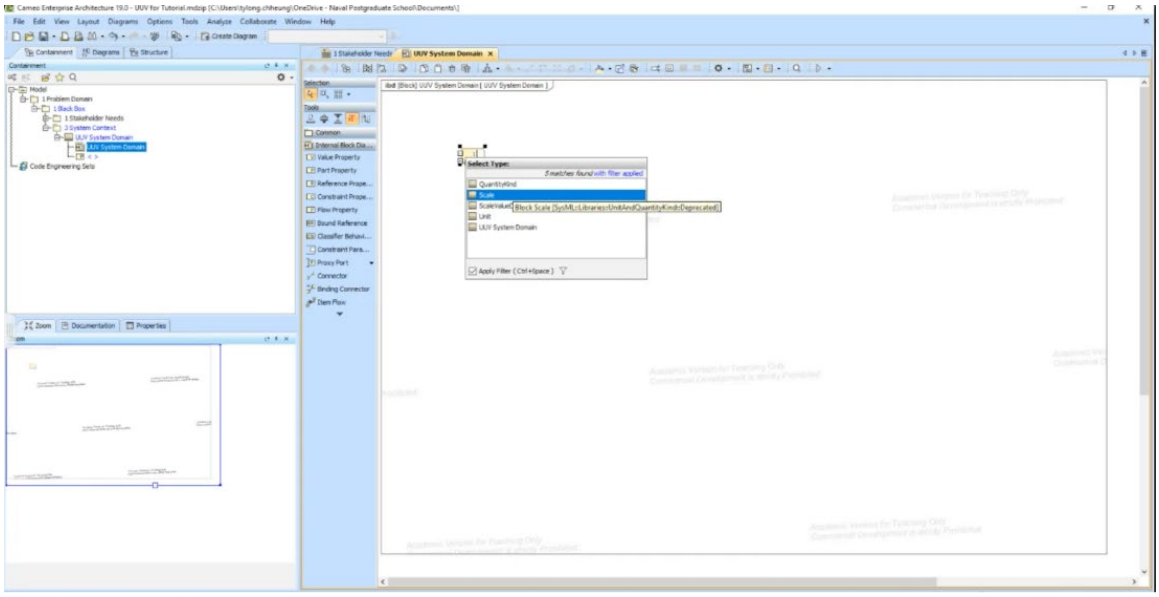


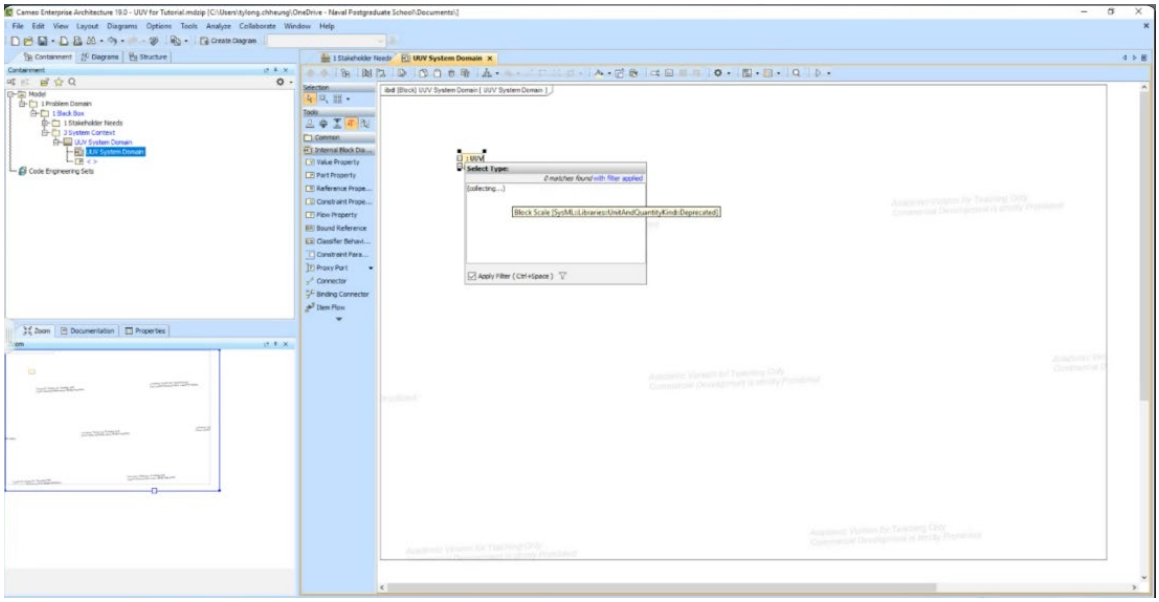
With the IBD open, we ensure that the *Type Selection Mode* is enabled on the IBD menu. *Type Selection Mode* must be enabled in order to type out the *Part Property*. Then we start off with building the first part property by selecting the *Part Property* option on the IBD menu.



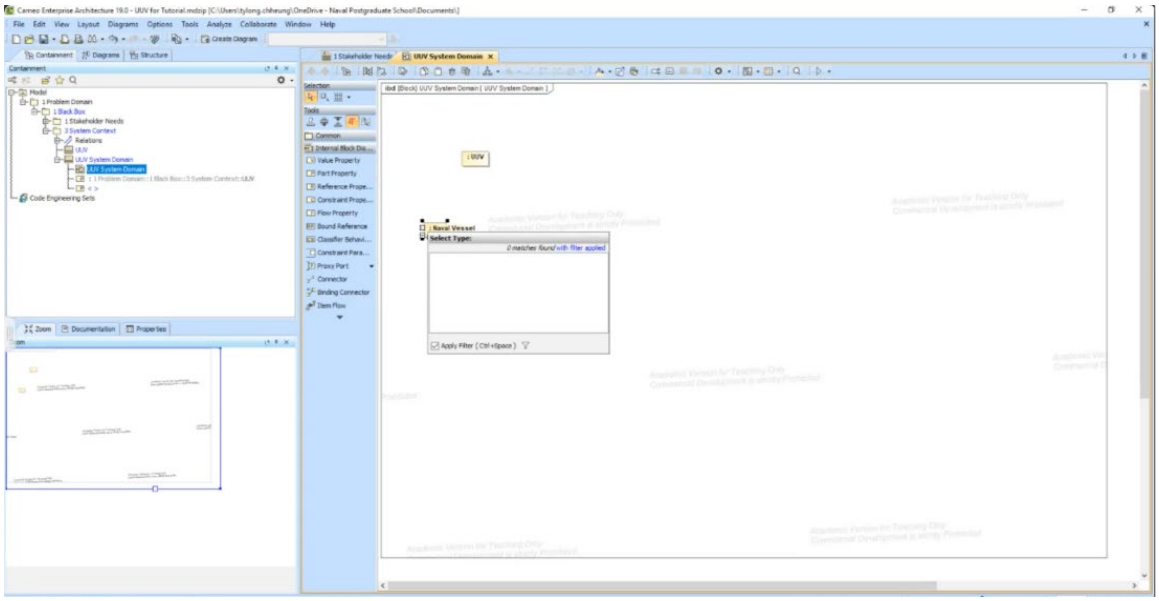


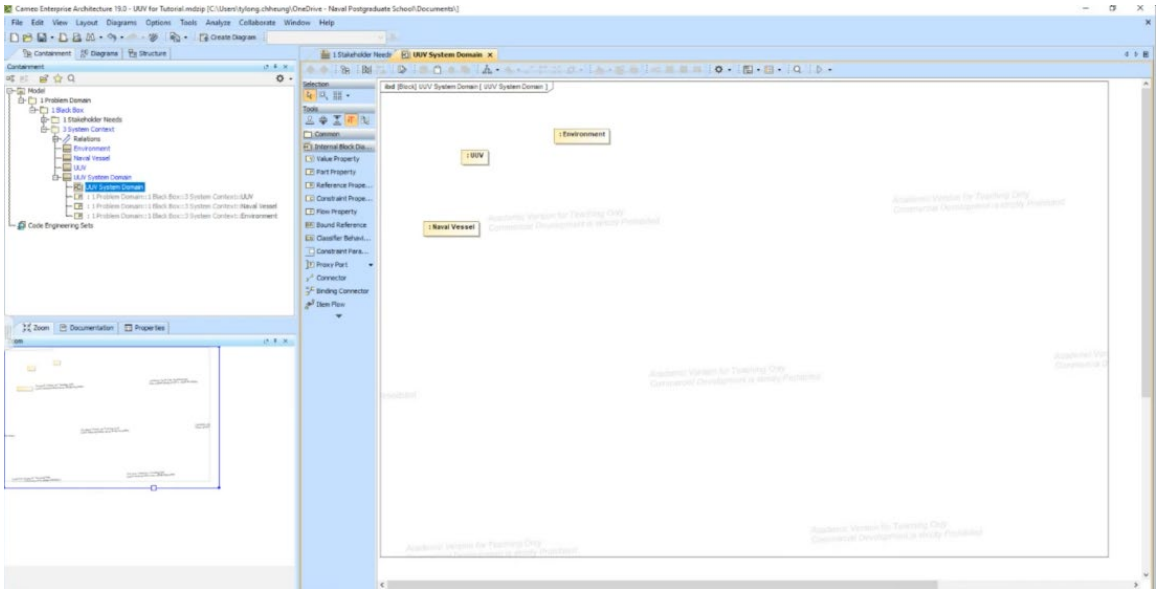
Then we click anywhere on the empty IBD. Then a pop-up menu will appear for *Select Type*. For this Part Property, we are going to type *UUV*.



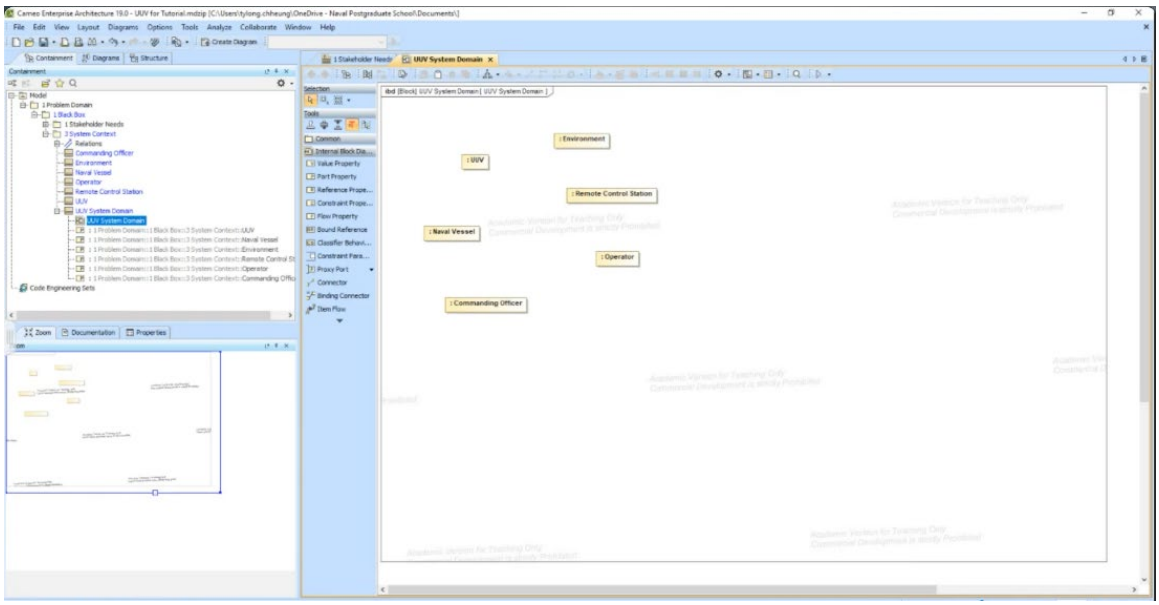


Then repeat this step of selecting *Part Property* and clicking on the empty space of the IBD. For the second one, we are going to type *Naval Vessel* for the part property. Once again select the *Part Property* option on the IBD menu. Then we click anywhere on the empty IBD and this time type *Environment* for the part property.





Repeating those two steps over and over until we get all six of our part properties. The six Part Properties are *UUV*, *Naval Vessel*, *Environment*, *Remote Control Station*, *Operator*, and *Commanding Officer*.



These six Part Properties are what we are looking for the *UUV System Domain*. Thank you.

B3: Final System Context Tutorial

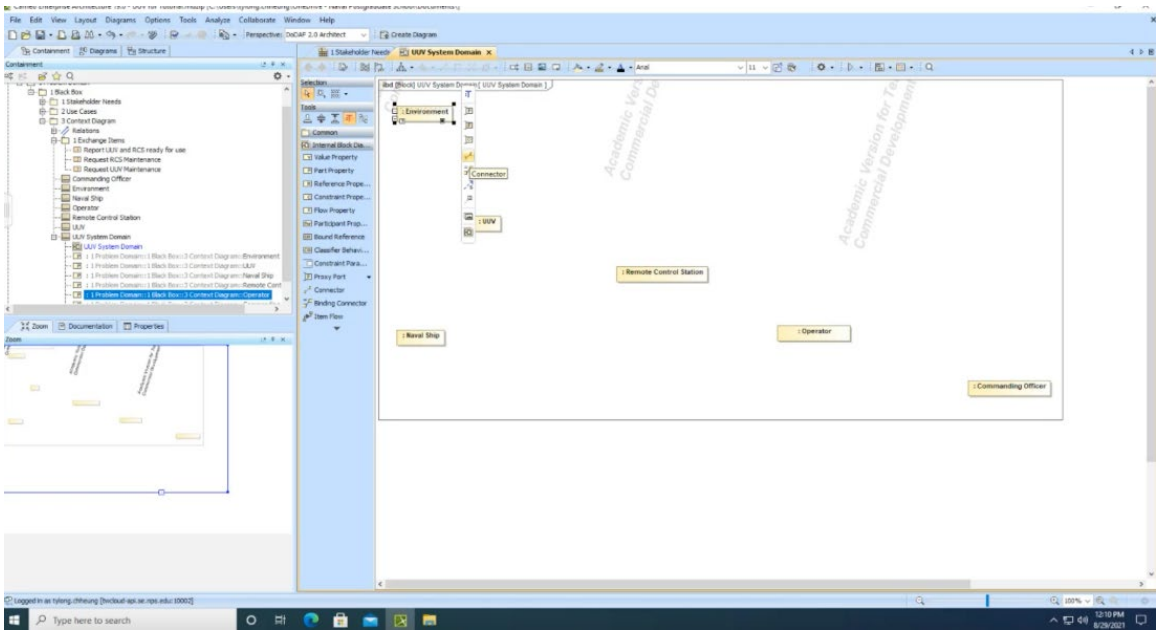
Background:

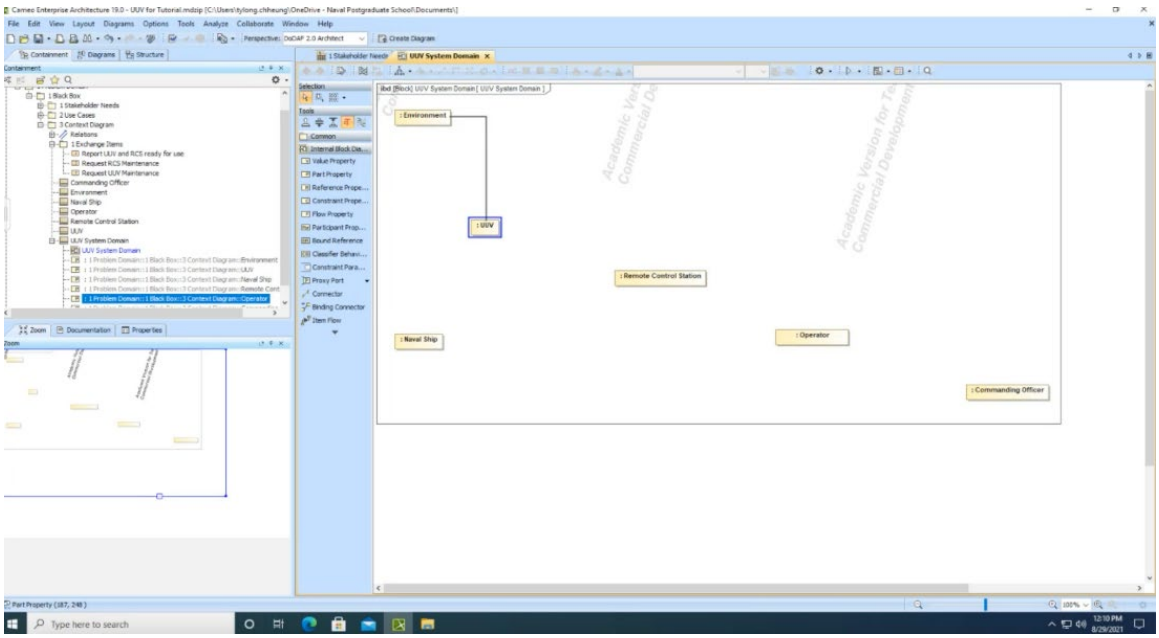
As mentioned in the initial phase of the system context, the system context illustrates an overall view of the system. All parts, whether having direct interaction with the system or not, are defined. In the final phase of the system context, the interactions are specified using connectors with item flows between the different parts. The item flows can be physical, informational, or controlling and are used to describe the nature of the interaction between the parts.

Step-By-Step Guide:

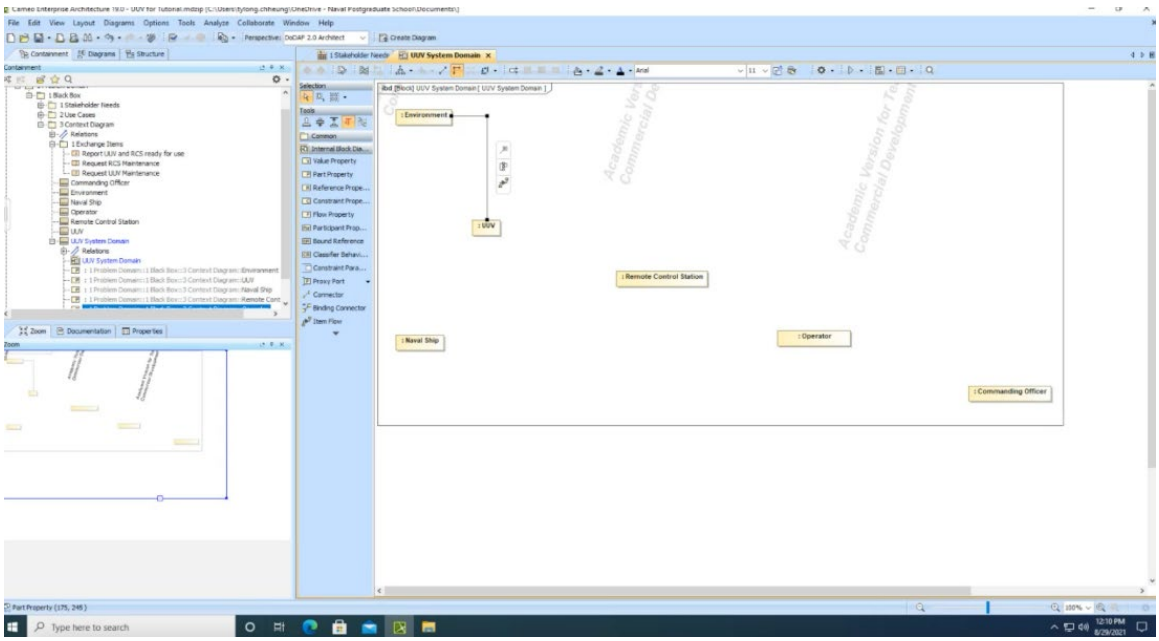
In this video we will cover how to complete the final system context diagram. Inside the IBD in the *UUV System Domain*, we will be selecting the *Environment* domain. From the smart manipulator tool bar, we will be selecting the *Connector* option.

Then we will drag the *Connector* and hover over the *UUV* domain, which causes the *UUV* domain to have a blue highlighted around it. And then click to select the *UUV* domain.

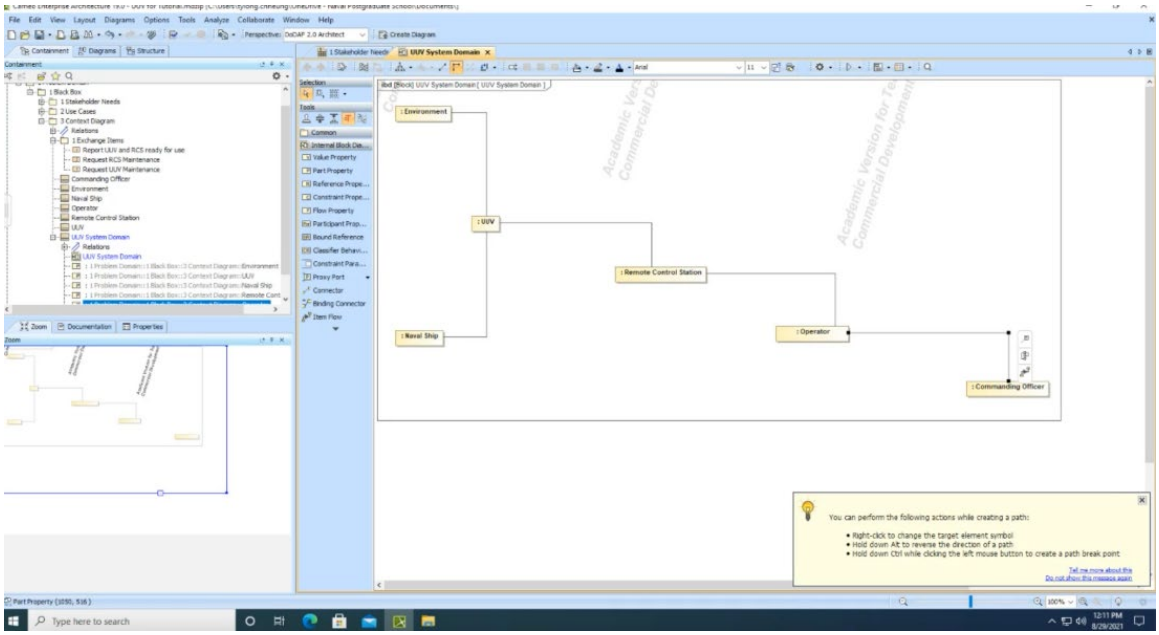




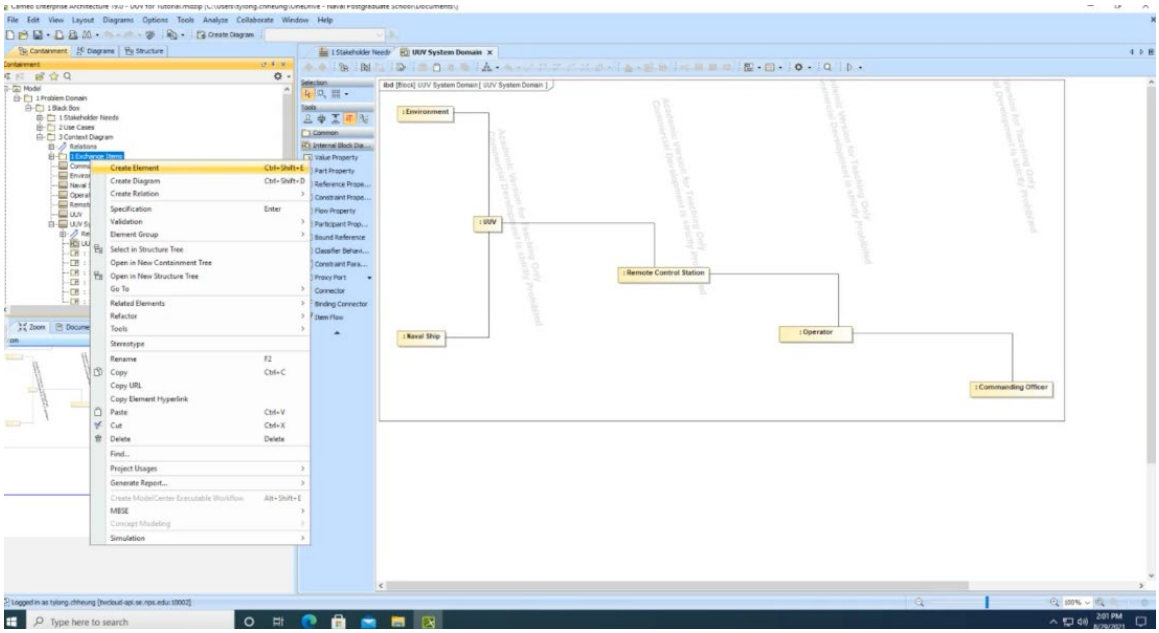
This will create the connection between the *Environment* domain and the *UUV* domain.



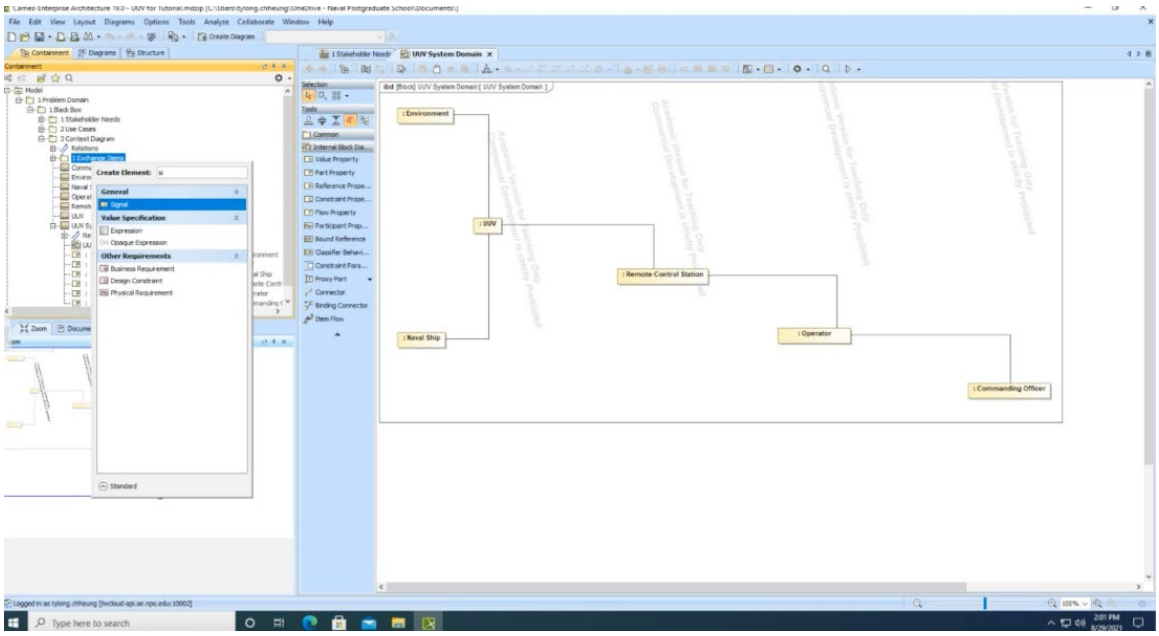
Repeat the same steps for the *UUV* domain and the *Naval Ship* domain. And then the *UUV* property to the *Remote Control Station* domain. Then the *Remote Control Station* domain to the *Operator* domain. Then finally the *Operator* domain to the *Commanding Officer* domain. Once all the domains are connected, it should look like the photo below.



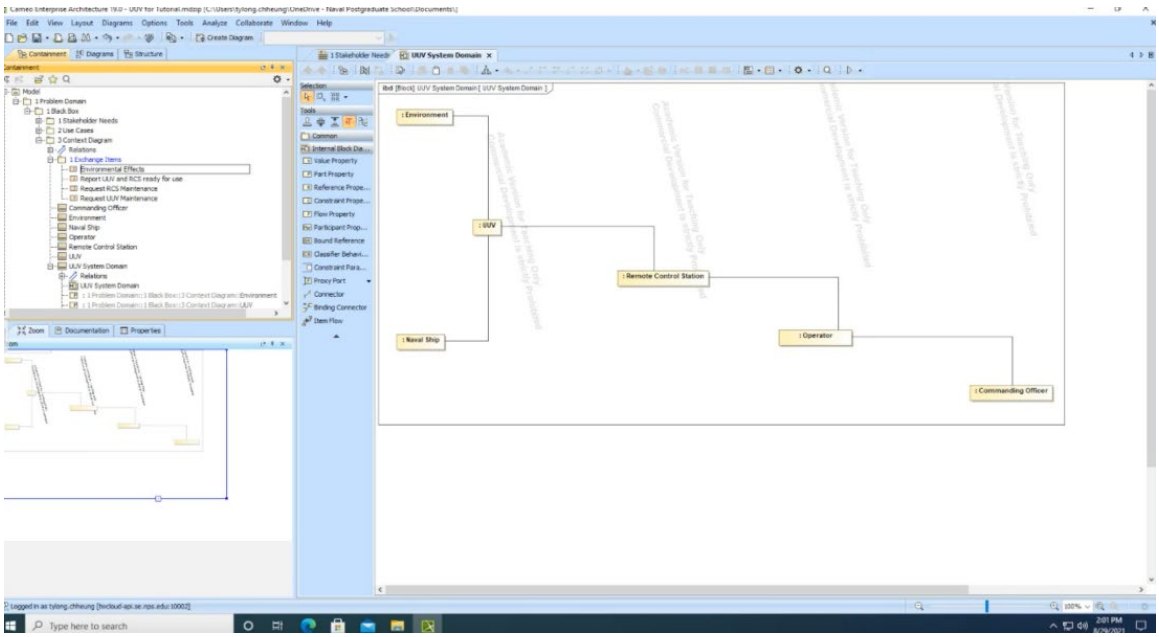
Once you have established all these connections, we will have to manage the flow of items between each connections. The *Environment* to the *UUV* to the *Naval Ship* and etc. Once the connections between each domain ire established, we will have to assign items between each participant of the system context. We start this by selecting the *Exchange Item* under the context diagram and right click to bring the drop-down menu. From there we will select *Create Element*.



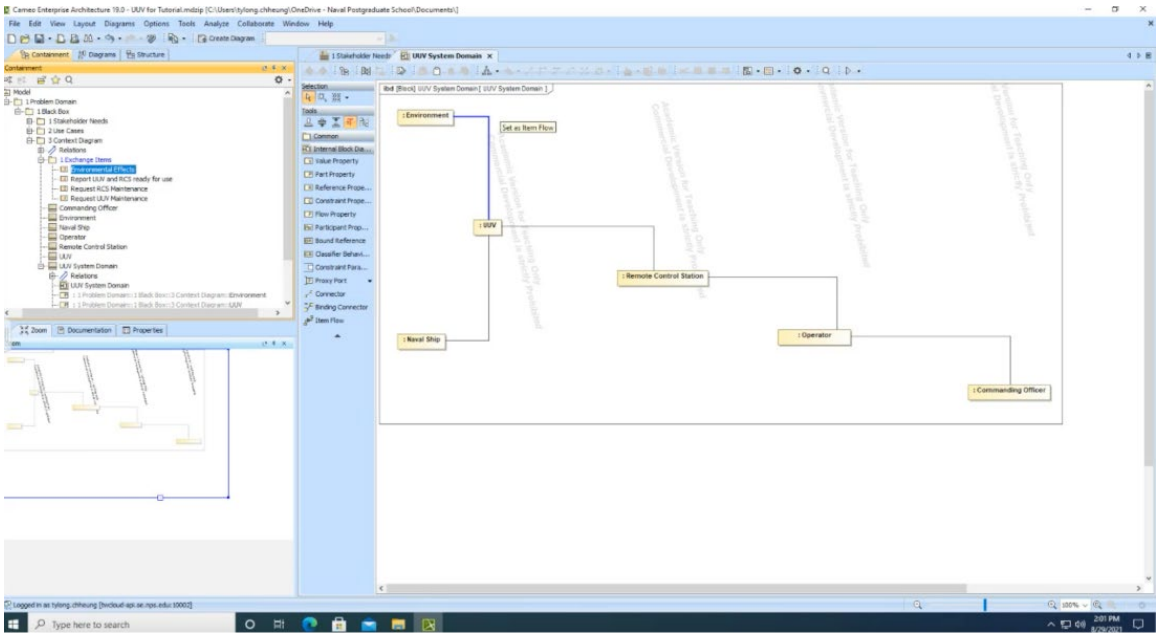
This will then open a pop-up menu, where we will be typing *si* to bring up the *signal*.



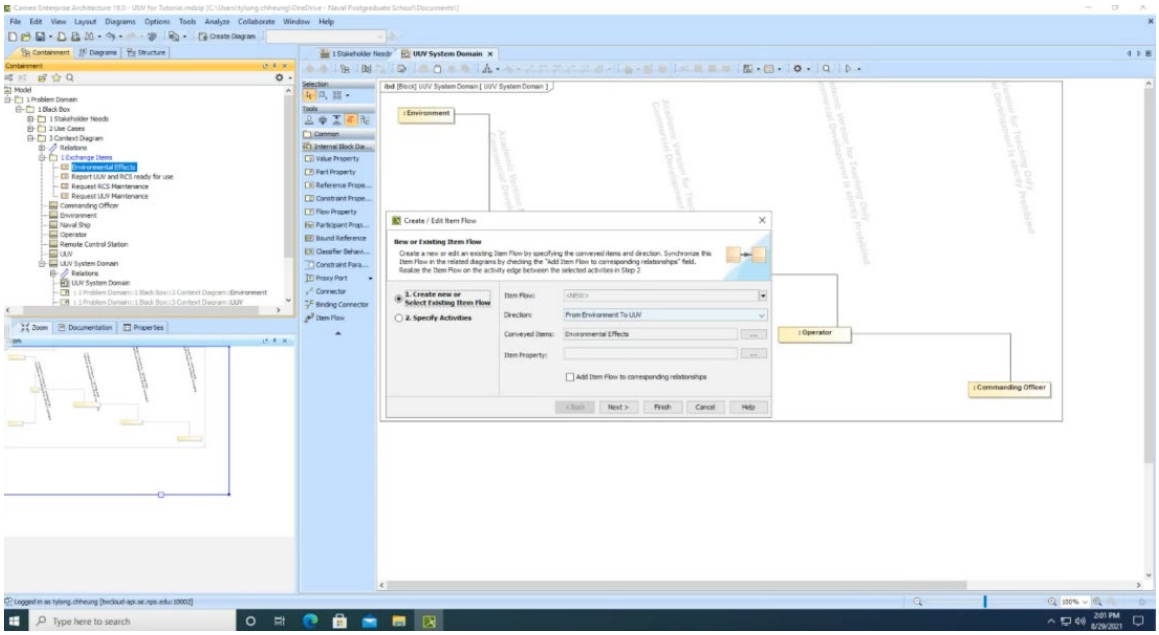
Then for this signal we will type *Environmental Effects* in the text field.



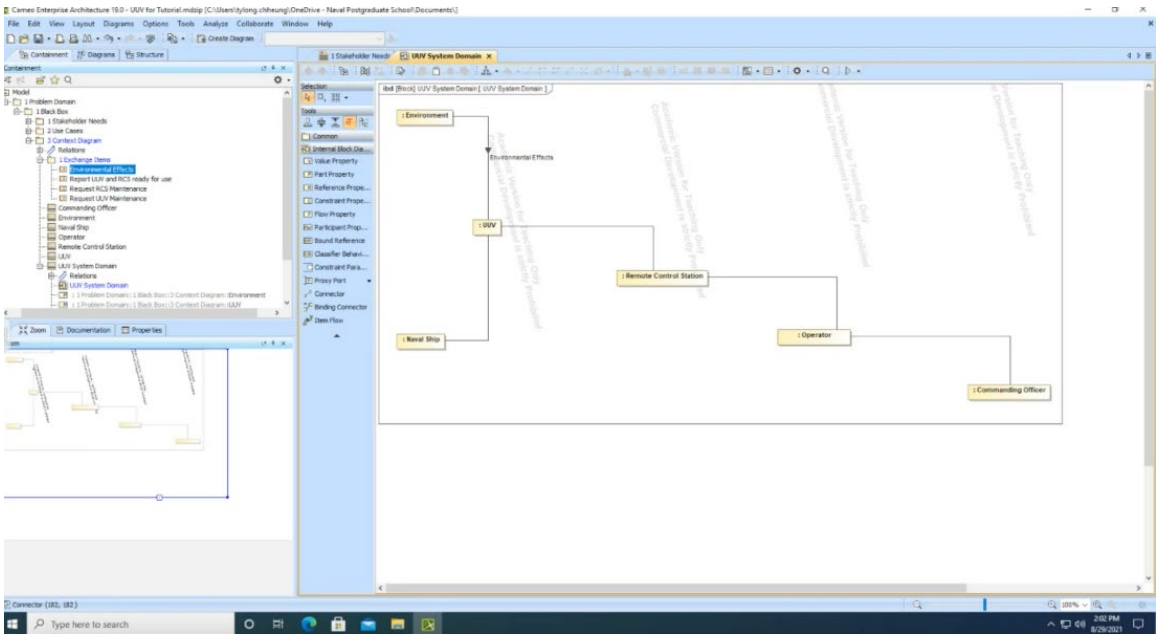
Click on *Environmental Effects* signal and drag to the connection between the *Environment* domain and the *UUV* domain.



Then when you click on that connection, the *Create/Edit Item Flow* popup window appears. From here we can see that the *Direction* shows *Environment To UUV*. Then we will click the finish button.

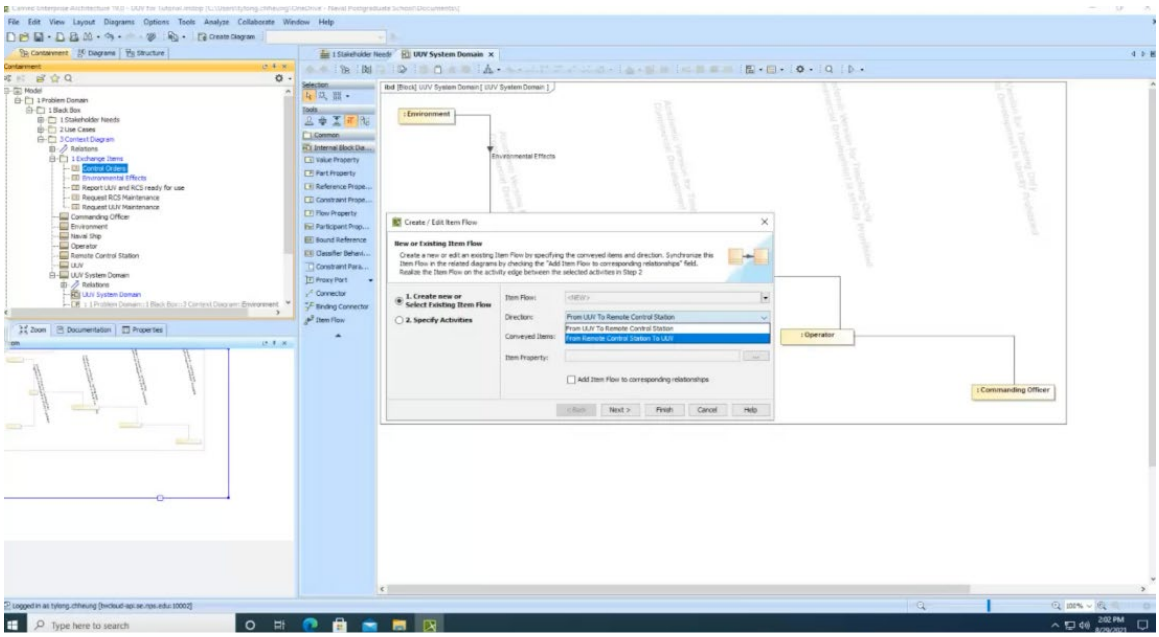


That will then establish the first item *Environmental Effects* signal that flows from the *Environment* domain to *UUV* domain.

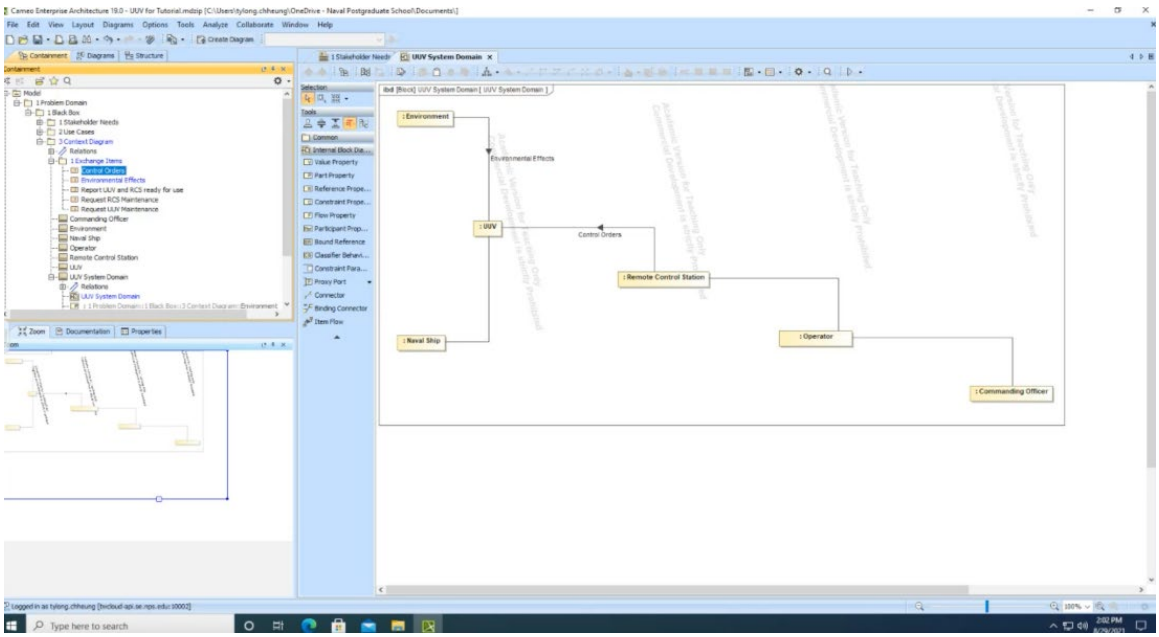


We will then be repeating this process by right clicking on the *Exchange Item* to create the second signal, which will be *Control Orders*. The *Control Orders* will be from the *Remote Control Station* domain to the *UUV* domain. Therefore, we will drag the signal and click on the connection between the *UUV* domain to the *Remote Control Station*

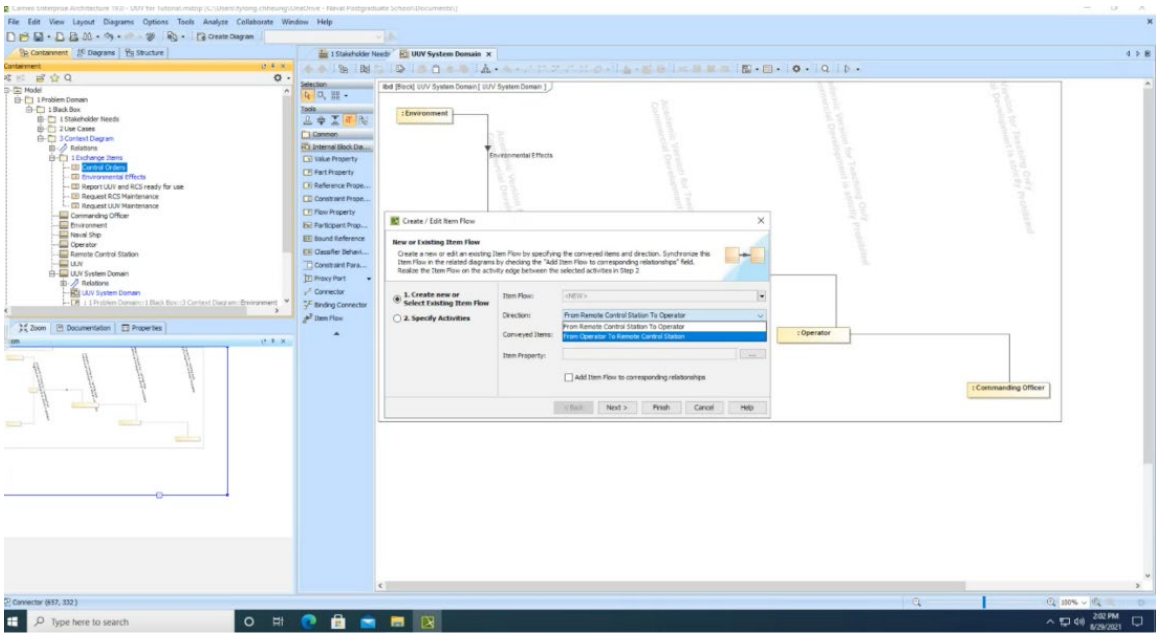
domain, which will then bring up the *Create/Edit Item Flow* popup window. Therefore, we will go to the *Direction* section and from the drop-down menu we will change to *From Remote Control Station To UUV*.



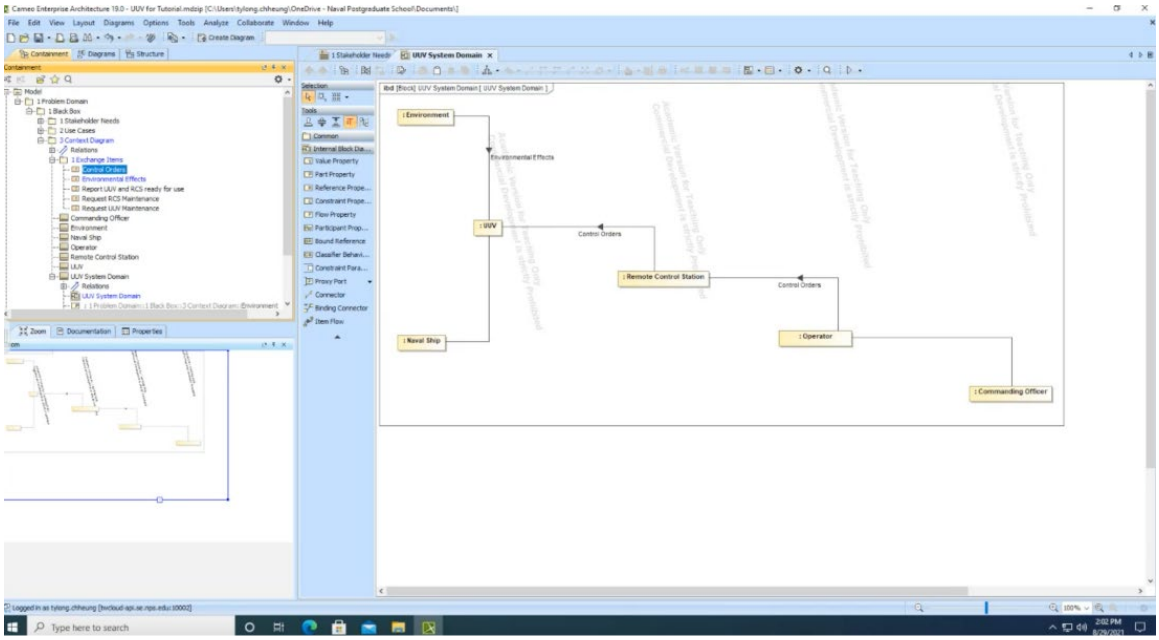
We will then close the *Create/Edit Item Flow* popup window by clicking on the finish button. The new signal will appear between *UUV* domain and the *Remote Control Station* domain as seen on the photo below.



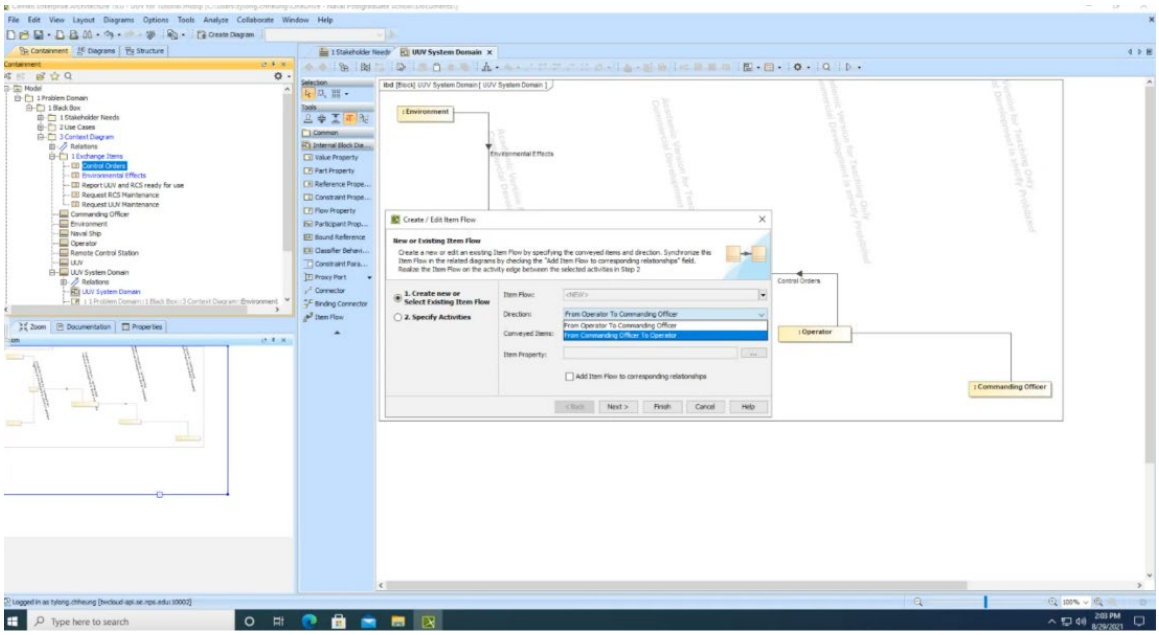
Then we will then again select *Control Orders* and drag to the connection between the *Remote Control Station* domain and the *Operator* domain. This will then bring up the *Create/Edit Item Flow* popup window and we will be selecting *From Operator To Remote Control Station* from the *Direction* drop-down menu.



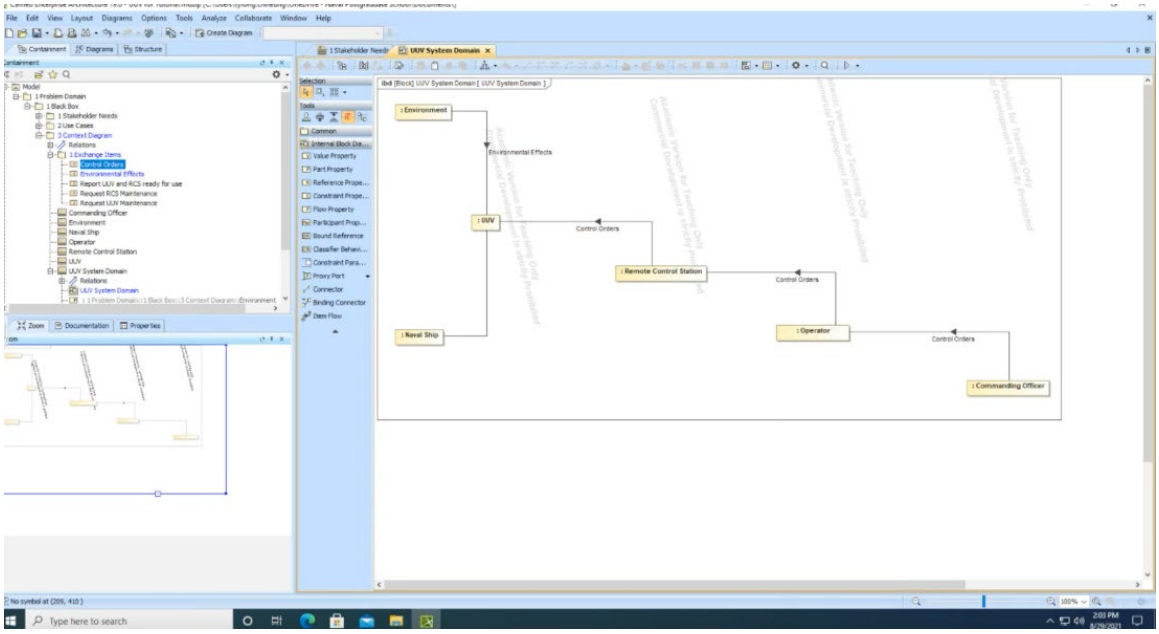
This will show the *Control Orders* signal flowing from the *Operator* domain to the *Remote Control Station* domain.



Once again, we will be selecting *Control Orders* and drag to the connection between the *Operator* domain and the *Commanding Officer* domain. This will then bring up the *Create/Edit Item Flow* popup window and we will be selecting *From Commanding Officer To Operator* from the *Direction* drop-down menu.

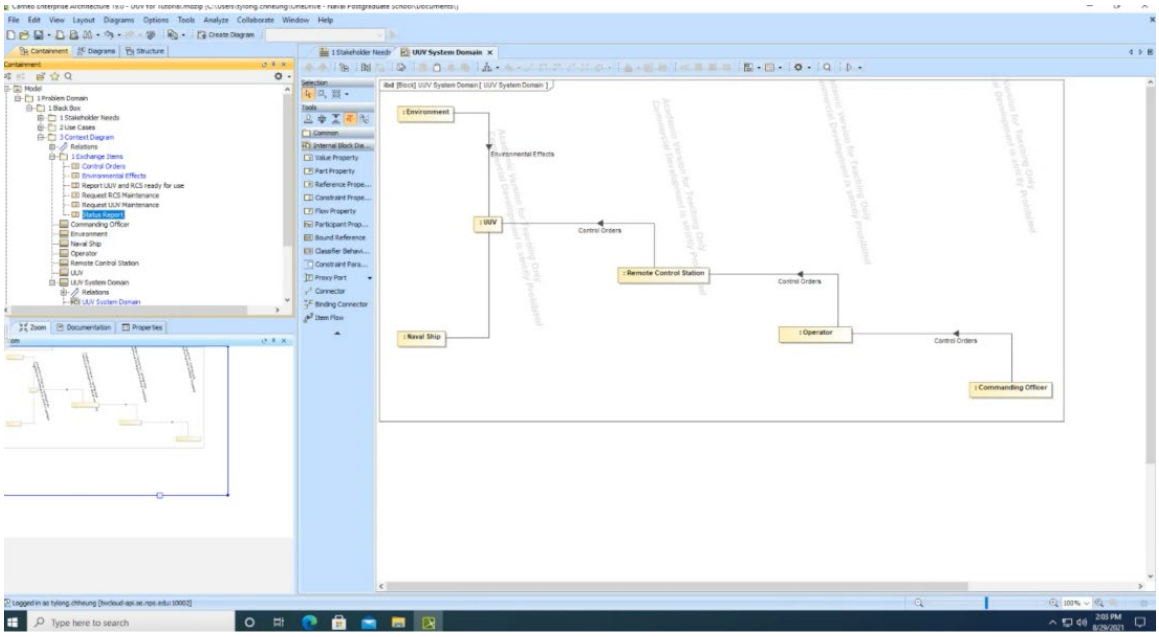


This will show the *Control Orders* signal flowing from the *Commanding Officer* domain to the *Operator* domain.

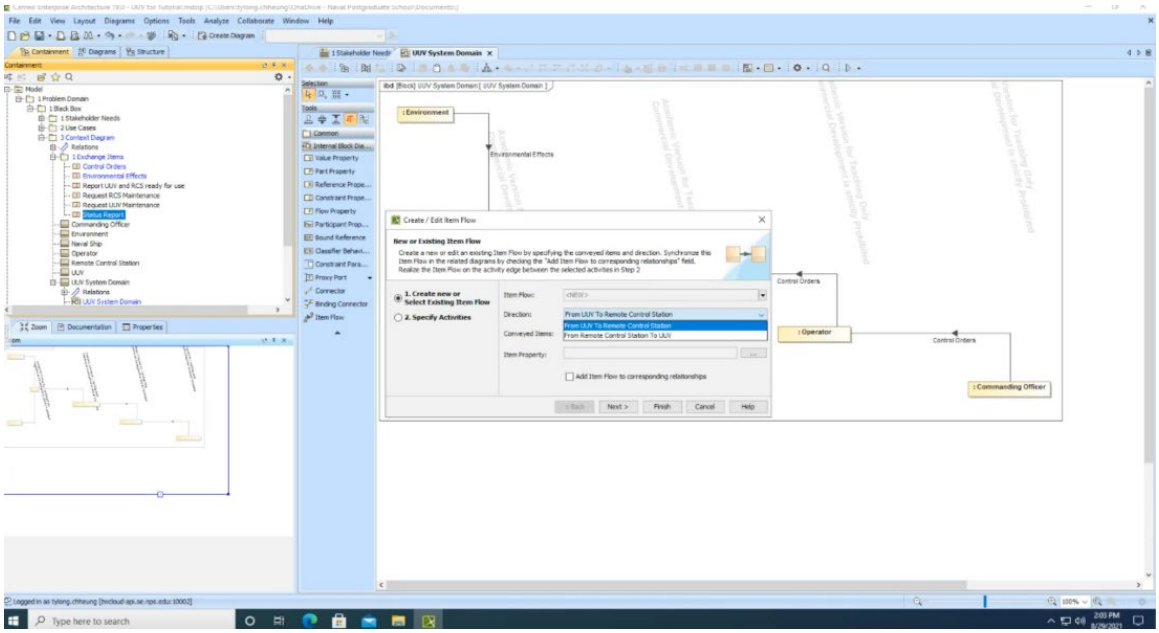


The next signal we will be creating will be *Status Report*. We will be selecting the *Exchange Item* under the context diagram and right click to bring the drop-down menu. From there we will select *Create Element*. The *Create Element* pop-up menu will appear

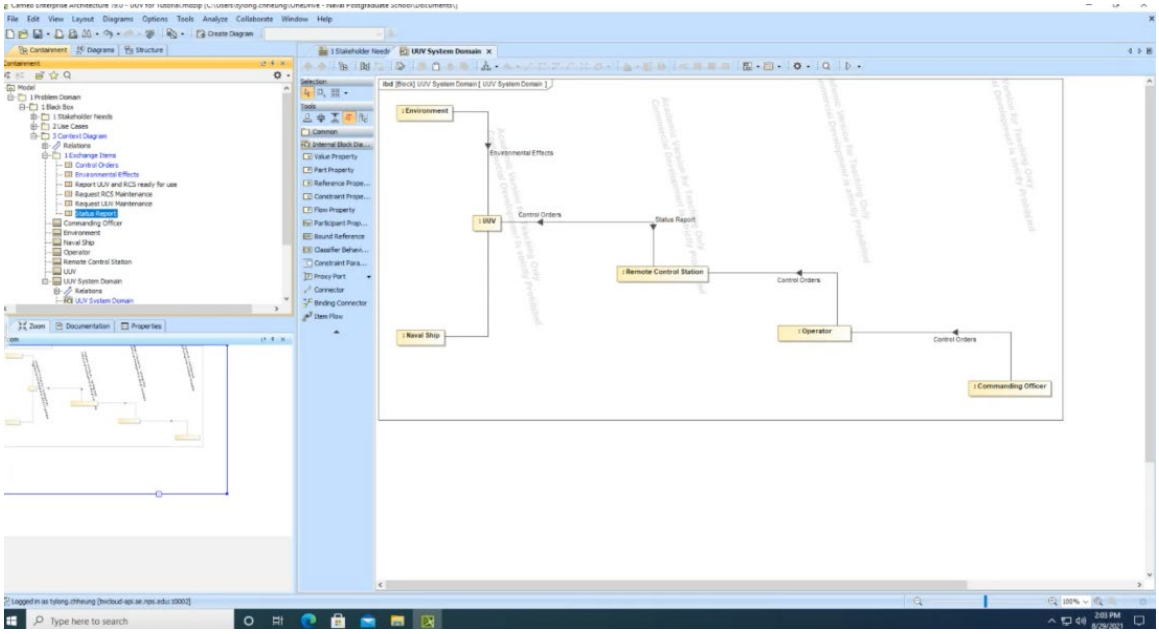
and we will be typing *si* to bring up the *signal*. Then for this signal we will type *Status Report* in the text field.



We will select the *Status Report* signal and dragging it to the connection between *UUV* domain and the *Remote Control Station* domain. The *Create/Edit Item Flow* popup window will appear and we will be selecting *From UUV To Remote Control Station* from the *Direction* drop-down menu.

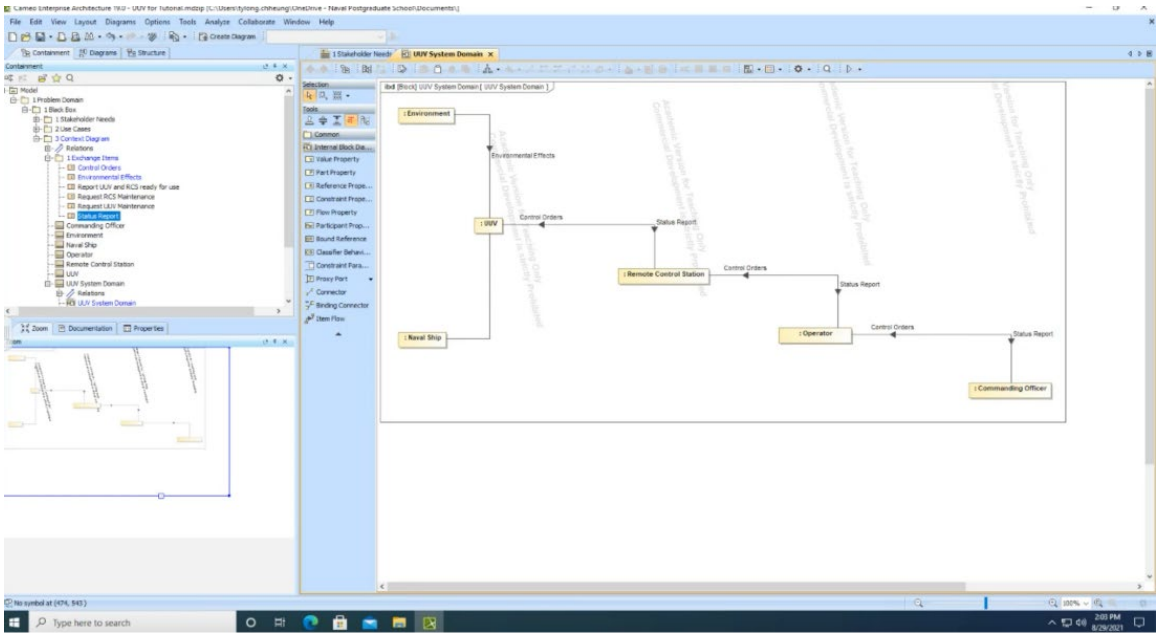


After we close the *Create/Edit Item Flow* popup window, the signal will be shown flowing from the *UUV* domain and the *Remote Control Station* domain.

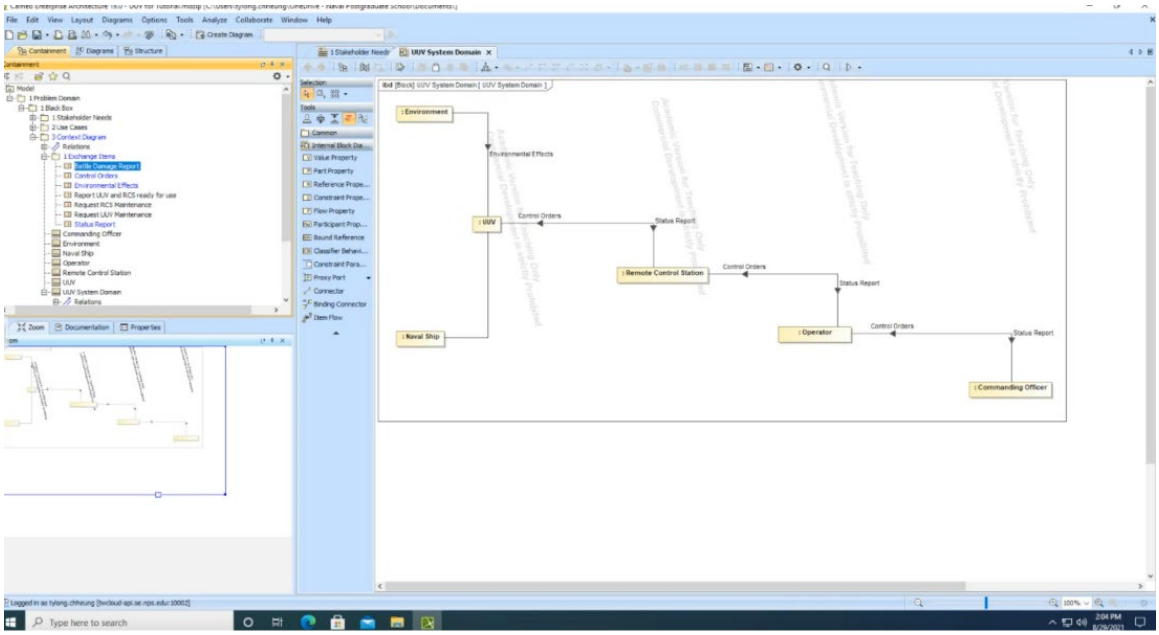


Then we will be repeating the process over and over again to show the *Status Report* signal flowing from the *Remote Control Station* domain to the *Operator* domain. Also

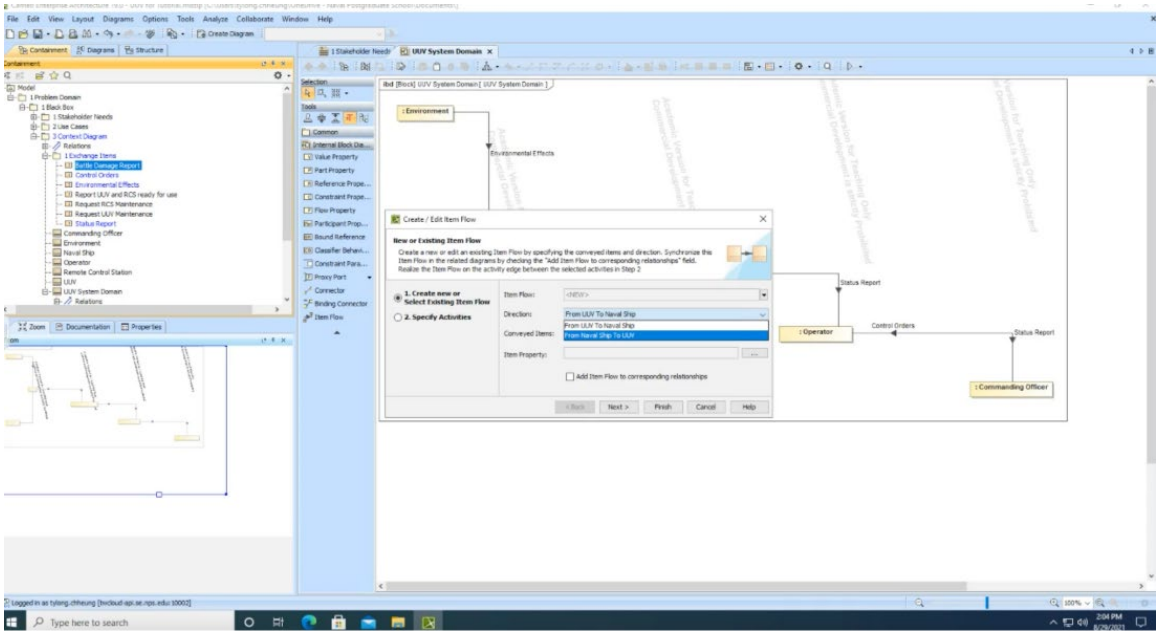
Status Report signal flowing from the *Operator* to the *Commanding Officer* domain. The photo below show all the *Status Report* signals and their proper flow direction.



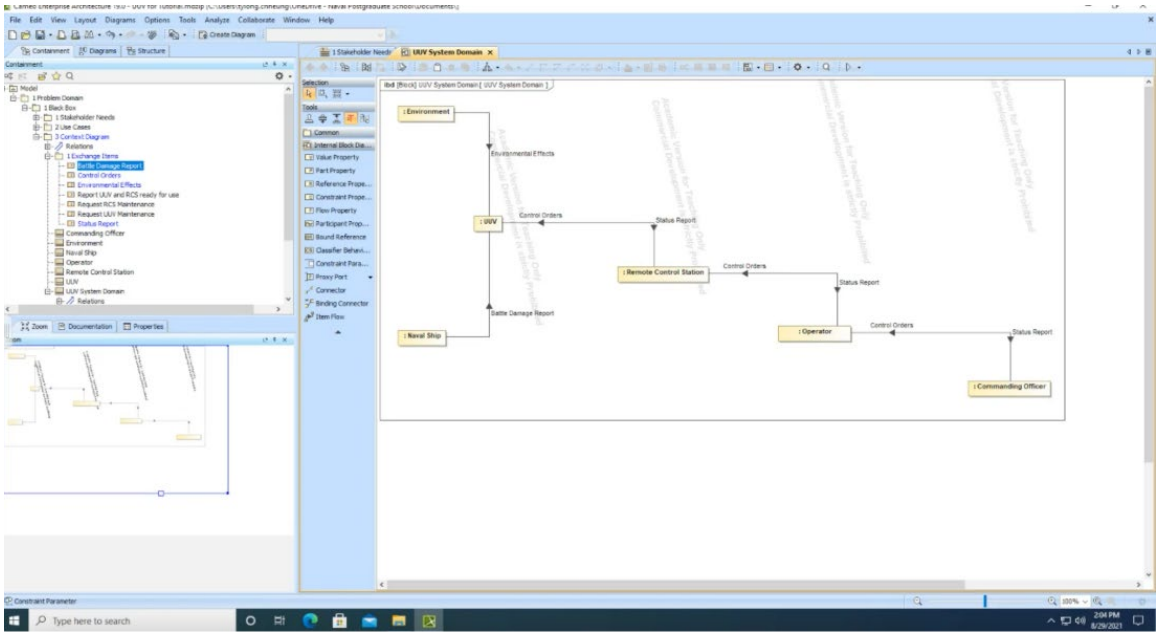
The last two signals we will be creating for the system context diagram will be *Battle Damage Report*. This signal will flow from the *Naval Ship* domain to the *UUV* domain. Once again we will be selecting the *Exchange Item* under the context diagram and right click to bring the drop-down menu. From there we will select *Create Element*. The *Create Element* pop-up menu will appear and we will be typing *si* to bring up the *signal*. Then for this signal we will type *Battle Damage Report* in the text field.



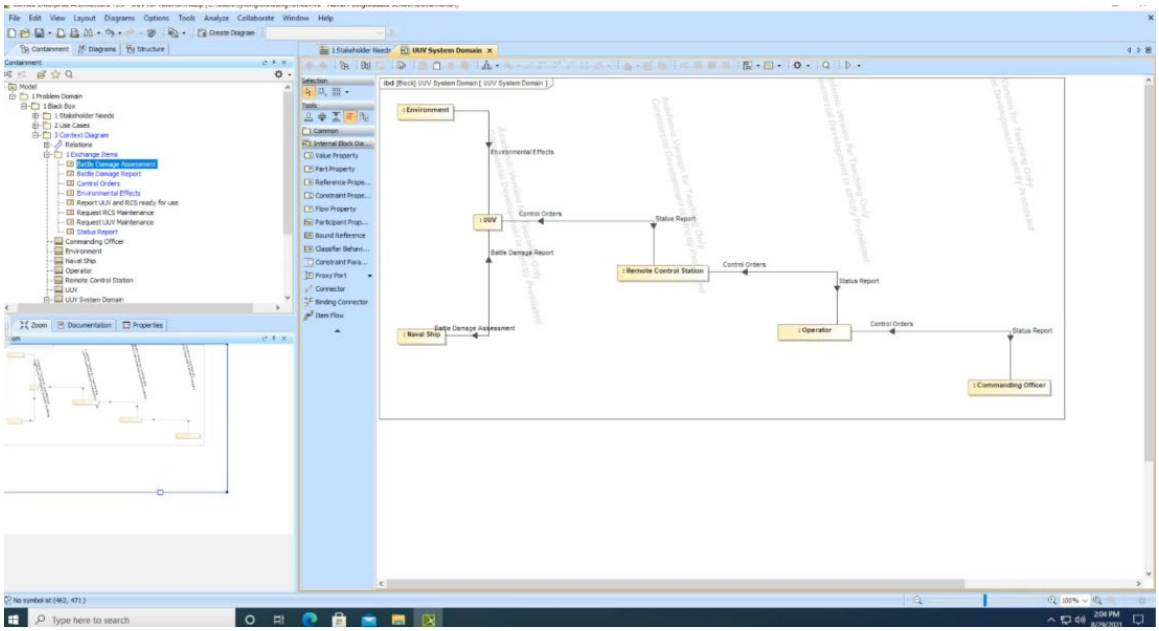
We will select the *Battle Damage Report* signal and dragging it to the connection between *UUV* domain and the *Naval Ship* domain. The *Create/Edit Item Flow* popup window will appear and we will be selecting *From Naval Ship To UUV* from the *Direction* drop-down menu.



After we close the *Create/Edit Item Flow* popup window, the *Battle Damage Report* signal will be shown flowing from the *Naval Ship* domain and the *UUV* domain.



Now the last signal we will create will be *Battle Damage Assessment* and the signal will flow from *UUV* domain to the *Naval Ship* domain. We will be selecting the *Exchange Item* under the context diagram and right click to bring the drop-down menu. From there we will select *Create Element*. The *Create Element* pop-up menu will appear and we will be typing *si* to bring up the *signal*. Then for this signal we will type *Battle Damage Assessment* in the text field. We will select the *Battle Damage Assessment* signal and dragging it to the connection between *UUV* domain and the *Naval Ship* domain. The *Create/Edit Item Flow* popup window will appear and we will be selecting *From UUV To Naval Ship* from the *Direction* drop-down menu. After we close the *Create/Edit Item Flow* popup window, the *Battle Damage Assessment* signal will be shown flowing from the *UUV* domain to the *Naval Ship* domain. Once you have that completed, the photo below shows the final system context diagram.



This concludes the tutorial on the final system context. Thank you.

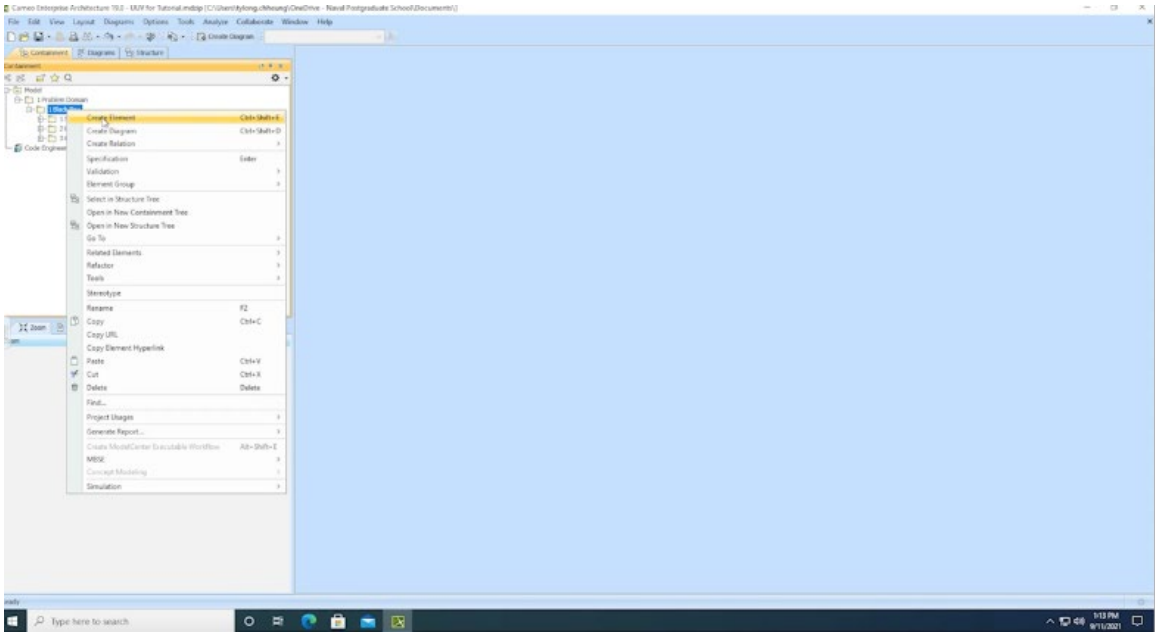
B4: Measures of Effectiveness Tutorial

Background:

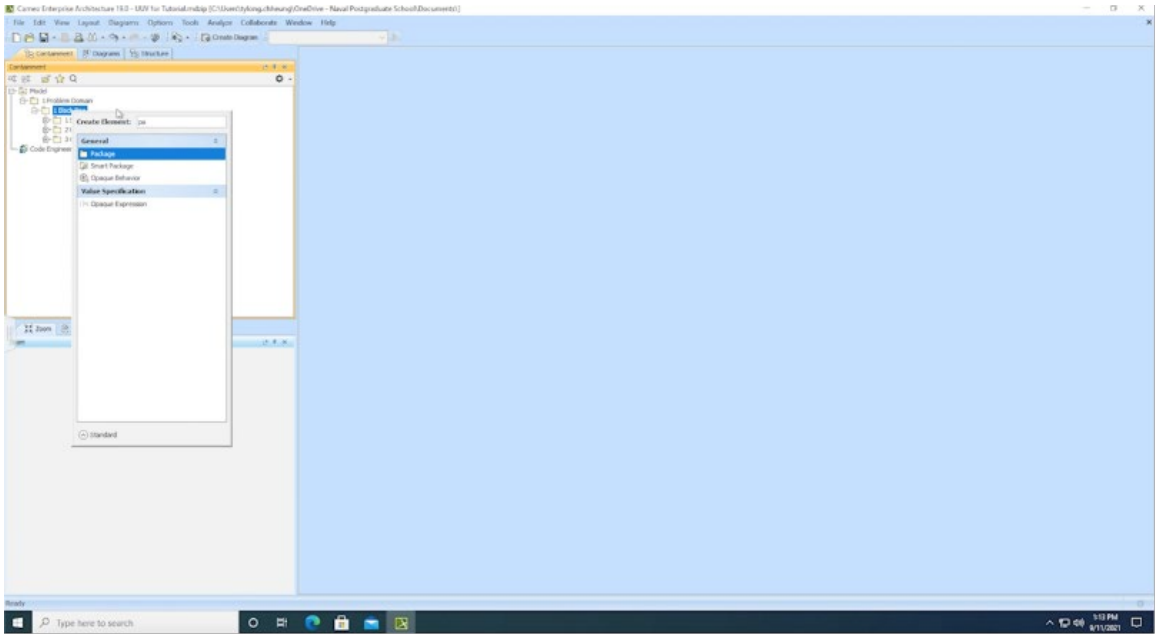
The measurements of effectiveness (MoEs) redefine the non-functional stakeholder needs that were established. MoEs describe how well a system carries out within the system context. In the problem domain, they serve as high level key performance indicators, such as total power or total mass. The MoEs are captured using a block definition diagram (bdd), where blocks are created and set as a super-type for the system of interest. Value properties or stereotypes are applied, which can be passed to subsequent blocks or components that the system may have.

Step-By-Step Guide:

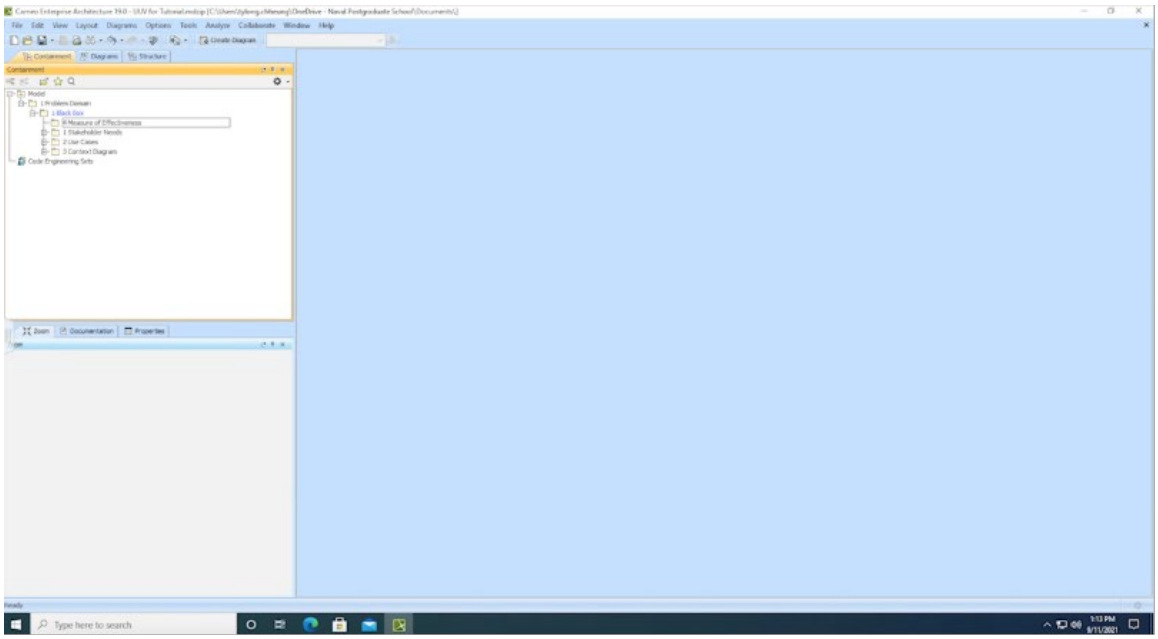
In this video, we will cover how the *Measure Of Effectiveness (MOE)* for the UUV. First, we start off by right clicking *1 Black Box* on the menu on the left-hand side. From the drop-down menu, then we will choose *Create Element*.



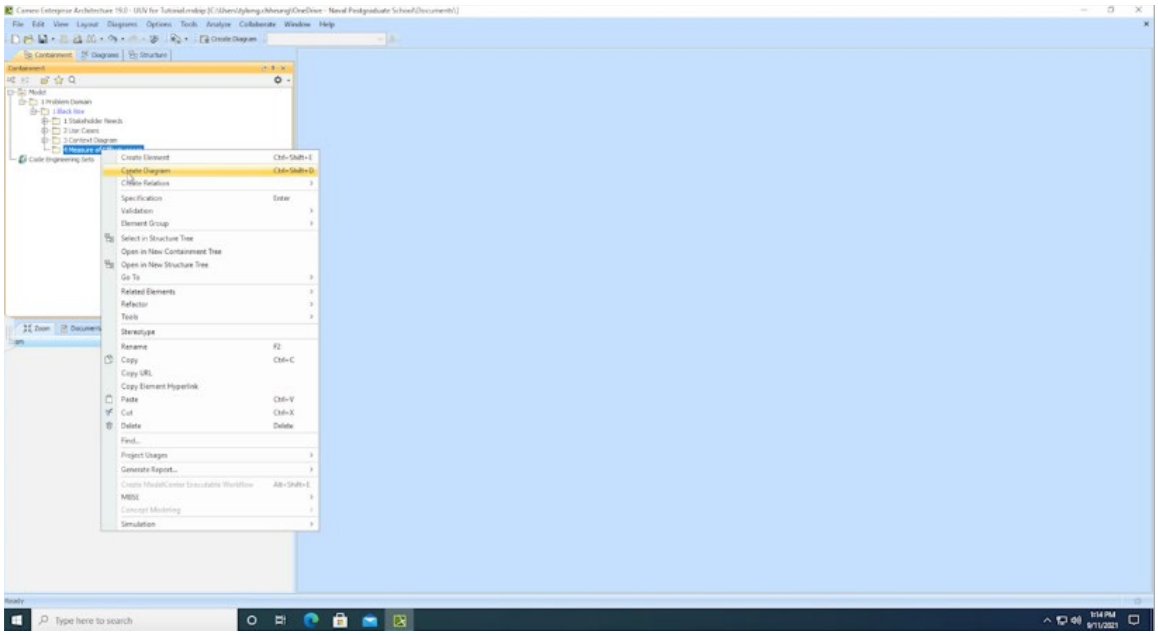
This will bring up the *Create Element* popup window. Then from there we will start to type out *Pa* and select *Package* from the *Create Element* popup window. Then hit the enter key on the keyboard.



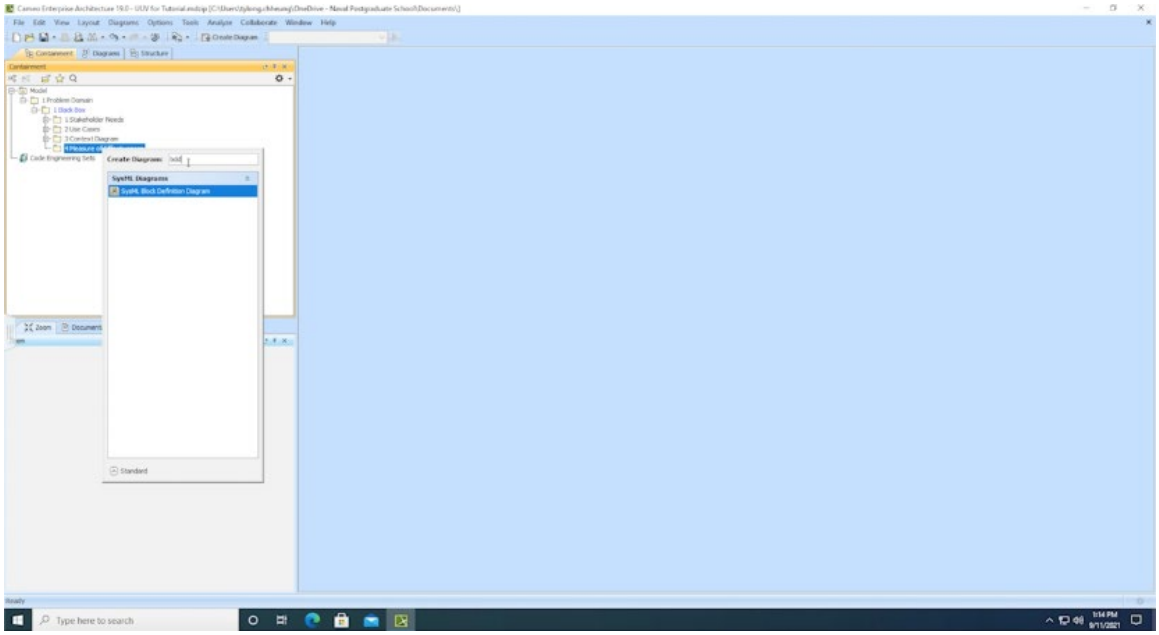
The we will start to type the label for the package which will be *4Measure Of Effectiveness*. Then we click the enter key on the keyboard to finish creating the package.



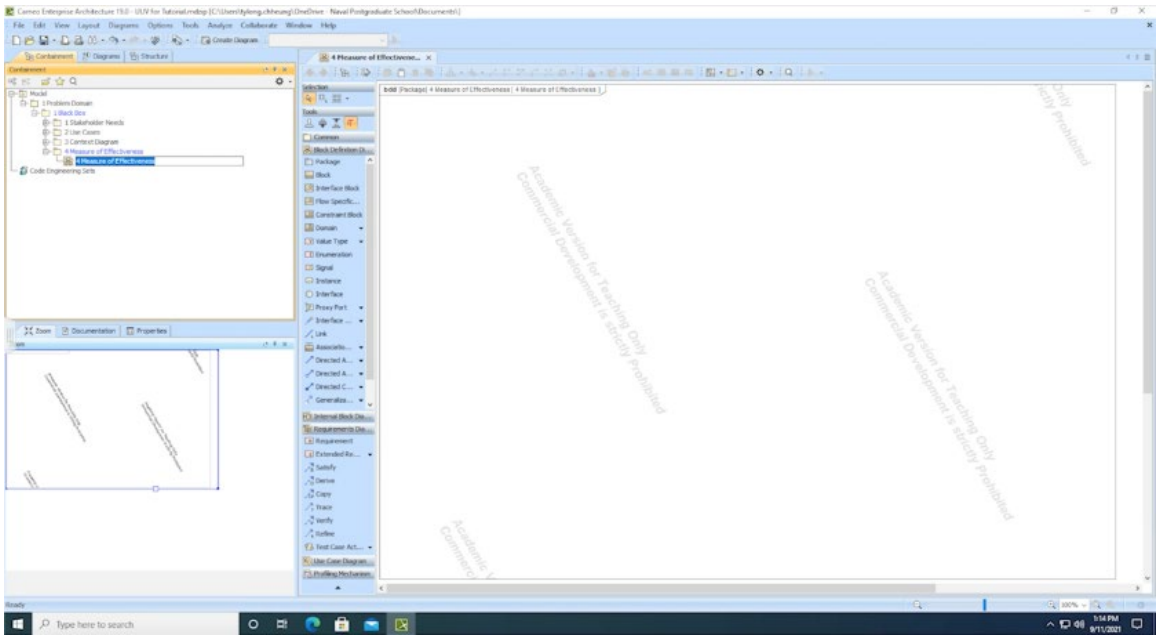
Once the package is created, we will right click on the newly created package to bring up the drop-down menu where we will select *Create Diagram*.



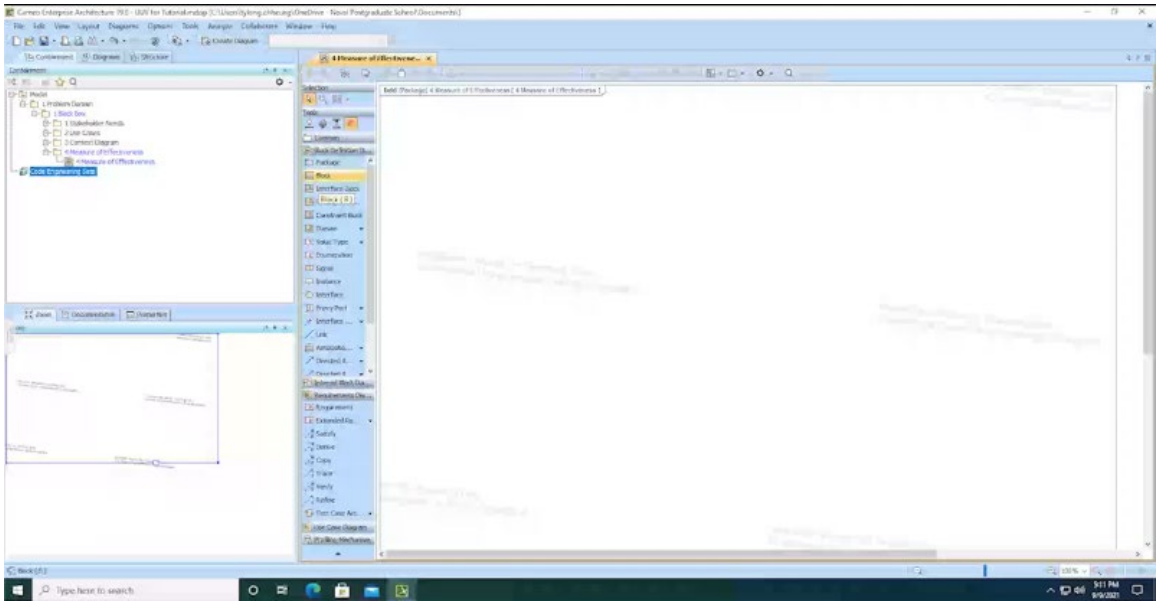
After selecting the *Create Diagram* option the popup window for *Create Diagram* will appear. From the search bar, we will start to type *BDD* for the *SysML Block Definition Diagram*.



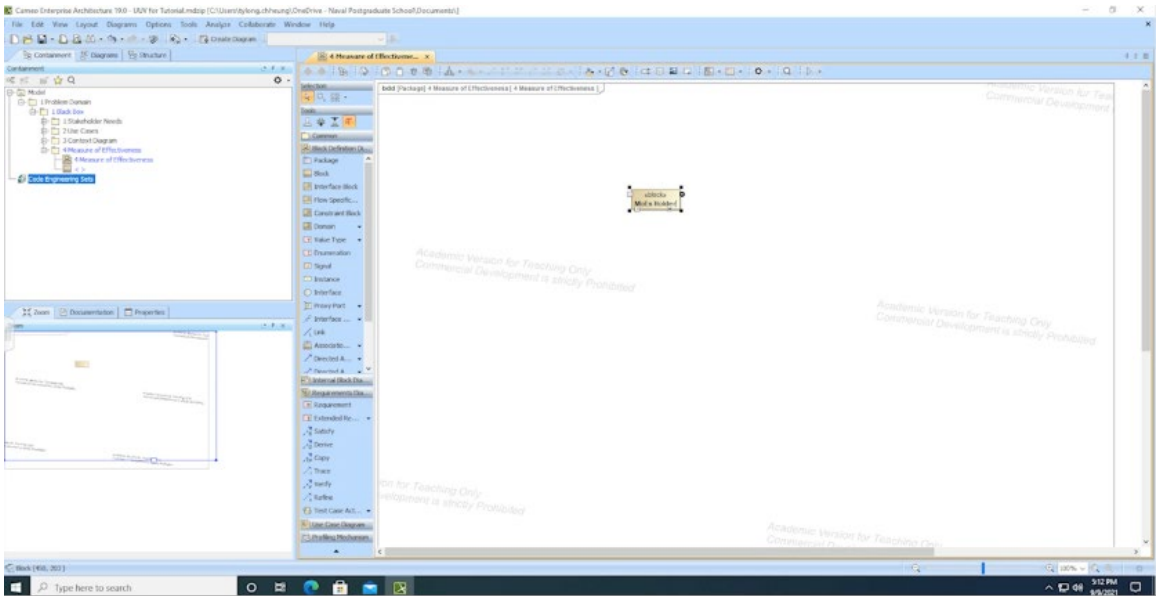
Once we select the *SysML Block Definition Diagram* option, the *Measure Of Effectiveness Block Definition Diagram* will be created.



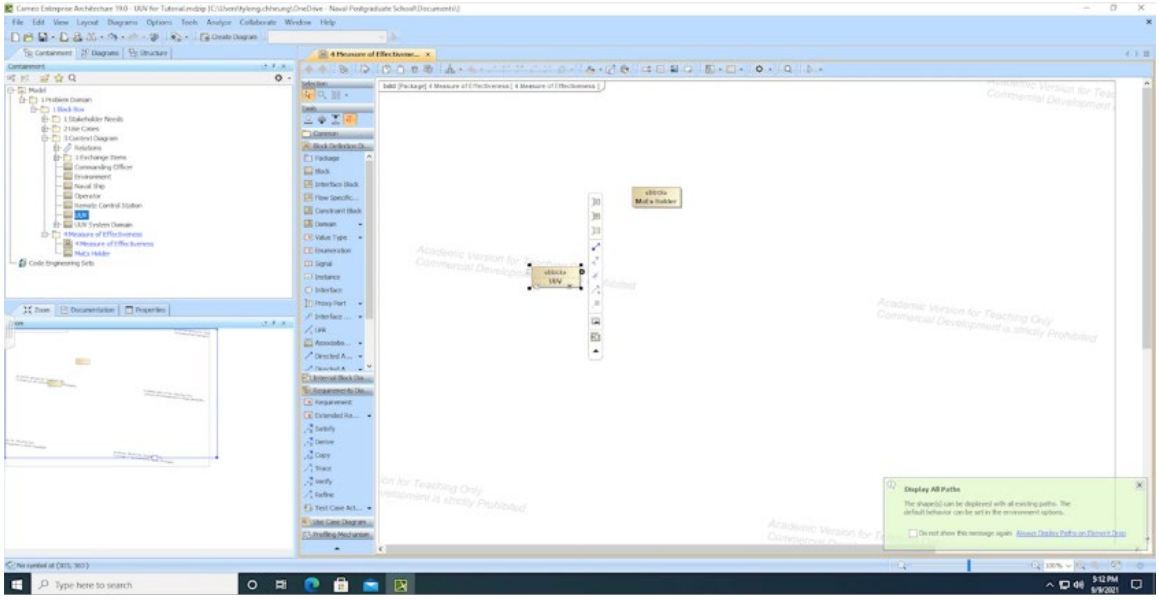
We will then click the *Block* button on the diagram tool bar menu.



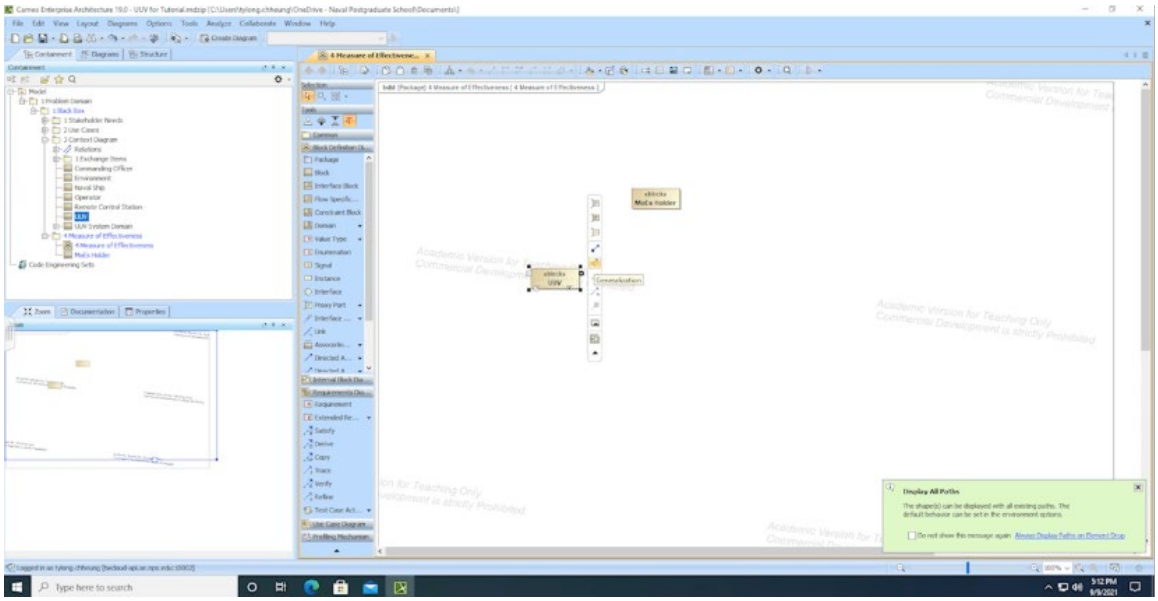
And then click on the empty space in the diagram plane to create an unnamed block on the plane, which we will label as *MOEs Holder*.



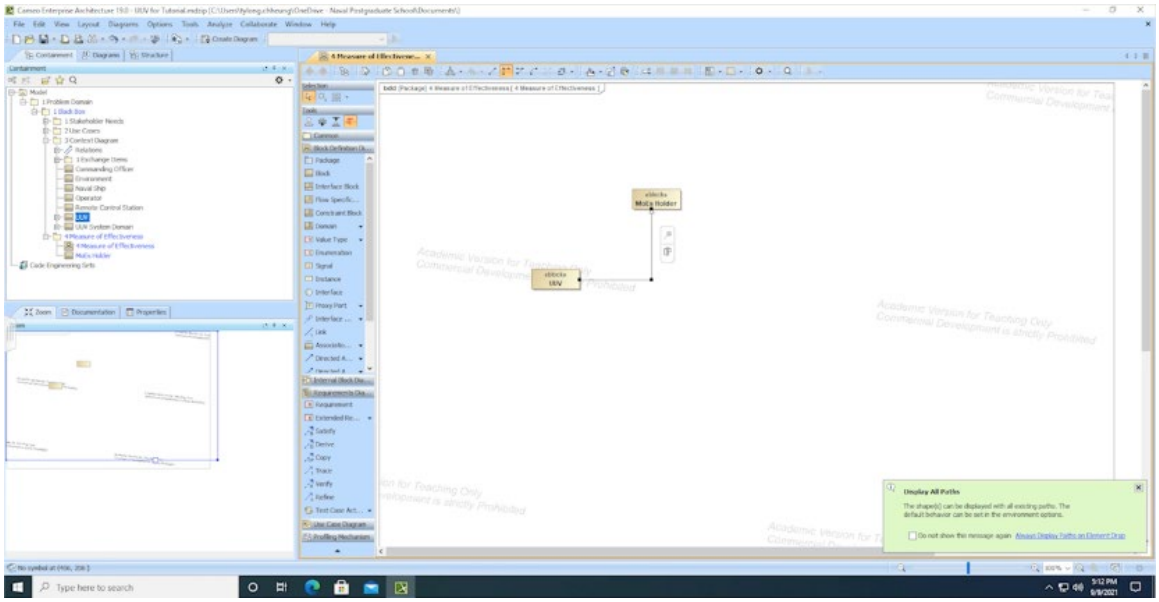
Then in the model menu on the left-hand side, we will select the *UUV* and drag it to the empty space of the diagram plane. This will create the block *UUV* on the diagram plane.



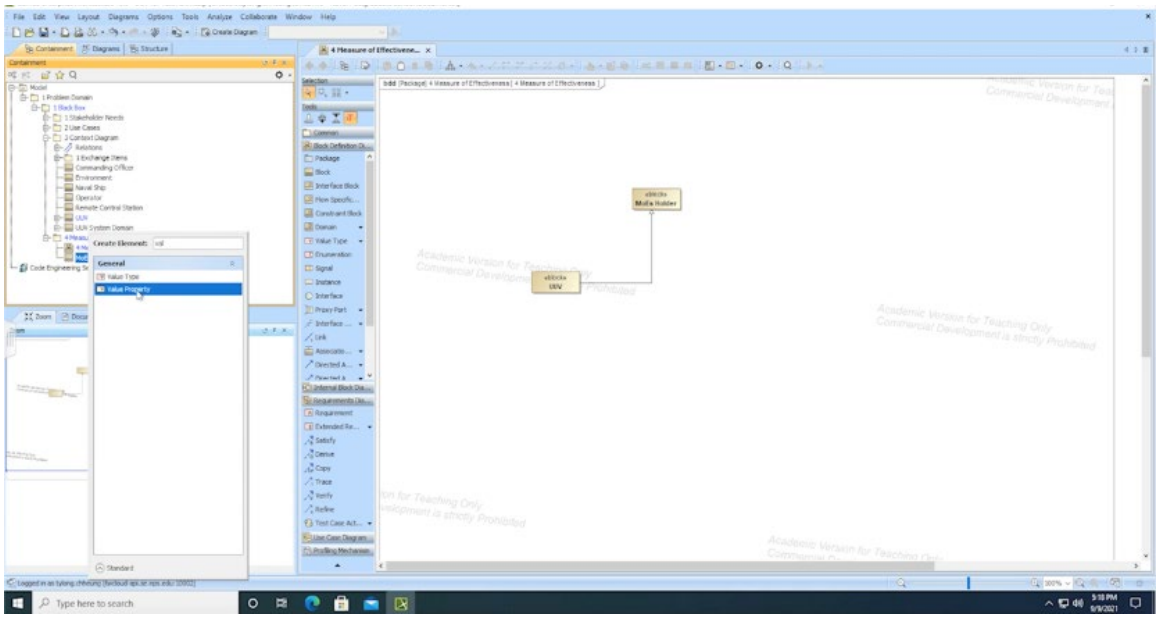
We will then from the manipulator tool bar, we will select *Generalization* button in order to generalize the *UUV* block to the *MOEs Holder* block.



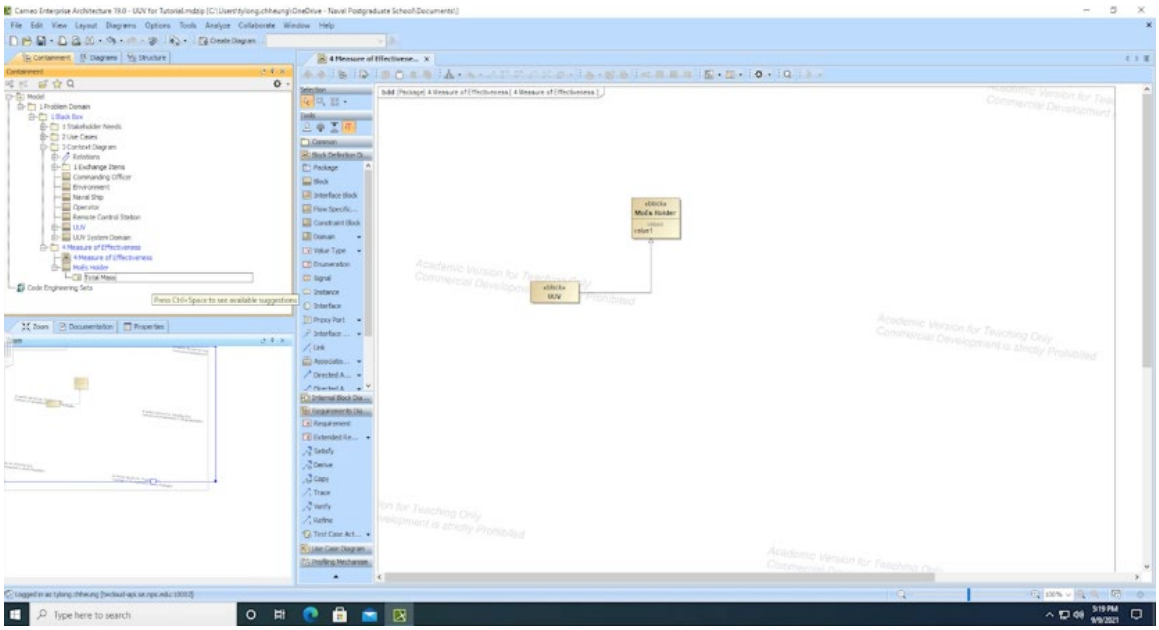
Once the connection is made the *MOE Holder* block becomes the supertype of the *UUV* block.



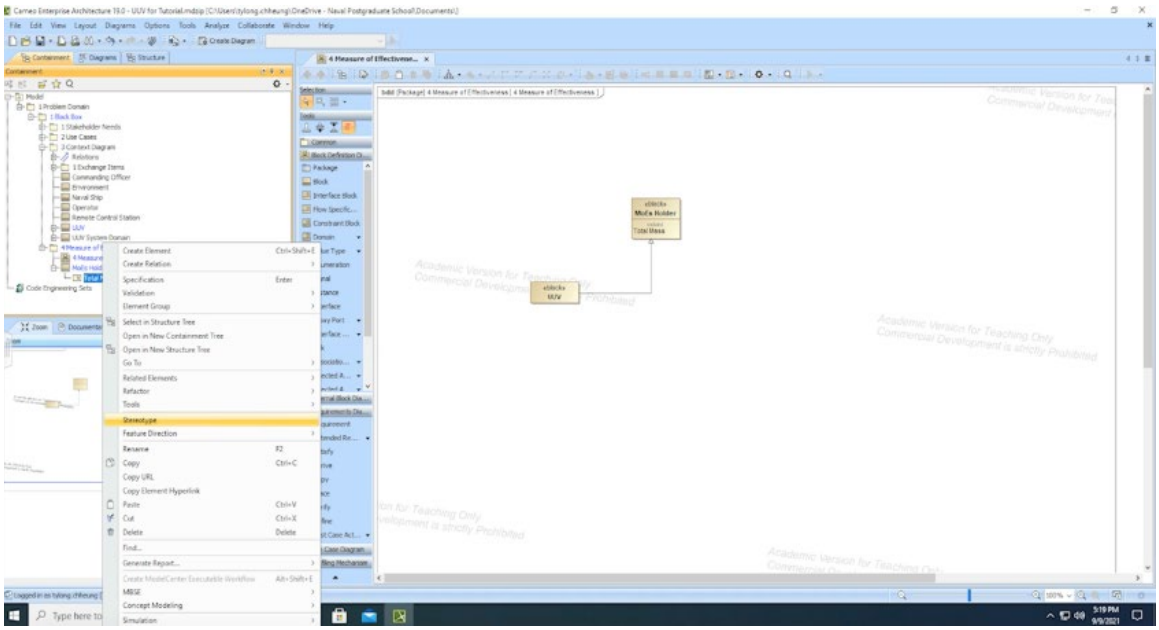
And from here we can select and right click the *MOE Holder* from the model browser menu on the left-hand side. From the drop-down menu, then we will choose *Create Element*. This will bring up the *Create Element* popup window. Then from there we will start to type out *Va* and select *Value Property* from the *Create Element* popup window. Then hit the enter key on the keyboard.



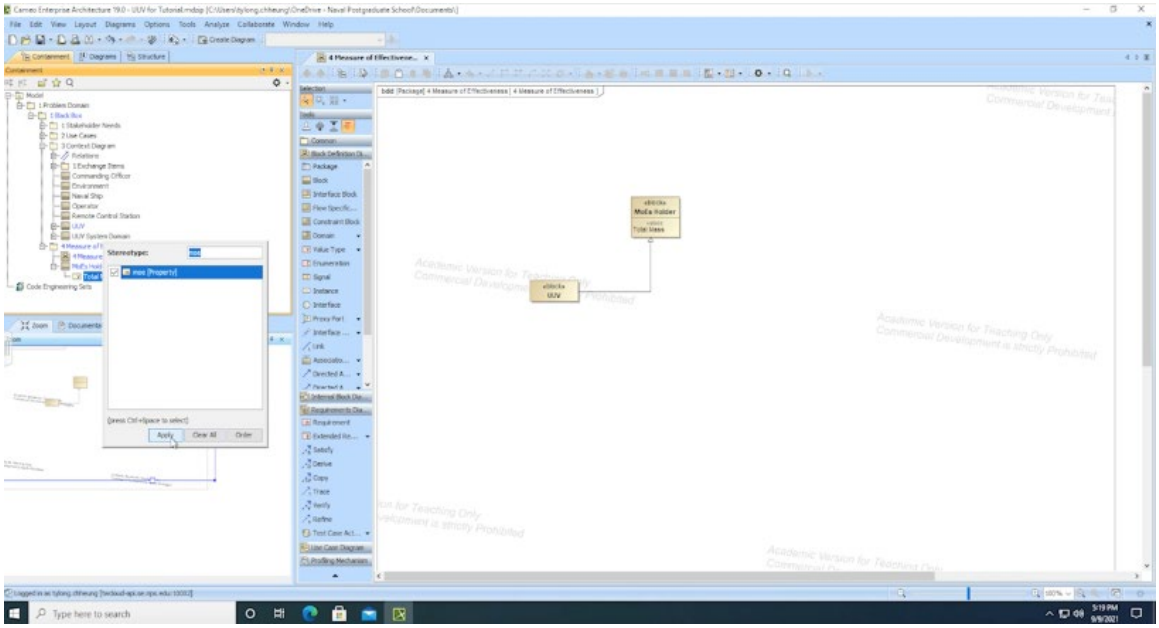
The we will start to type the label for the value property which will be *Total Mass*. Then we click the enter key on the keyboard to finish creating the package. The value property will be the total mass of the UUV.



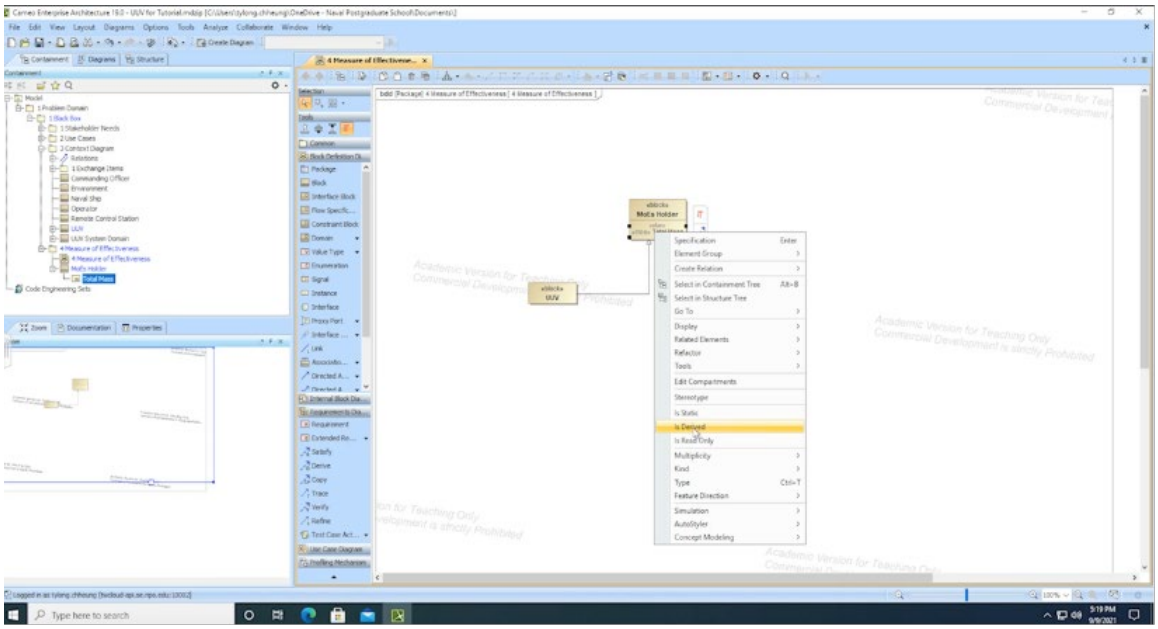
Once that is completed right click *Total Mass* value property from the model browser menu on the left-hand side. From the drop-down menu, then we will choose *Stereotype*.



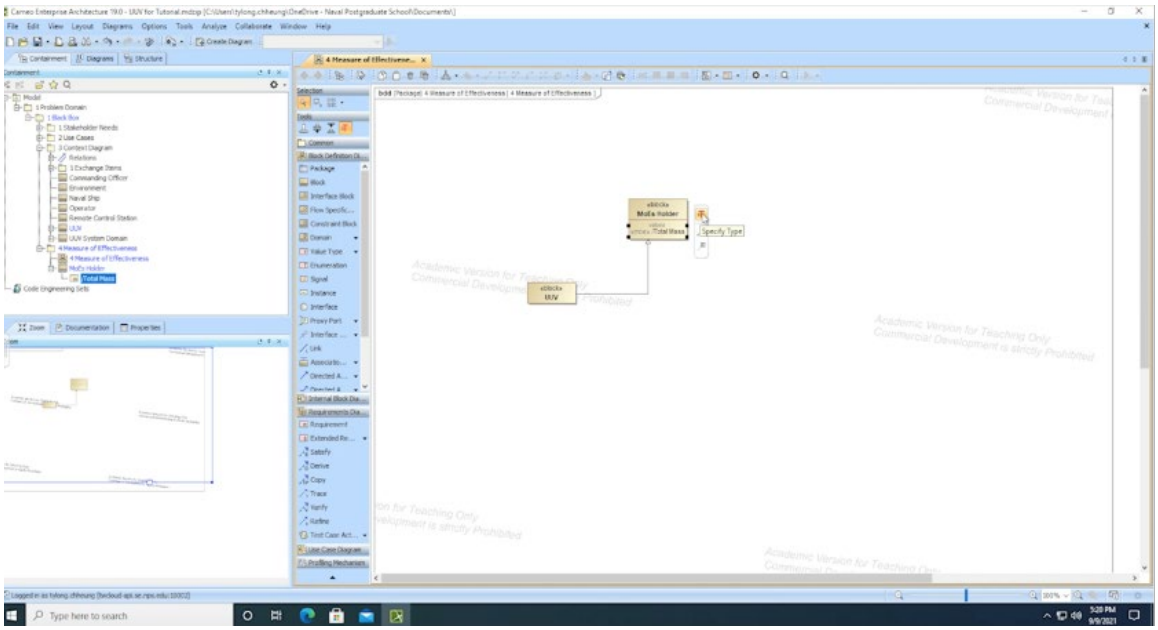
This will bring up the *Stereotype* popup window. Then from there we will start to type out *MOE* and select *MOE [Property]* and then hit the apply button.



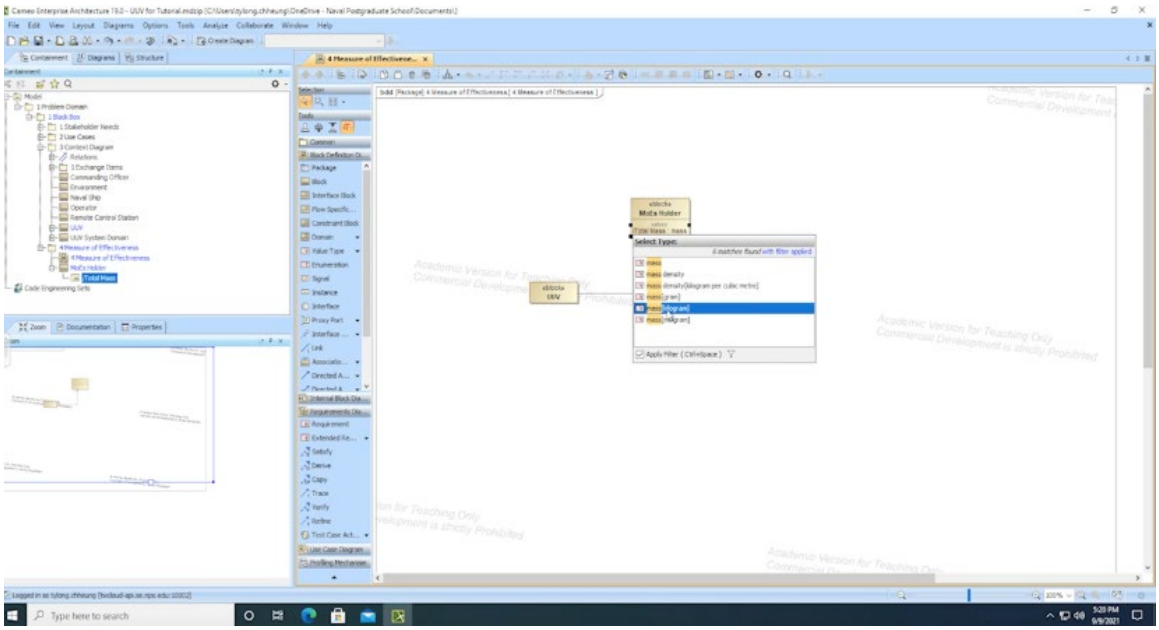
This will allocate the *Total Mass* value property to the *MOE Holder* block. Then we will right click the *Total Mass* value property on the diagram plane. From the drop-down menu we will choose *Is Derived*.



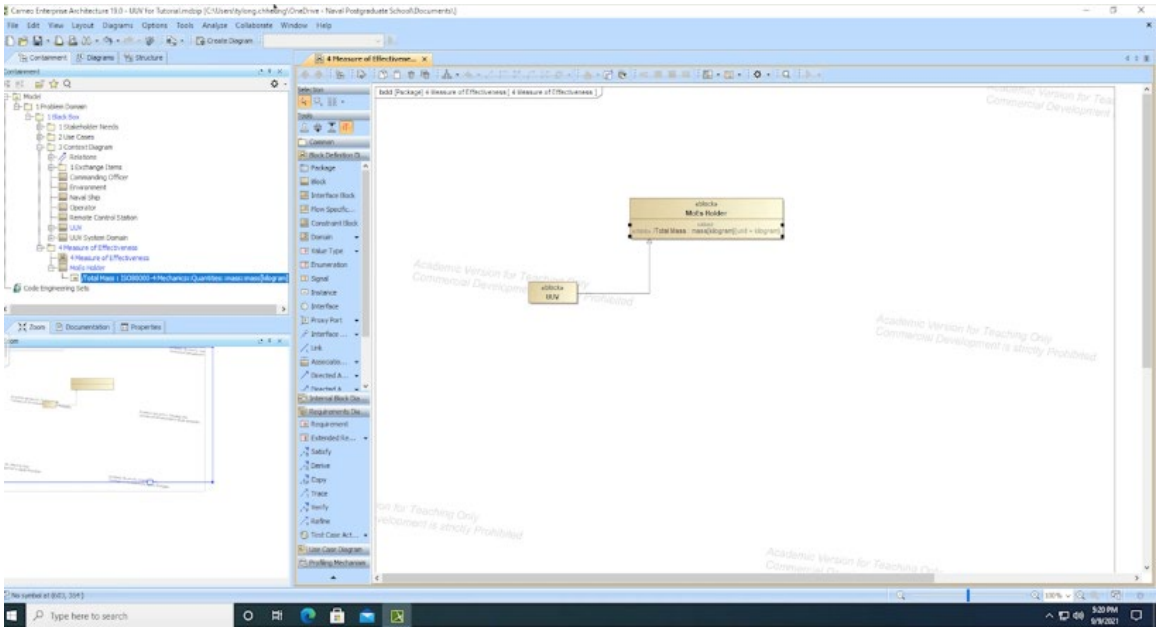
Once we have the *Total Mass* value property allocated to derive then we will select the *Specific Type(:T)* from the smart manipulator tool menu.



This will bring up the *Specific Type(:T)* popup window. For *Specific Type* we can type up *mass* and select *mass [kilogram]*.



This will give you the value of the MOE *Total Mass* that is derived and the mass is units of kilogram.



This is how you create the initial MOE block. Thank you.

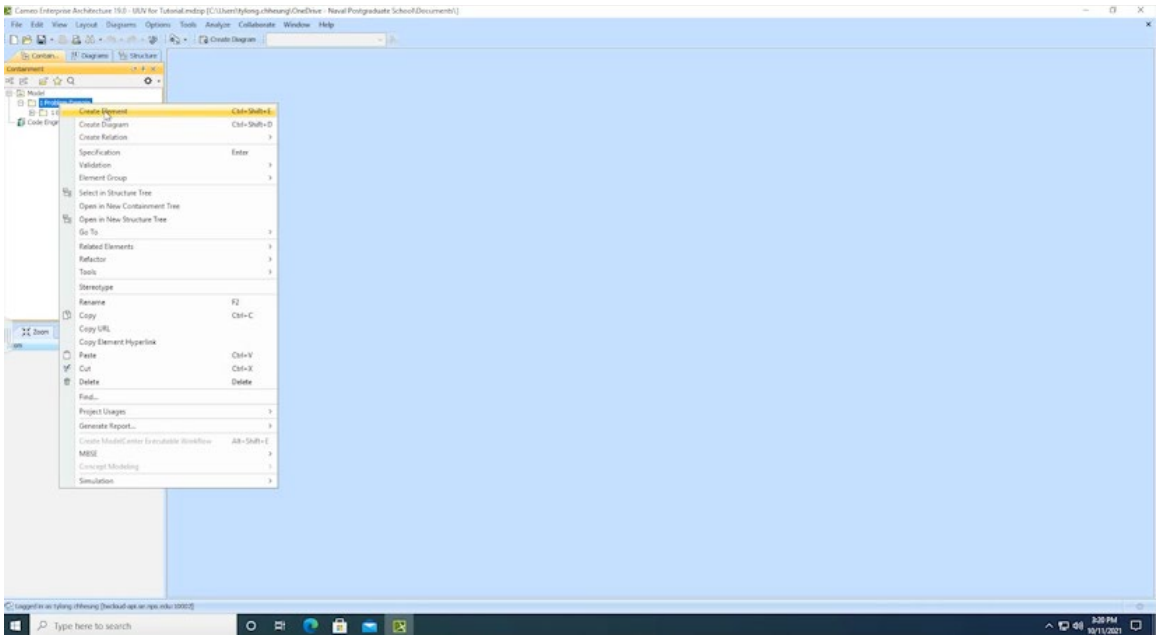
W2: Functional Analysis Tutorial

Background:

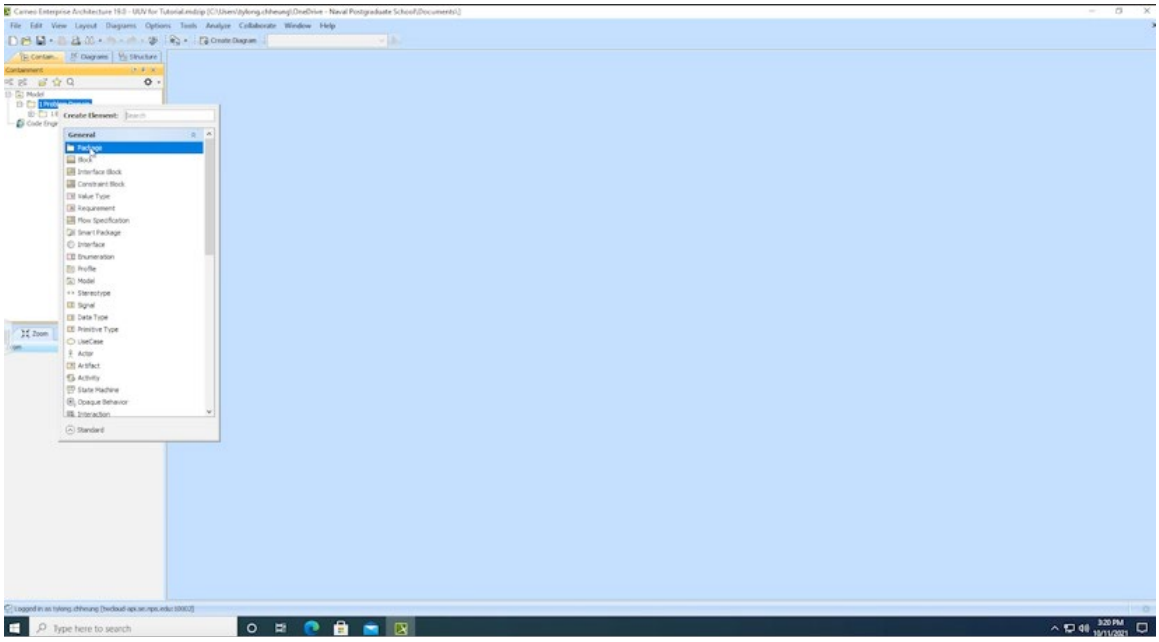
This section marks the start of the white-box perspective, where detail in how the system shall operate will be understood. This part of the model goes into a deeper analysis of functions and are grouped together by functional blocks. Iterations of functional decomposition can be performed to represent the same detail in the problem definition. From there, logical subsystems can be identified as well as the interfaces between them. In this part of the tutorial, functions of the system of interest are specified and decomposed using SysML activity diagrams. Each function can be decomposed as many times as needed to start to identify the logical subsystems that perform these functions.

Step-By-Step Guide:

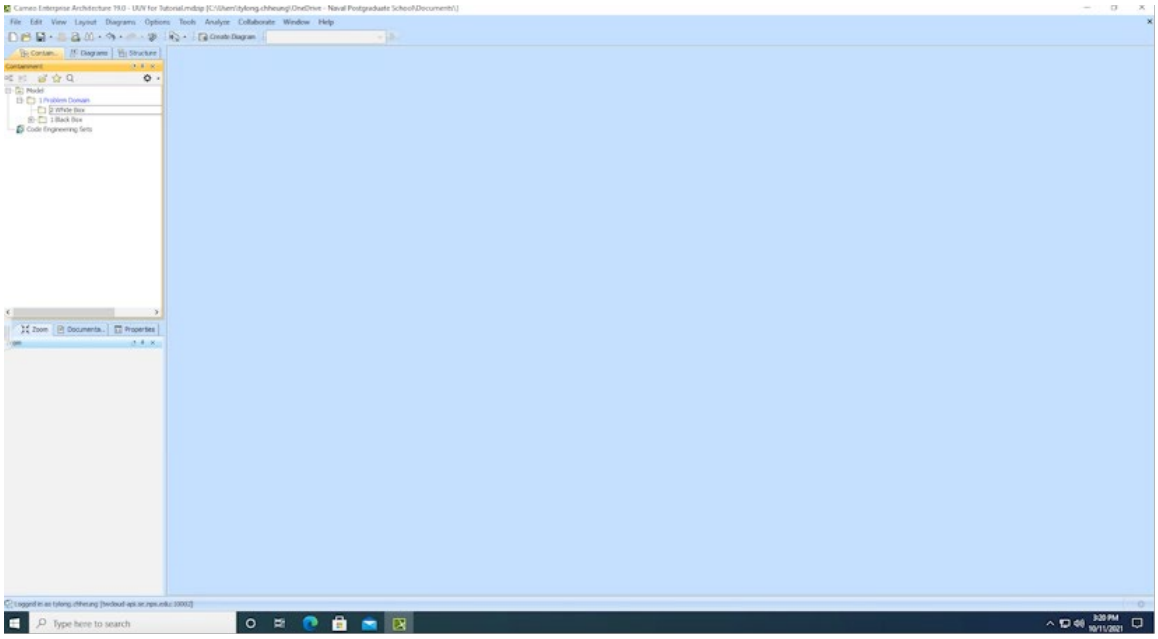
In this video, we will cover how to create the functional analysis of the UUV. We will start off by right clicking the *Problem Domain* in the containment tree. And from the drop-down menu select the *Create Element* option.



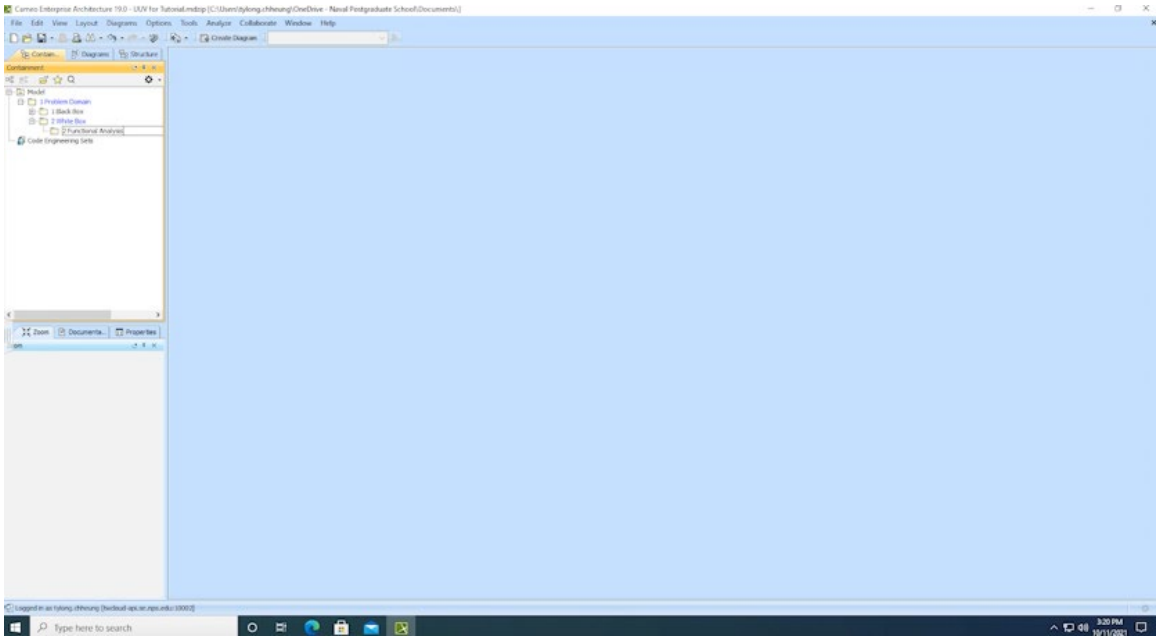
In doing this the *Create Element* popup menu will appear, where we will click on *Package* option.



This will create a new package in the containment tree, which we will label the package as *2 White Box*. The functional analysis is in the *White Box* category. Once you hit enter on the keyboard, the *White Box* package is created.

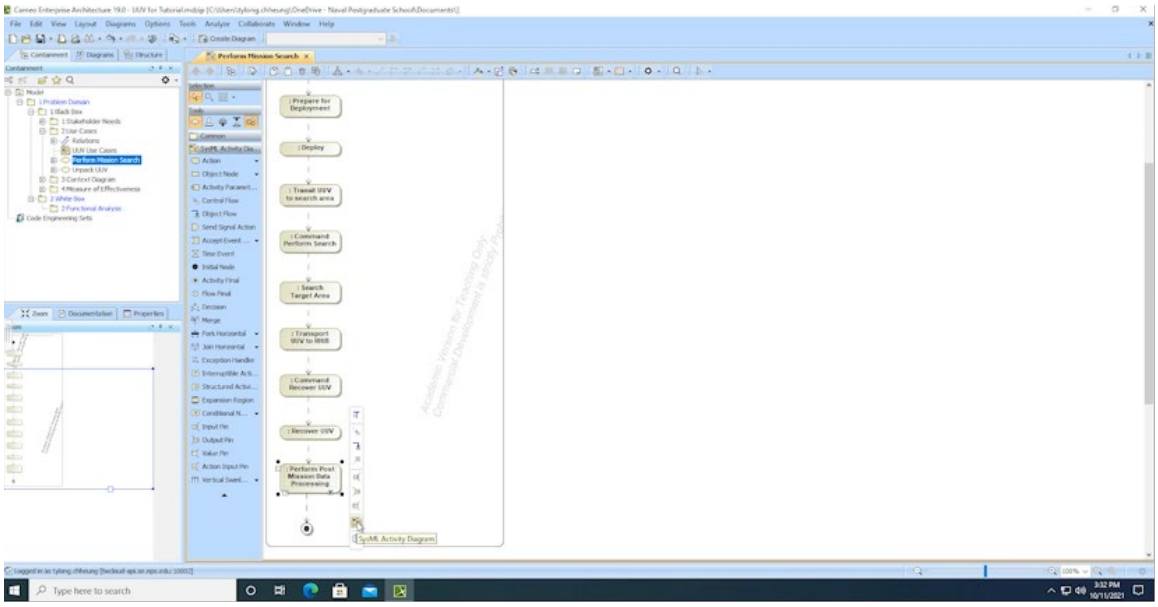


Then we will right click on the *White Box* package in the containment tree and from the drop-down menu we will select once again *Create Element* option. This will make the *Create Element* popup menu appear and we will select the *Package* option to create another package. We will label the new package *2 Functional Analysis* and now the *2 White Box* package has the sub package *2 Functional Analysis*.

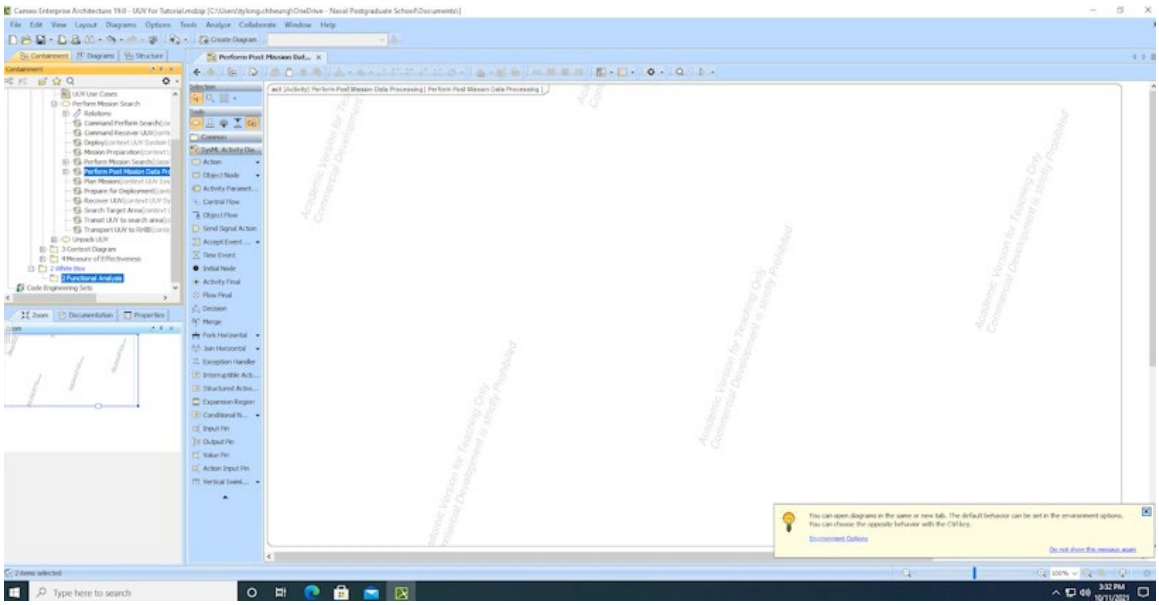


Once you have the *Functional Analysis* package created, you will then extend the *Black Box* package and extend the subfunction *Use Cases* and open the *Perform Mission*

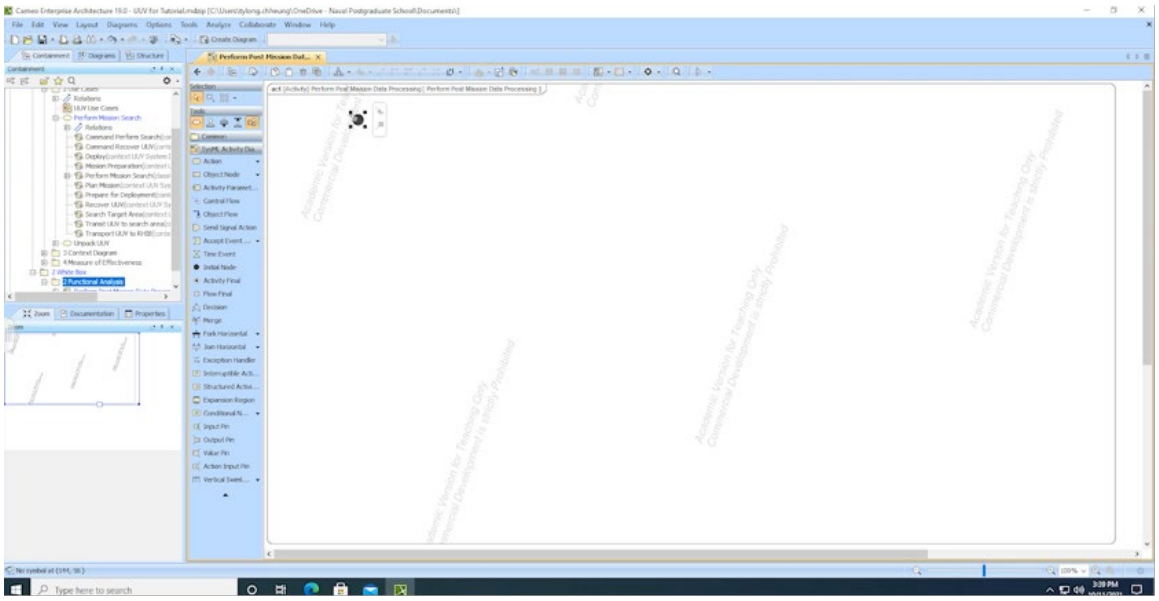
Search diagram that was previous completed in the prior tutorials. Then in the *Perform Mission Search* diagram select and right click the *Perform Post Mission Data Processing* action. In the smart manipulator tool bar we will then select the *SysML Activity Diagram* option.



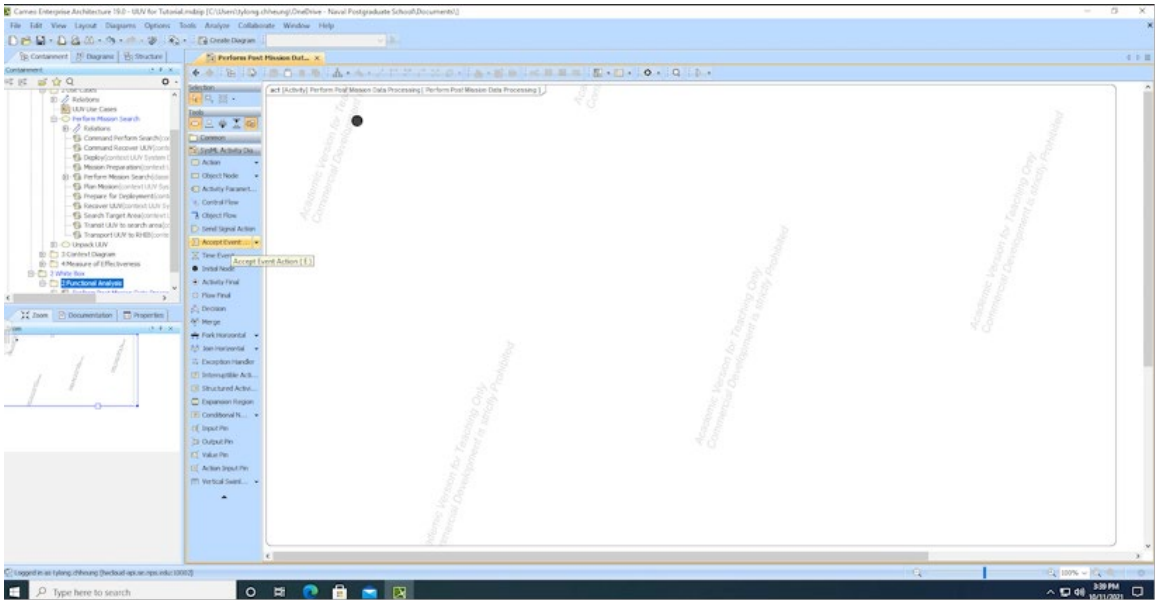
This will create the *SysML Activity Diagram* for the *Perform Post Mission Data Processing* call behavior action. Then we will head over to the containment tree on the left menu and expand the *Perform Post Mission Data Processing* activity. From there we will select the activity *Perform Post Mission Data Processing (Context UUV System Domain)* and drag that to the *Functional Analysis* package.



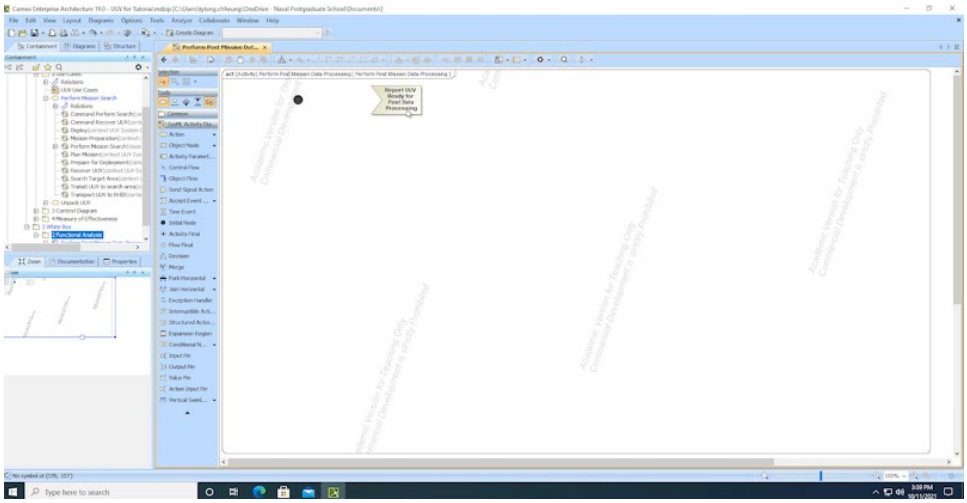
Then you extend the *Functional Analysis* package and then you will see the *Preform Post Mission Data Processing* activity. Having the activity diagram open ensures that you will be able to go to the diagram pallet and be able to select the *Behavior Creation Mode* button. But before that we will select the *Initial Node* button and drag it to an empty space in the diagram pane. We do this in order to start the *Functional Analysis*.



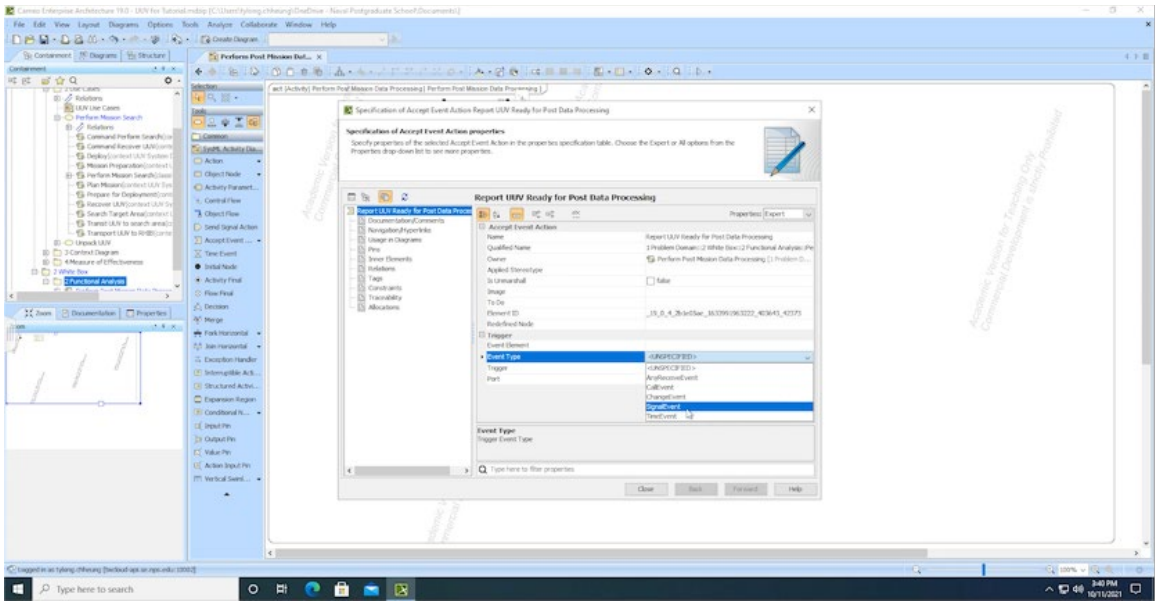
Once you select this node, we need to create an event that calls upon this function. Therefore, in the diagram pallet we will select the *Accept Event Action (E)* button.



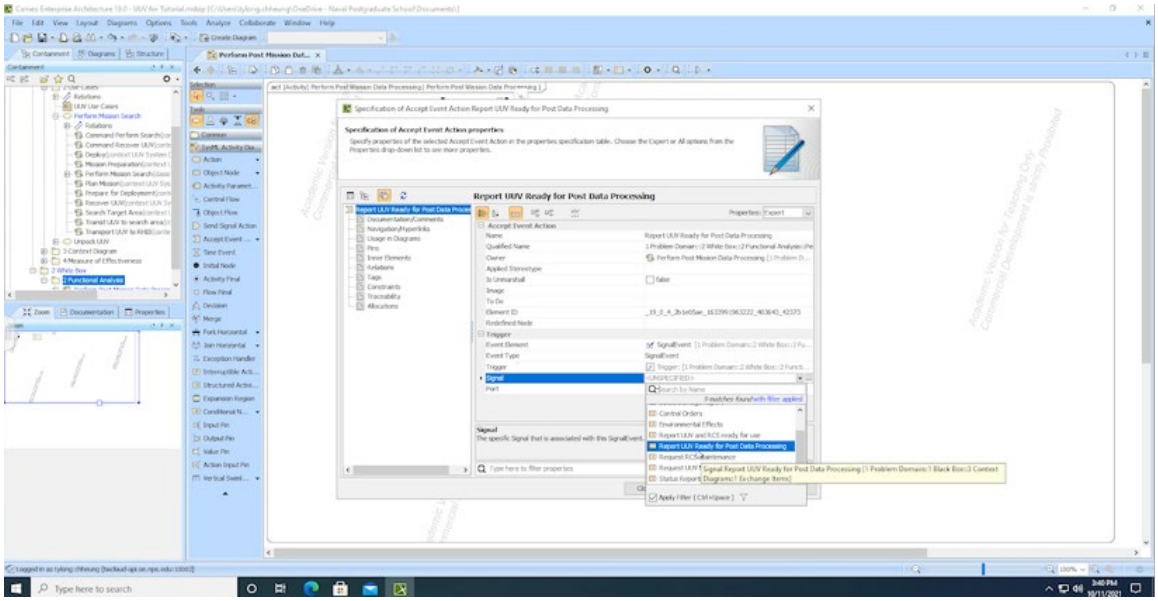
Then we will drag it to an empty space in the diagram pane. And we will label the event as *Report UUV Ready For Post Data Processing*.



Then we will open the event and a *Specification Of Accept Event Action Report UUV Ready For Post Data Processing* popup window will appear. First, we are going to specify the *Trigger* and from the drop-down menu we will select the *SignalEvent* Trigger.

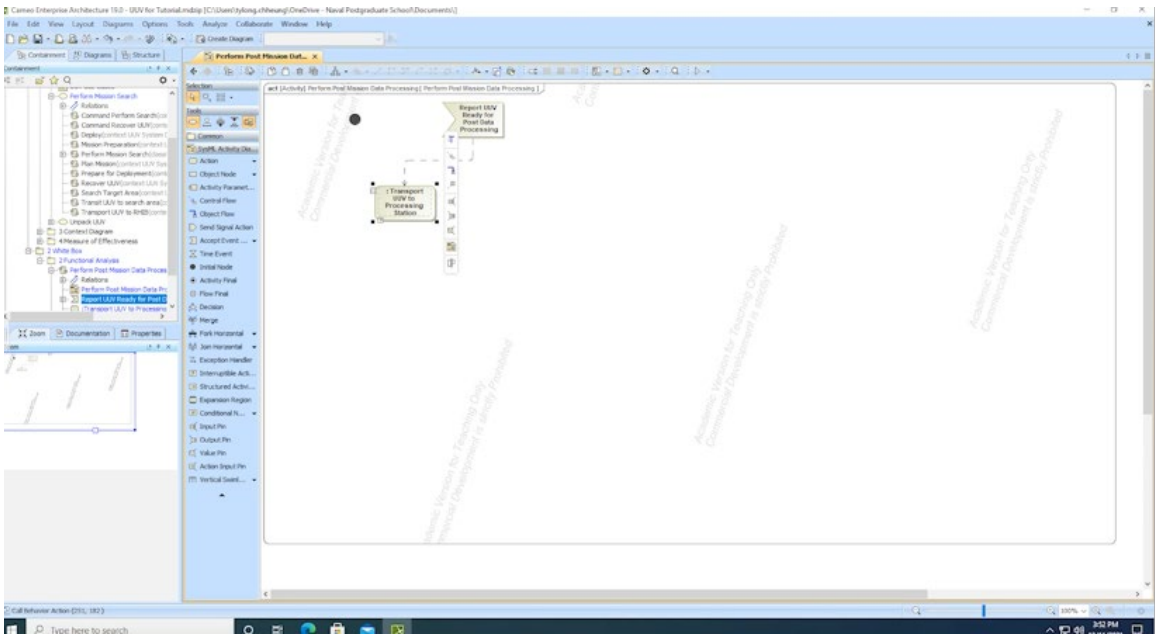


Then we are going to setup the *Signal* from the drop-down menu we will select the *Report UUV Ready For Post Data Processing* Exchange Item. And then click on the *Close* button to close the *Specification Of Accept Event Action Report UUV Ready For Post Data Processing* popup window.

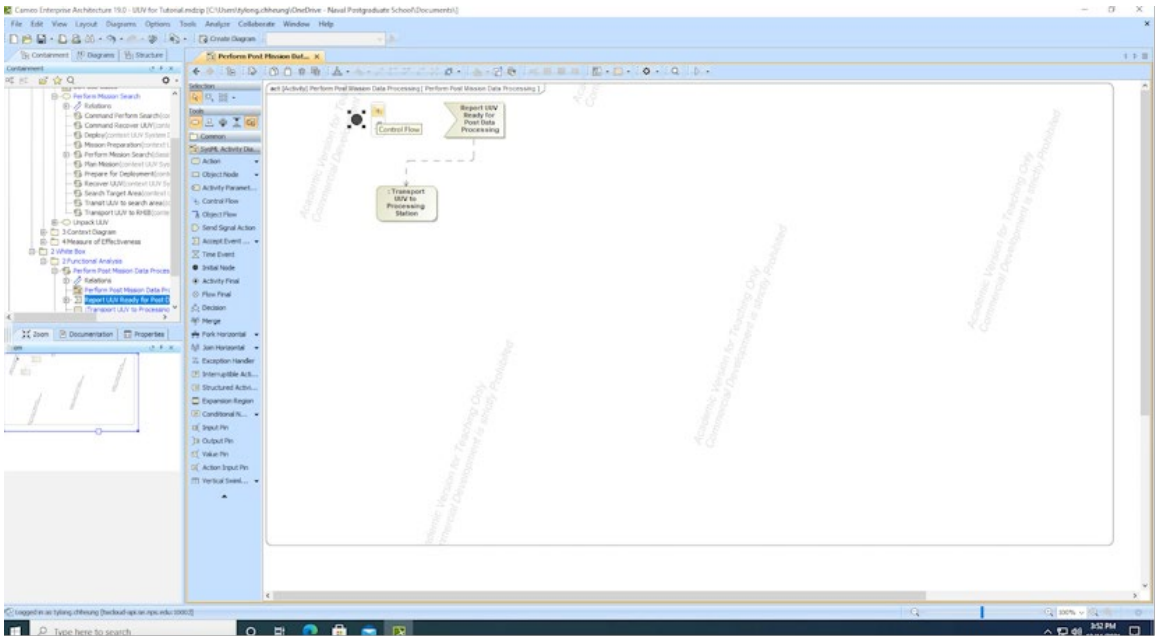


Now we are ready to start building the *Functional Analysis*. From the smart manipulator tool bar of the *Report UUV Ready For Post Data Processing* event item we will be selecting the *Control Flow* button and we are going to click an empty space in the

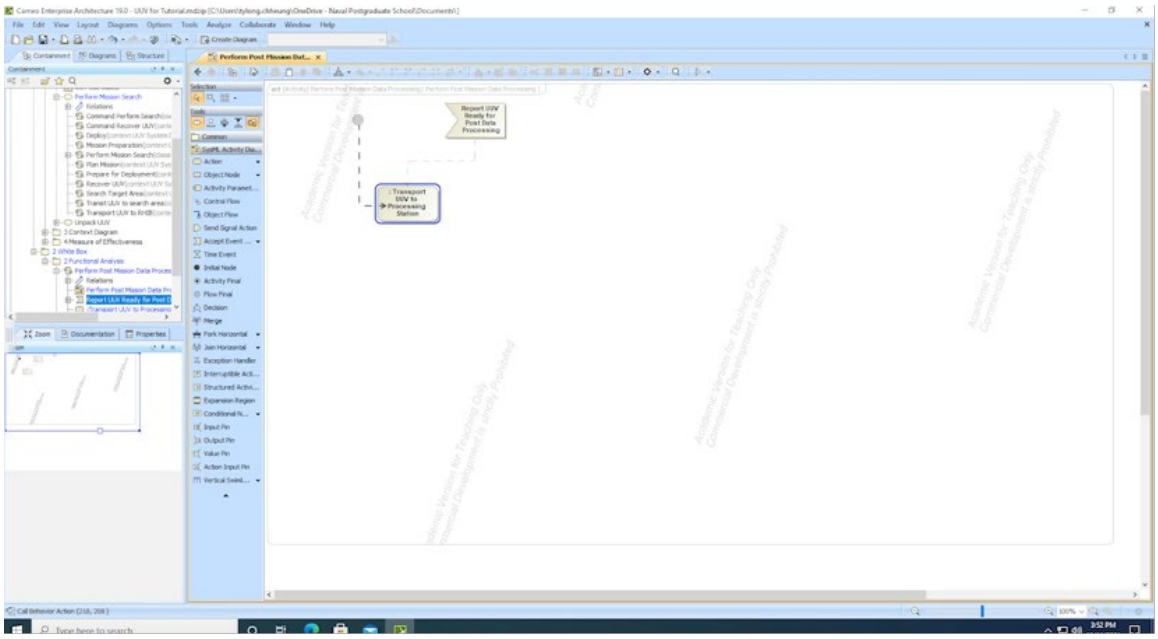
diagram pane. This will create an action which we will label as *Transport UUV to Processing Station*.



We will then go back to the *Initial Node* and from the smart manipulator tool bar we will select the *Control Flow* button.



Then we will drag it over the *Transport UUV to Processing Station* Action which will be highlighted in blue.



From the manipulator tool bar from the *Transport UUV to Processing Station* Action, we will select the *Control Flow* button and we are going to click an empty space in the diagram plane. This will create an action which we will label as *Connect External Drive To Control Station*.

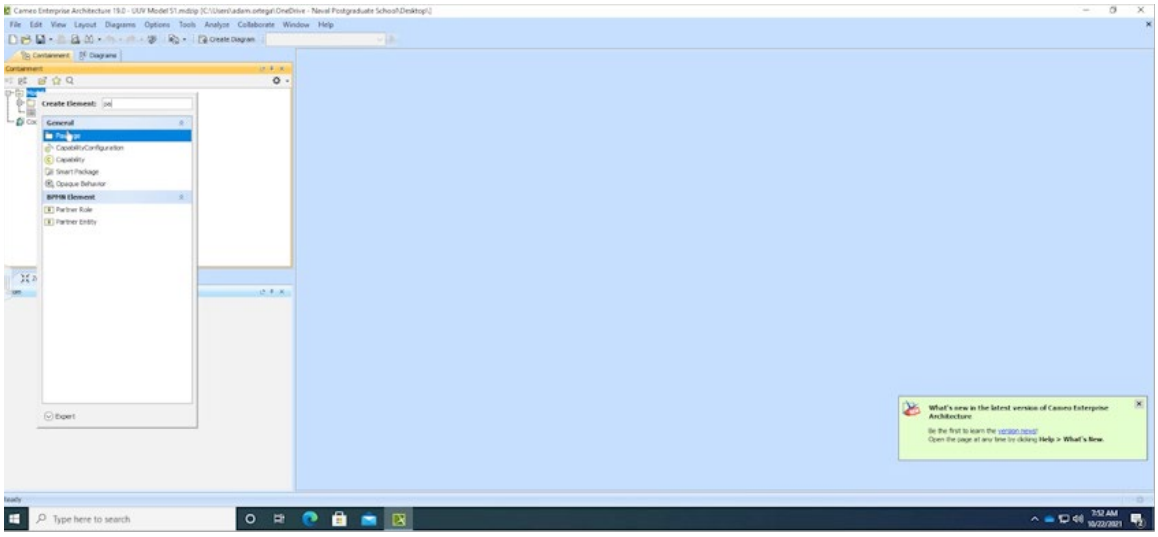
S1: Initial System Requirement Tutorial

Background:

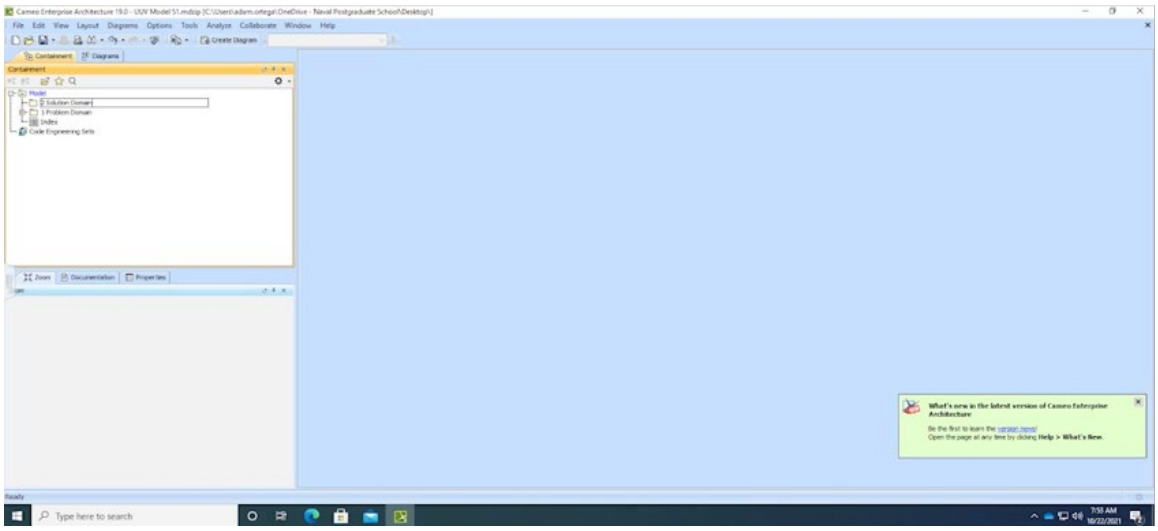
With the problem domain complete, the start of the solution domain begins to define the specifics of the logical system design. One or more solutions can be created for the problem, however, for the purpose of this tutorial, only one solution architecture will be defined for implementation. A separate package will be created for the solution domain and initial system requirements will be derived and refined from the stakeholder needs in the problem domain. System requirements can be specified using a SysML requirement diagram, using relationships to point from system requirements to stakeholder needs. The rest of the system solution architecture can be used to refine the final system requirements specification.

Step-By-Step Guide:

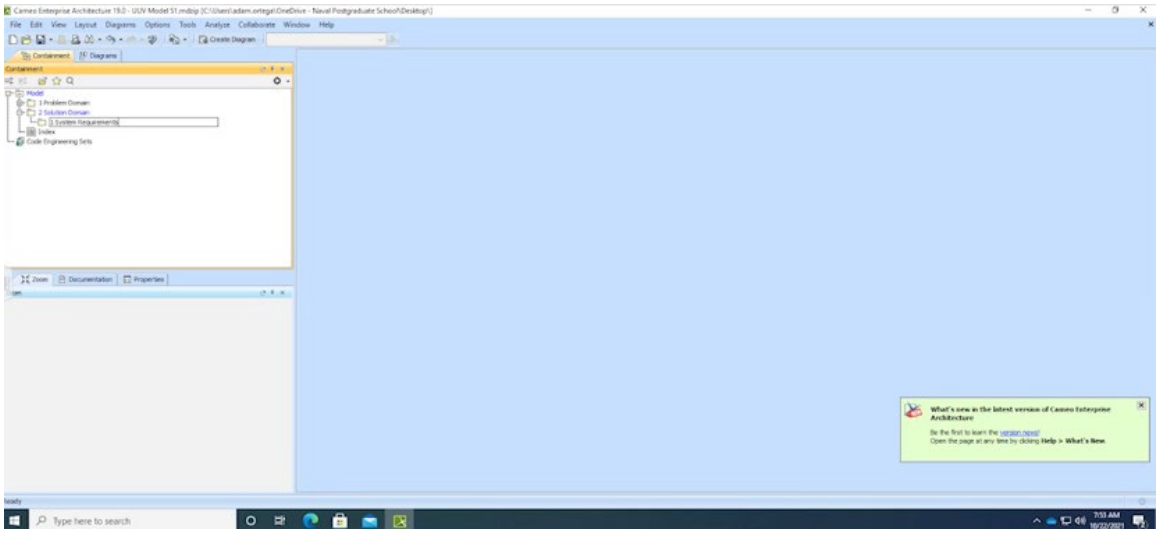
Now that the *Problem Domain* is completed, we will be starting the *Solution Domain*, where we start to decide on a precise model of the logical design of the UUV. So, we will begin developing the actual solution for the problem. First, we need to create a new package for the *Solution Domain*, so we will right click on the *Model* package and from the drop-down menu we will select the *Create Element* option. Then a *Create Element* popup menu will appear and on the search bar we will type *pa* and select the package option.



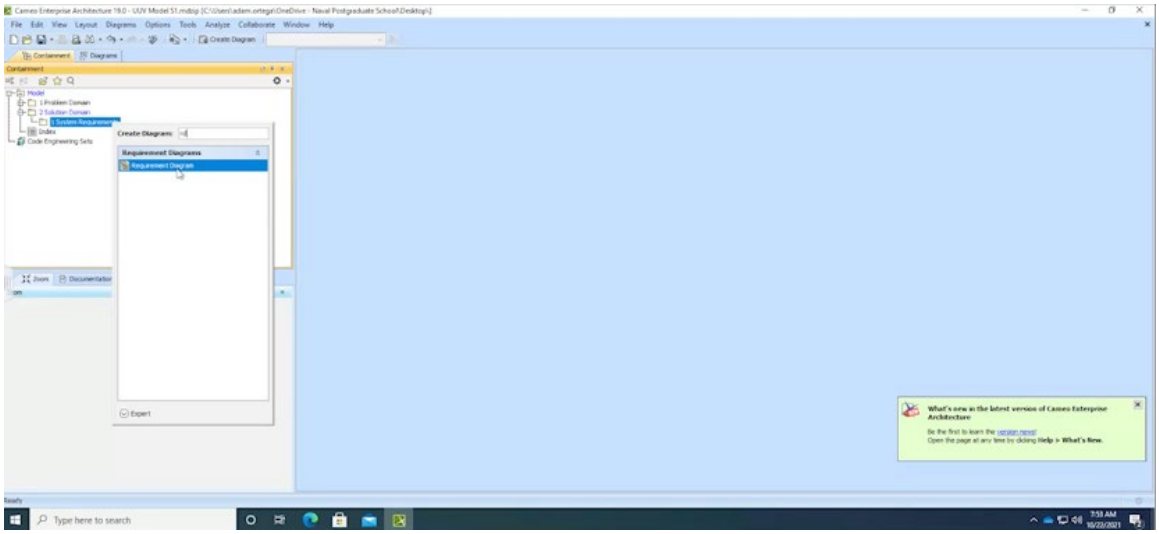
Then we will name the new package as *2 Solution Domain*.



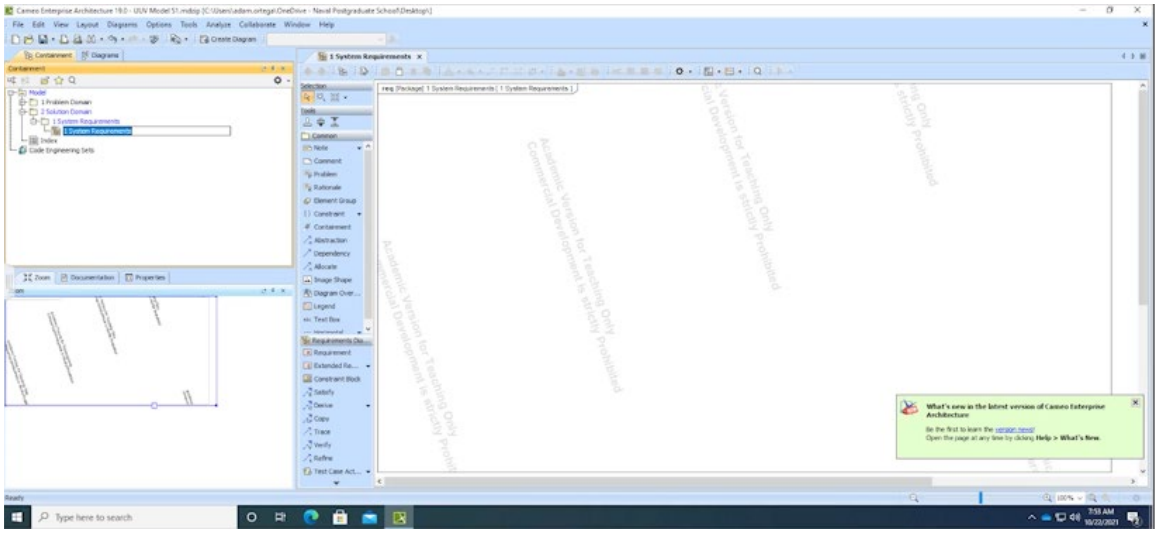
We will create another package inside the *Solution Domain* for *System Requirements*. So we will right click the *Solution Domain* package and from the drop-down menu we will select the *Create Element* option. Then a *Create Element* popup menu will appear and on the search bar we will type *pa* and select the package option. Then we will name the new package as *1 System Requirements*.



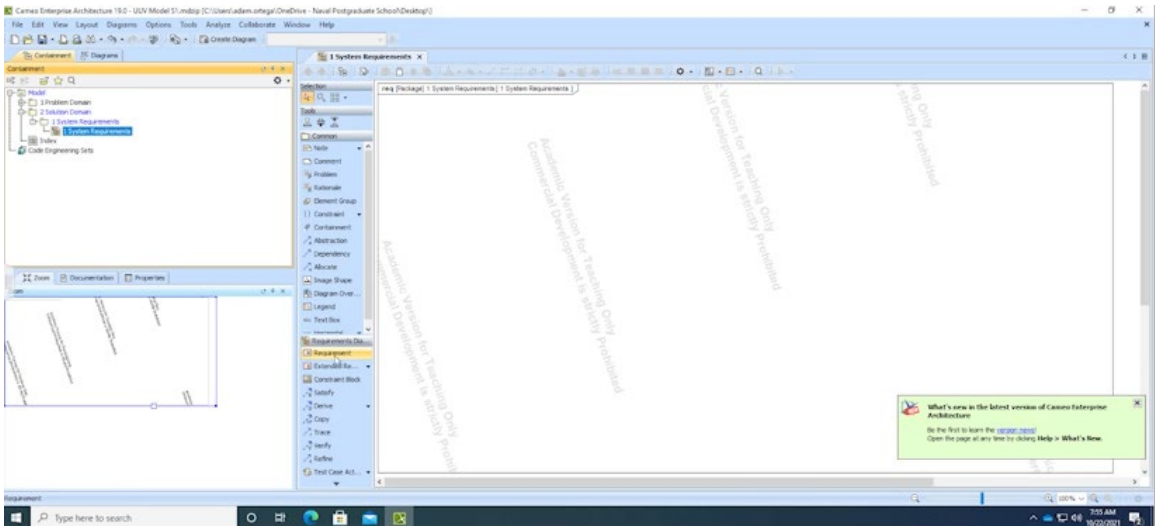
Now we will right click this package and from the drop-down menu we will select the *Create Diagram* option. Then a *Create Diagram* popup menu will appear and on the search bar we will type *rd* and select the *Requirement Diagram* option.



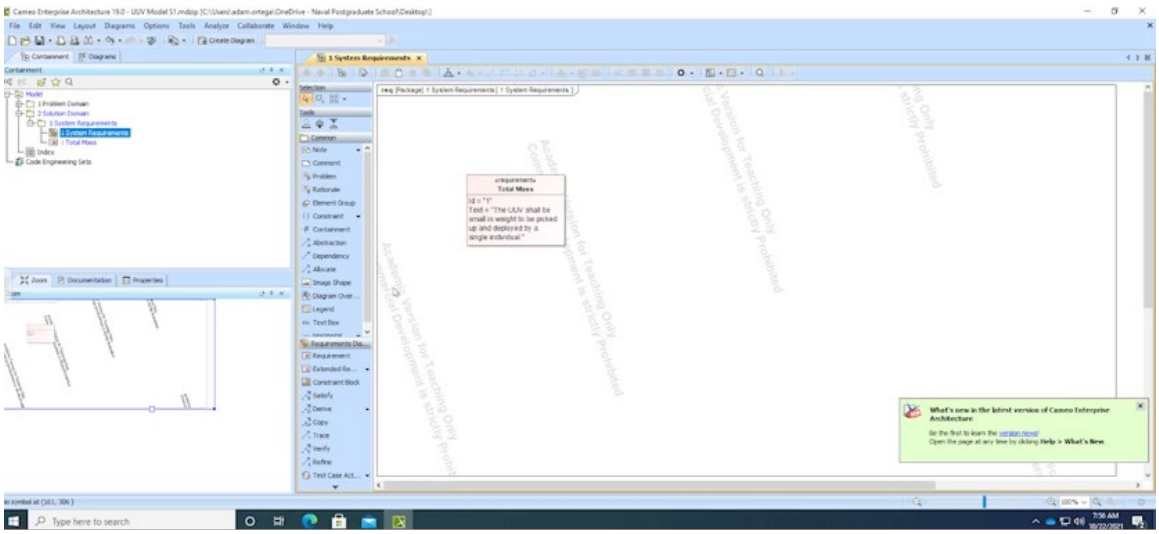
And we can go ahead and leave the generated label for the diagram the same, which is labeled as *1 System Requirement*.



Now with the diagram plane open we will click on the *Requirement* option in the diagram pallet.

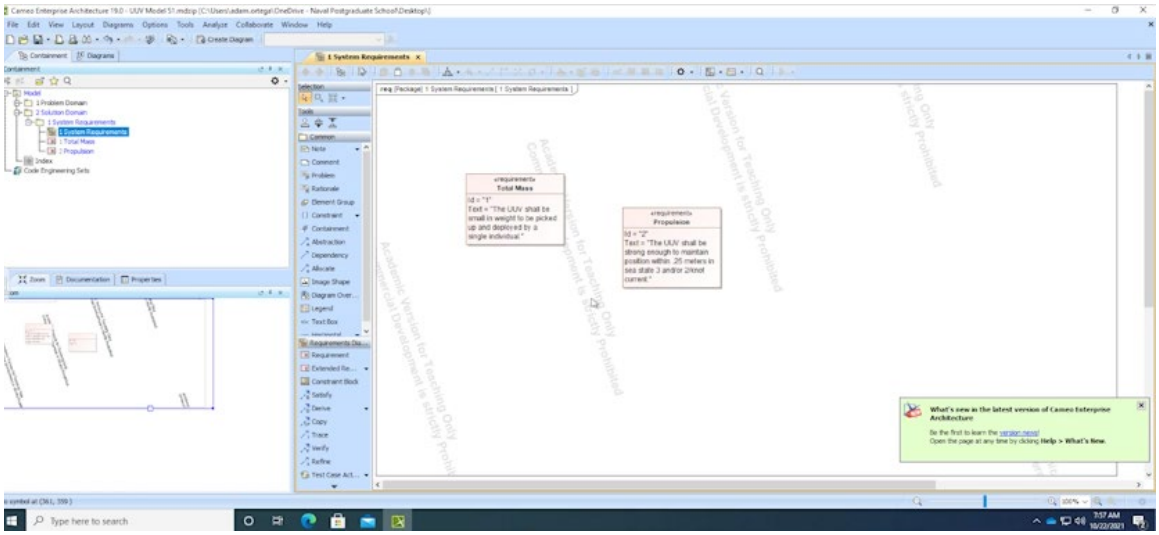


And then drag it to an empty space in the diagram pane. Then the box will appear and we will call this one *Total Mass*. Then we will click on the *Text=* in the lower half of the diagram box. And we will type: *The UUV shall be small in weight to be picked up and deployed by a single individual*. Then we will click outside the requirement and anywhere else in the diagram plane.

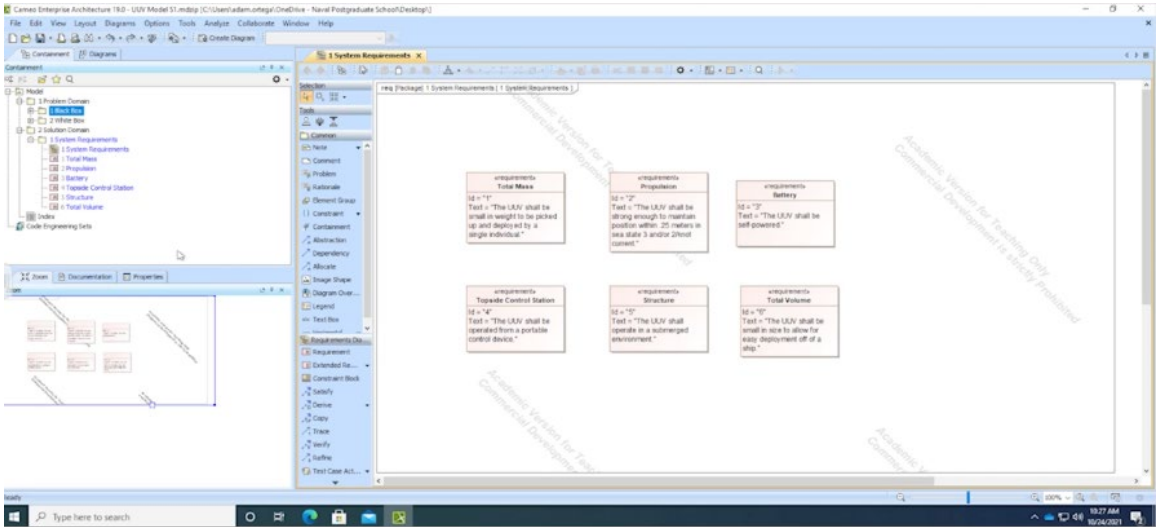


Now we will go ahead and create the rest of the requirements in the same manner. So again click on the *Requirement* button in the diagram pallet and drag it to an empty space in the diagram plane. Then the box will appear and we will call this one *Propulsion*. Then we will click on the *Text=* in the lower half of the diagram box. And we will type: *The UUV shall be strong enough to maintain position within .25 meters in sea state 3*

and/or 2/knot current. Then we will click outside the requirement box and anywhere else in the diagram plane.

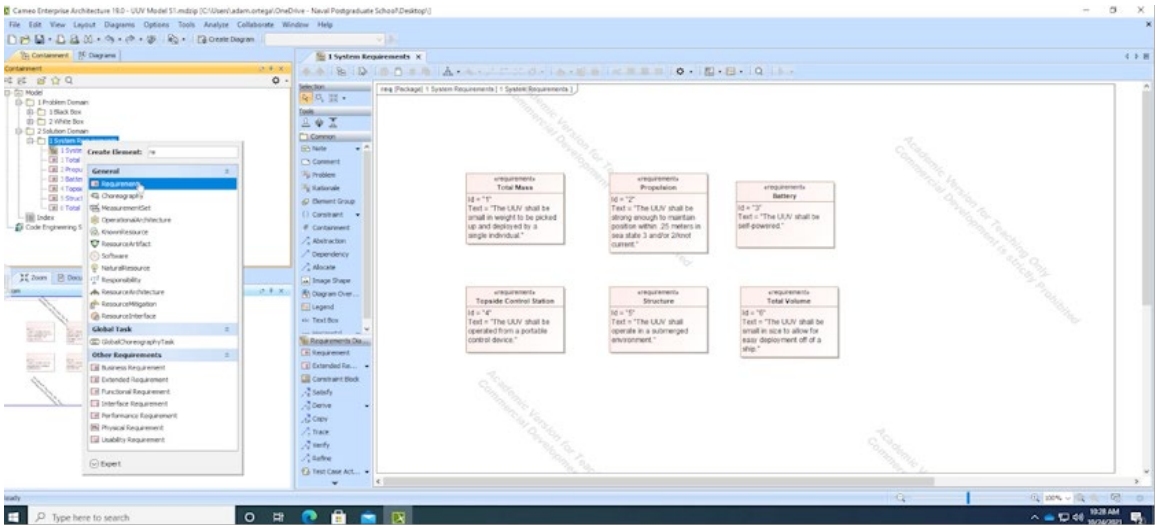


We will continue to buildout the rest of the requirements in the same fashion. Therefore, the rest of the requirements should be similar to the ones in the photo below which are the following: *Battery, Topside Control Station, Structure,* and *Total Volume*.

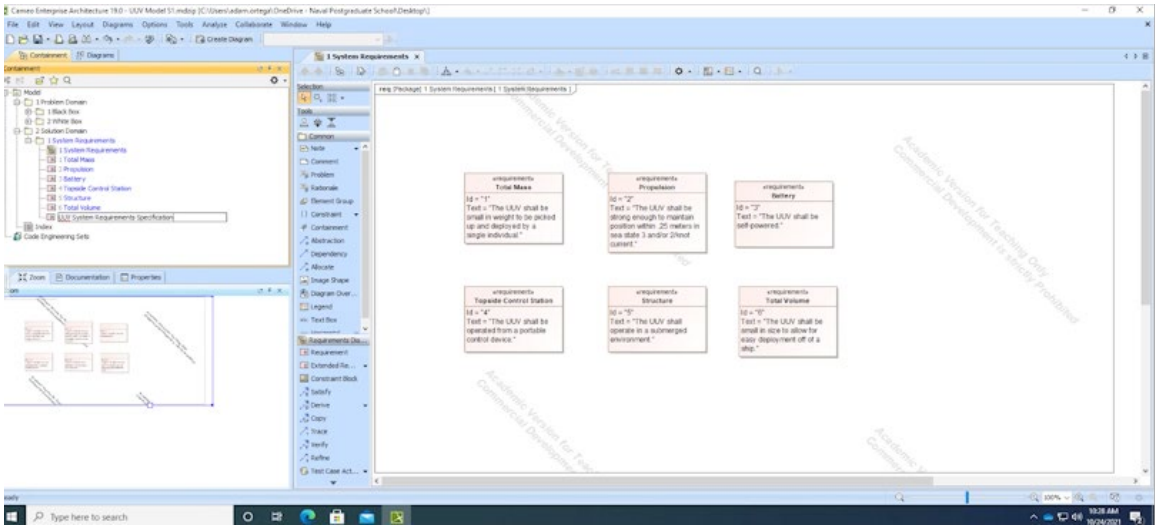


Now before we can group and organized these requirements, we need to create another requirement similar to what was done in the B1-W1 tutorial. So, we will right

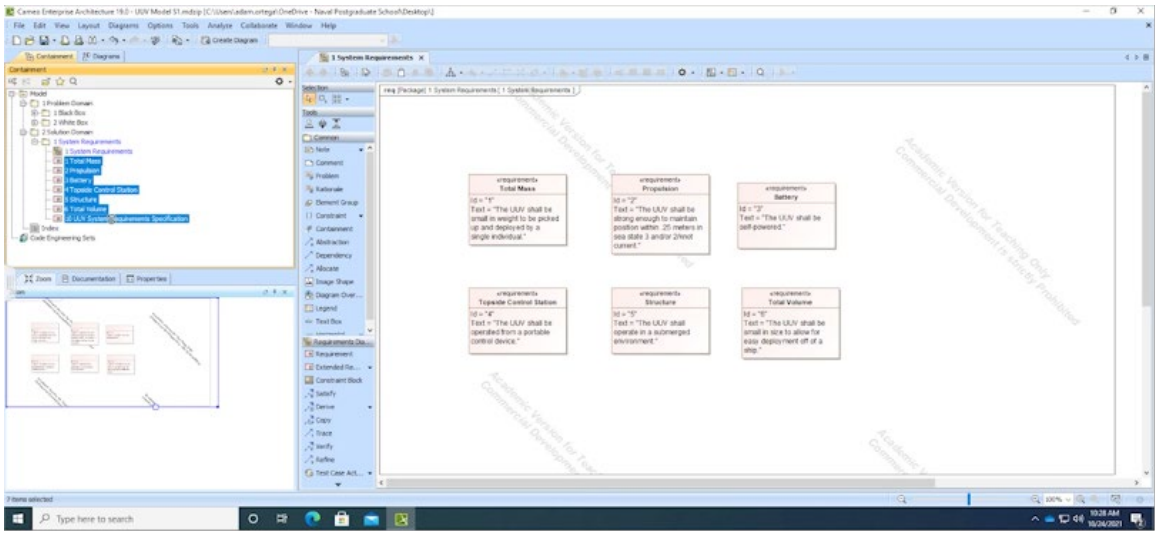
ahead and right click the *1 System Requirement* package and from the drop-down menu we will select the *Create Element* option. Then a *Create Element* popup menu will appear and on the search bar we will type *re* and select the requirement option.



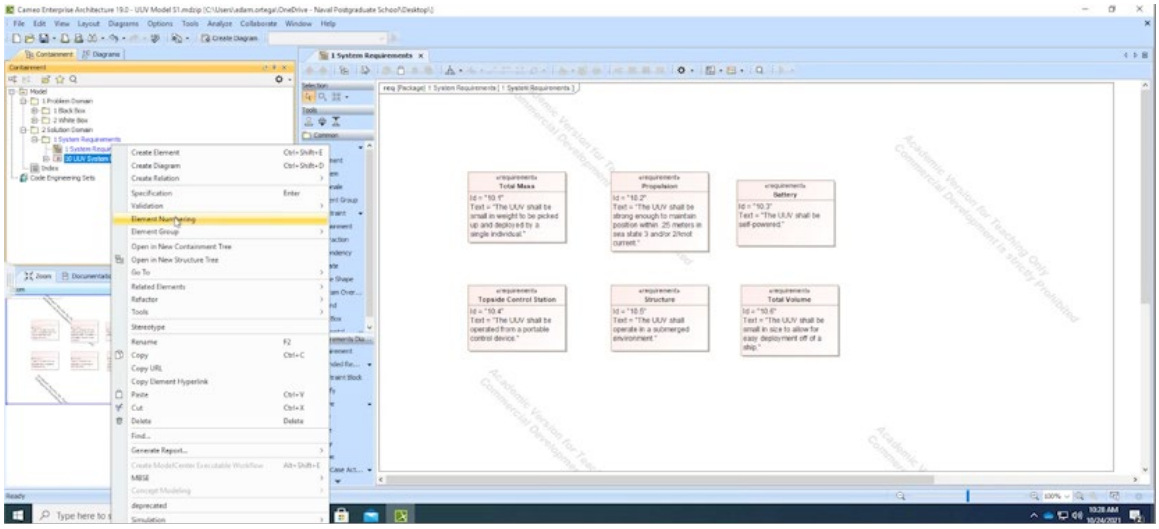
Then we will label the new requirement as *UUV System Requirements Specification*.



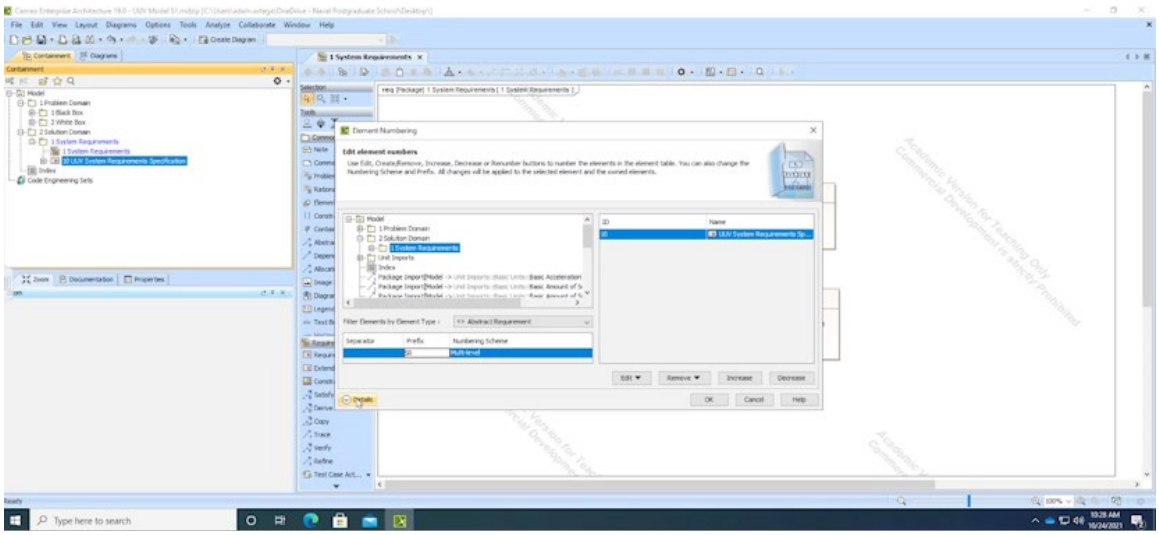
Now we can select all of the previously created requirements and drag that on top of the requirement that was just created.



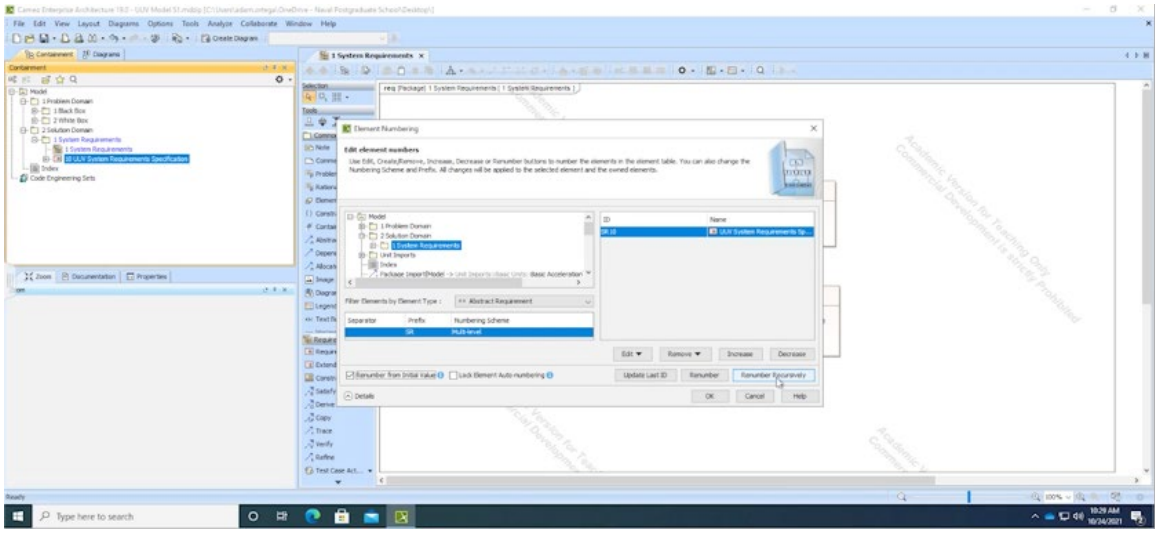
Then we are going to right click it and from the drop-down menu we will select the *Element Numbering* option.



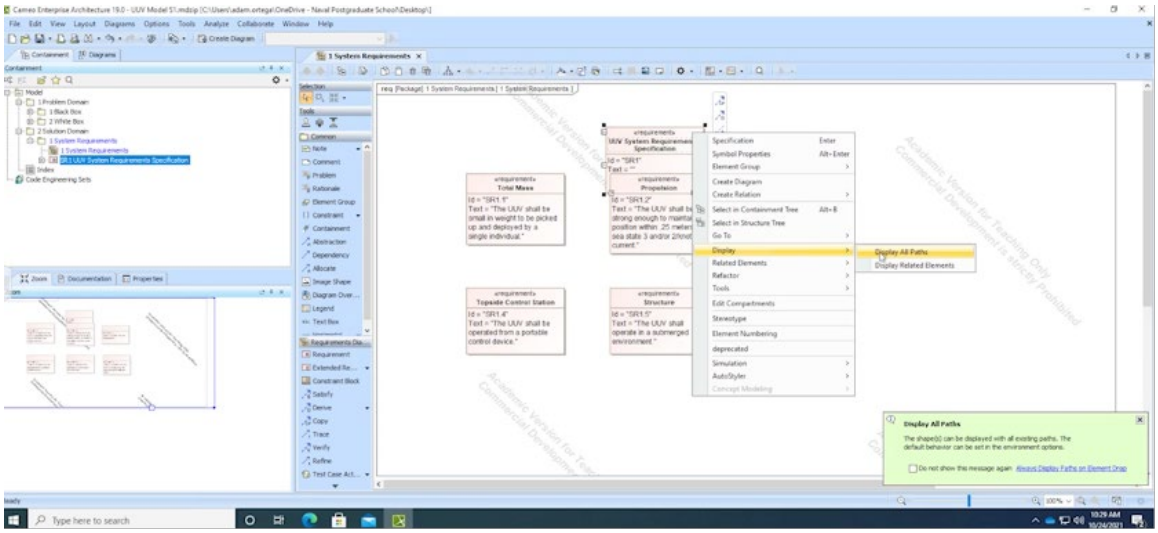
Then an *Element Numbering* popup window will appear and in the prefix box we will type out *SR* for System Requirements.



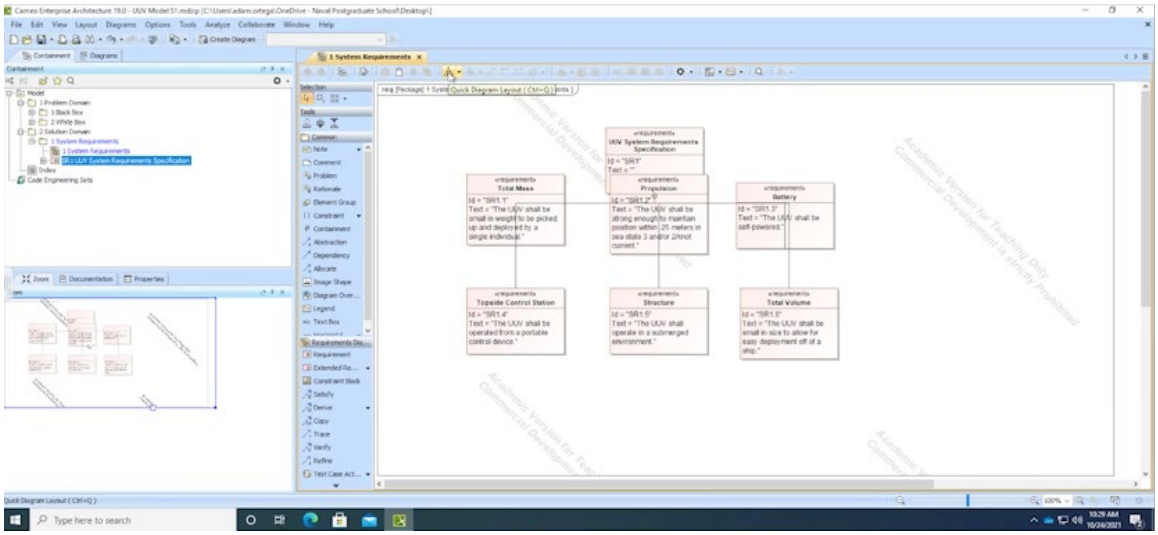
Then we will click on the *Details* button and this will bring up two new check boxes. We will click and check the *Renumber From Initial Value* option and then click on the *Renumber Recursively* button. Then we click on the *Ok* button to close the *Element Numbering* popup window.



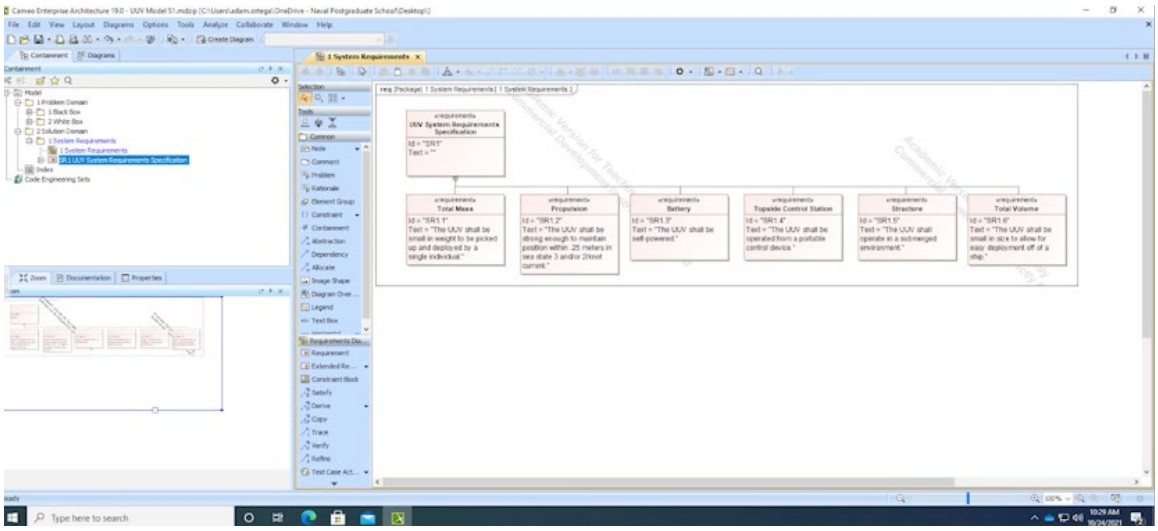
Now we will drag this requirement to the diagram plane. We can right click and from the drop-down menu we will select the *Display* option and then select *Display All Paths.*



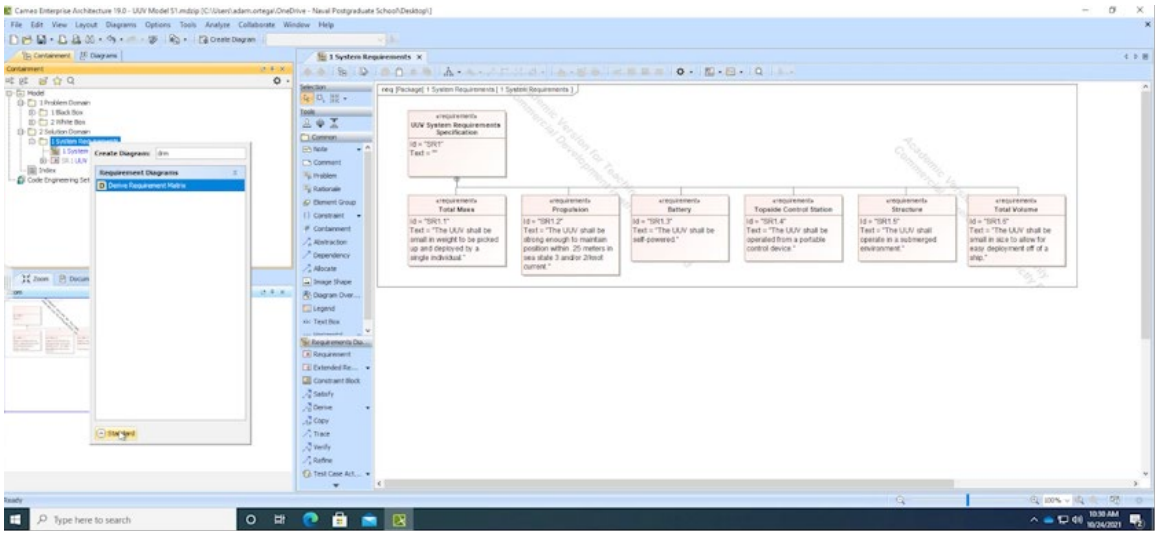
Then we click on an empty space in the diagram plane and then we click on the *Quick Diagram Layout* button.



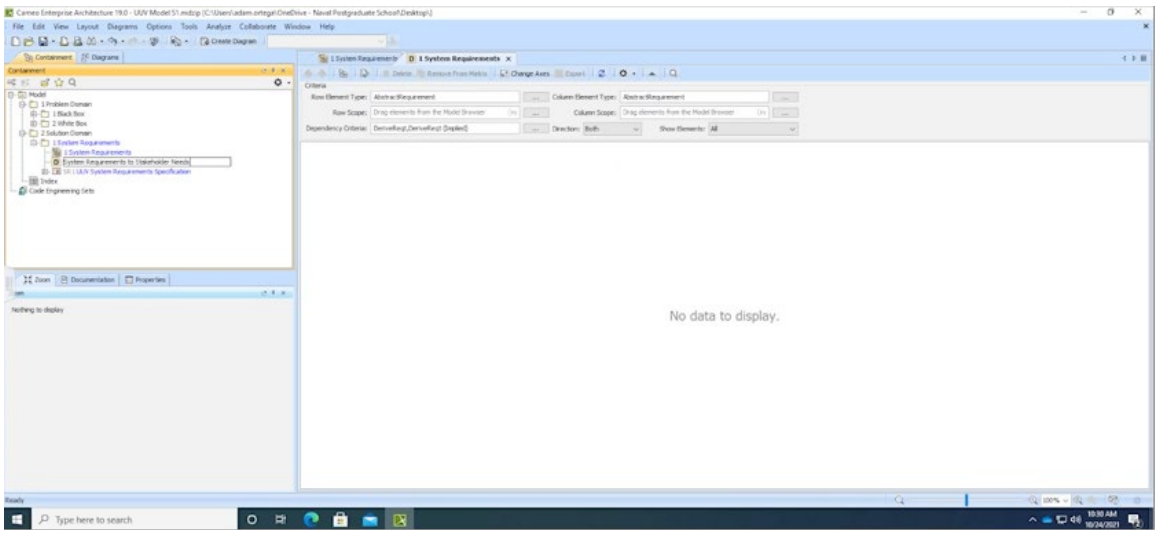
The diagram layout will be organized and look similar to the photo below.



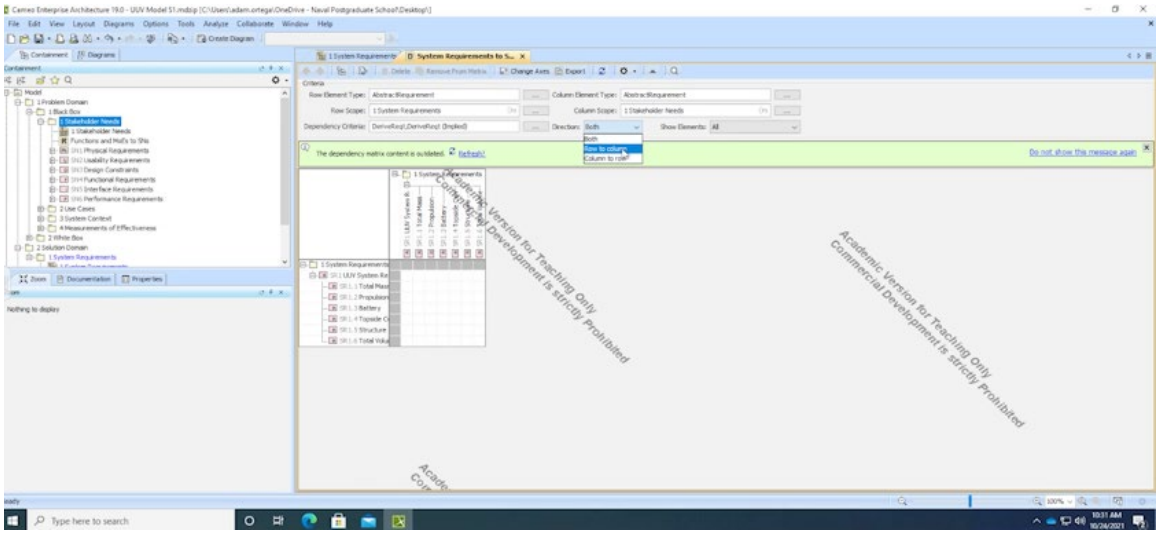
Now that this is complete, we will need to establish traceability to the stakeholder needs by creating a derived requirement matrix. So, we have to go back to the containment tree and we will right click the *1 System Requirement* package and from the drop-down menu we will select the *Create Diagram* option. Then a *Create Diagram* popup menu will appear and on the search bar we will type *drm*. If nothing comes up in the search list then click on the *expert* button on the bottom left corner of the *Create Diagram* popup menu. This will the show the result and we will select the *Derive Requirement Matrix* option.



Then we will name the *Derive Requirement Matrix* as *System Requirements to Stakeholder Needs*.

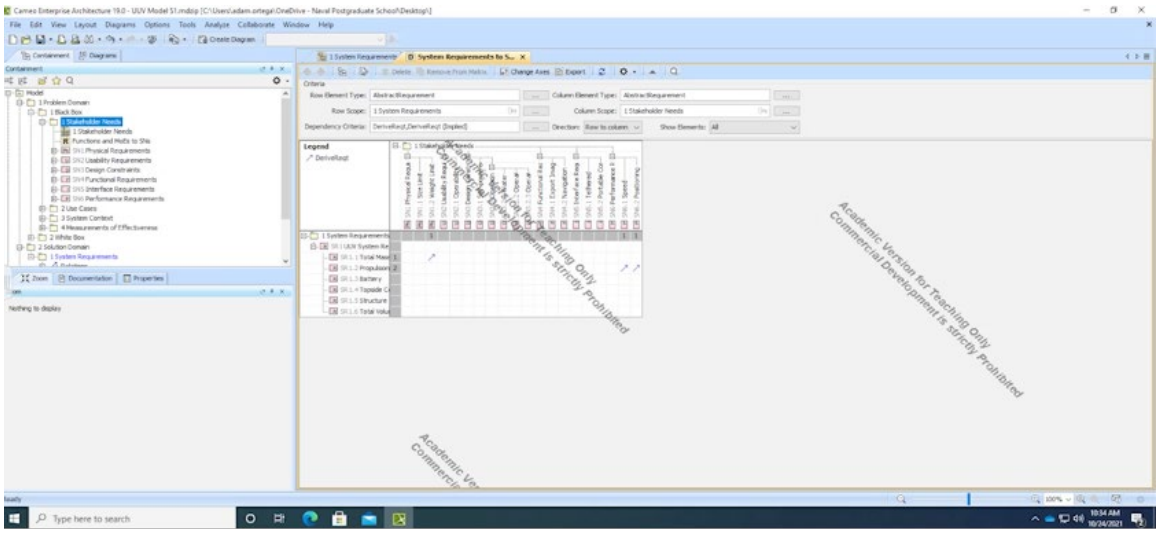


Now with the diagram open we will drag the *System Requirement* package to the *Row Scope*. Then we will grab the *Stakeholder Needs* package from *Black Box*. Therefore, we are going to go to the containment tree of the left-hand side and expand the *Black Box* and select the *Stakeholder Needs* package and drag it to the *Column Scope*. Then we can click on the *Direction* drop-down menu and select the *Row to Column* option. And we will click on the *Refresh* hyperlink to update the matrix.

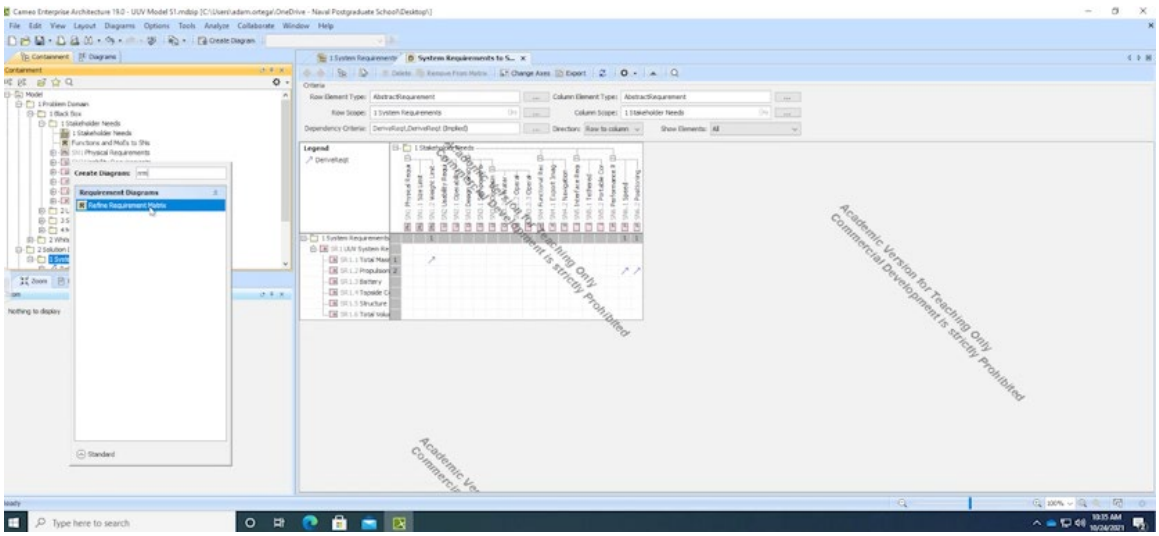


Now we can begin to specify the derivation relationship in the matrix by double clicking on the intersections we can create the derivation relationships between the two. So for example, we can select and double click on the *Total Mass* and the *Weight Limit*. We

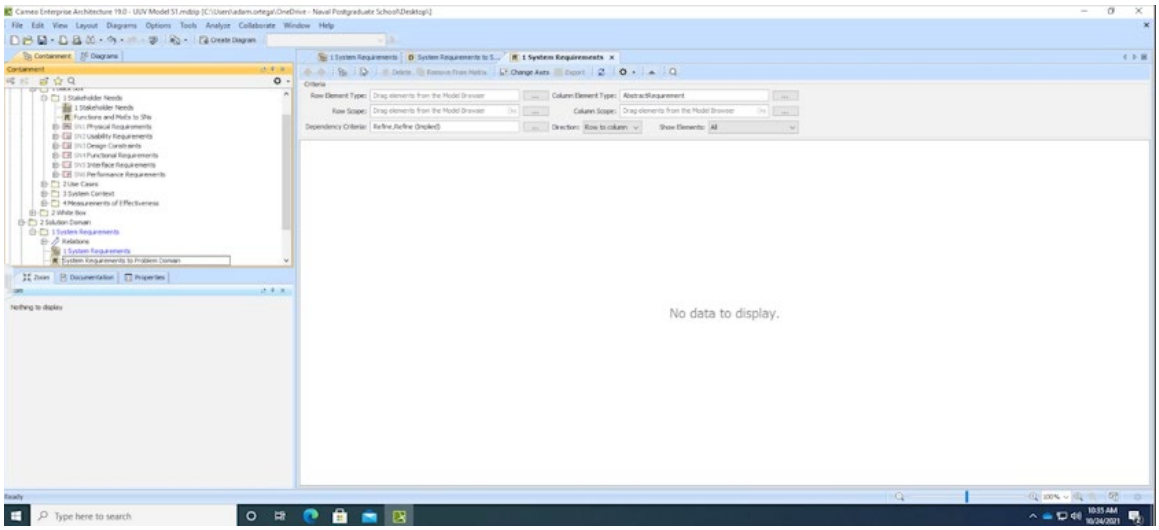
also have the *Propulsion* with *Speed* and *Positioning*. And we can continue to repeat the same process to establish the relationships between the two.



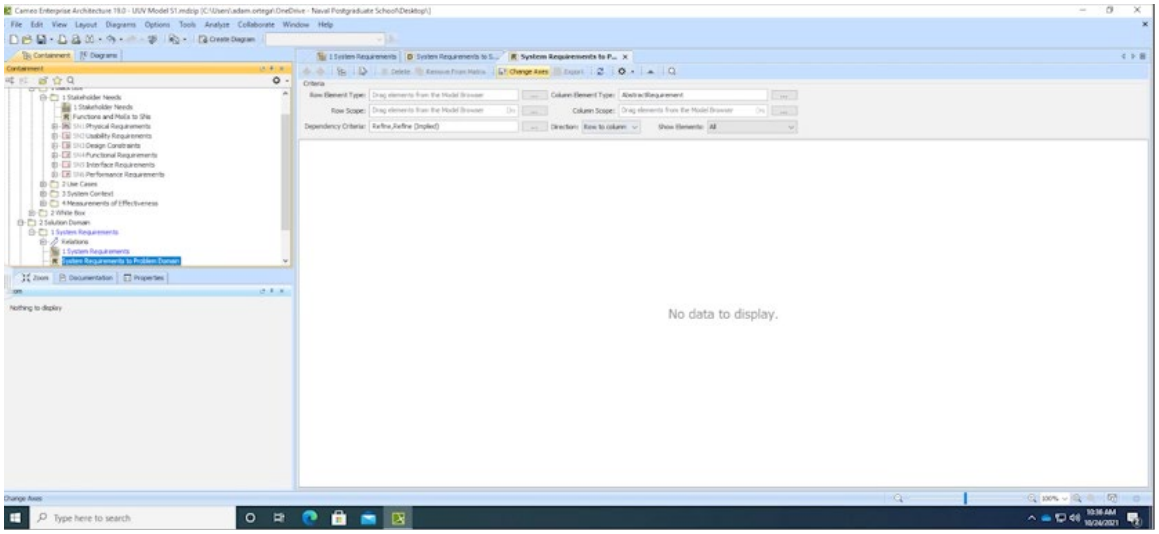
And we this is completed the final diagram that we will be creating in this tutorial will be establishing traceability to the rest of the *Problem Domain*. So in the model browser we will right click on the *System Requirement* package and from the drop-down menu we will select the *Create Diagram* option. Then a *Create Diagram* popup menu will appear and on the search bar we will type *rrm*. If nothing comes up in the search list then click on the *expert* button on the bottom left corner of the *Create Diagram* popup menu. This will the show the result and we will select the *Refine Requirement Matrix* option.



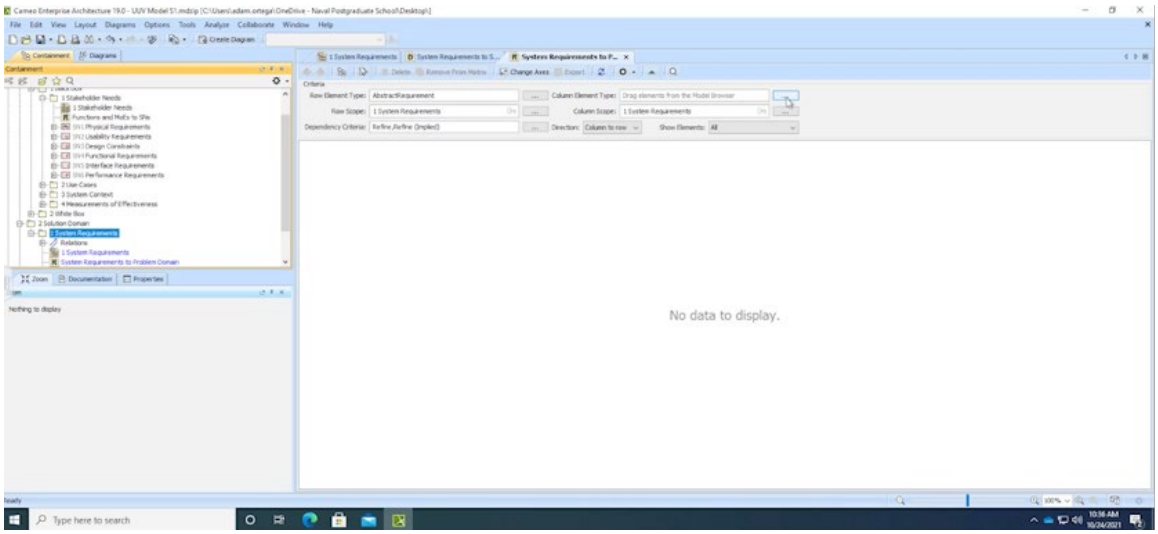
Then we will name the *Refine Requirement Matrix* as *System Requirements to Problem Domain*.



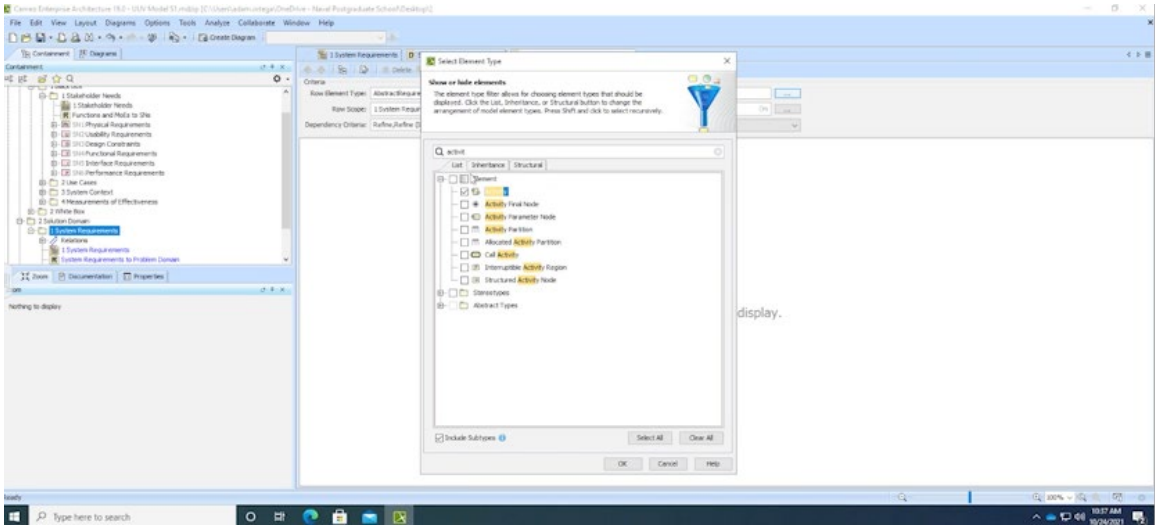
Now in the matrix toolbar we will go ahead and click on the *Change Axes* button to swap the rows and columns



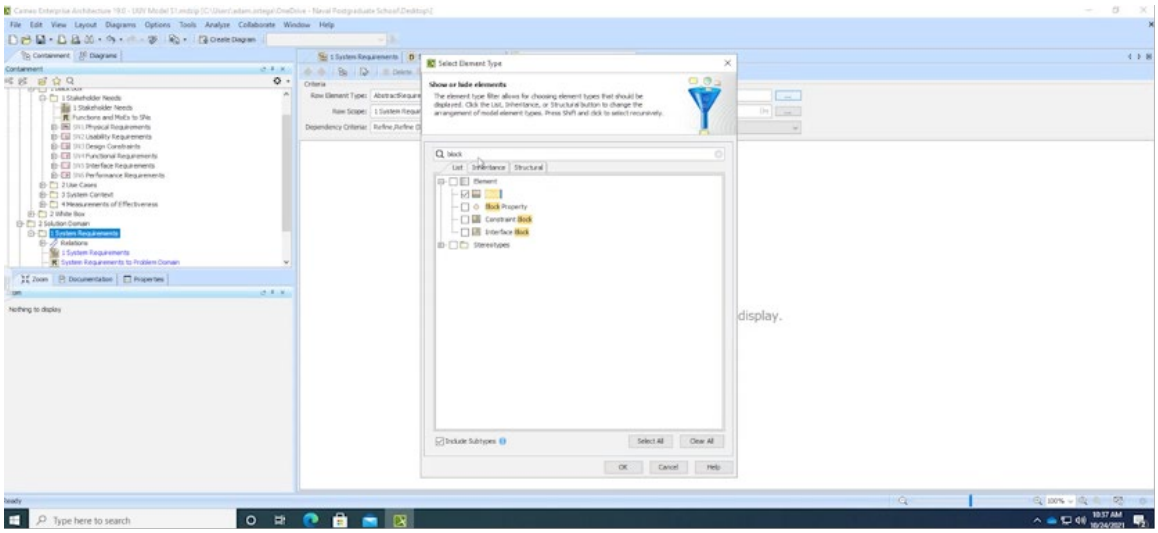
And now for the *Row Scope* we will grab the *System Requirement* package again and drop that in. And for the *Column* we need to select the *Column Element Type* by clicking on the three dot (…) button.



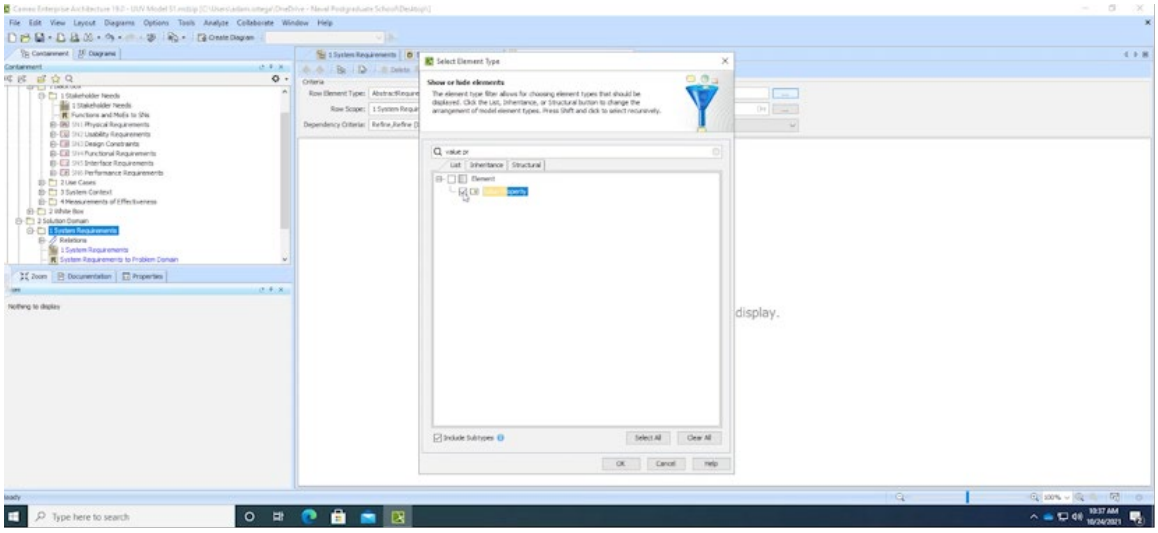
Once the button is clicked the *Select Element Type* popup window will appear. Then we will click on the *Clear All* button and then in the search bar we will type *activity* and from the result list we will select the *Activity* option.



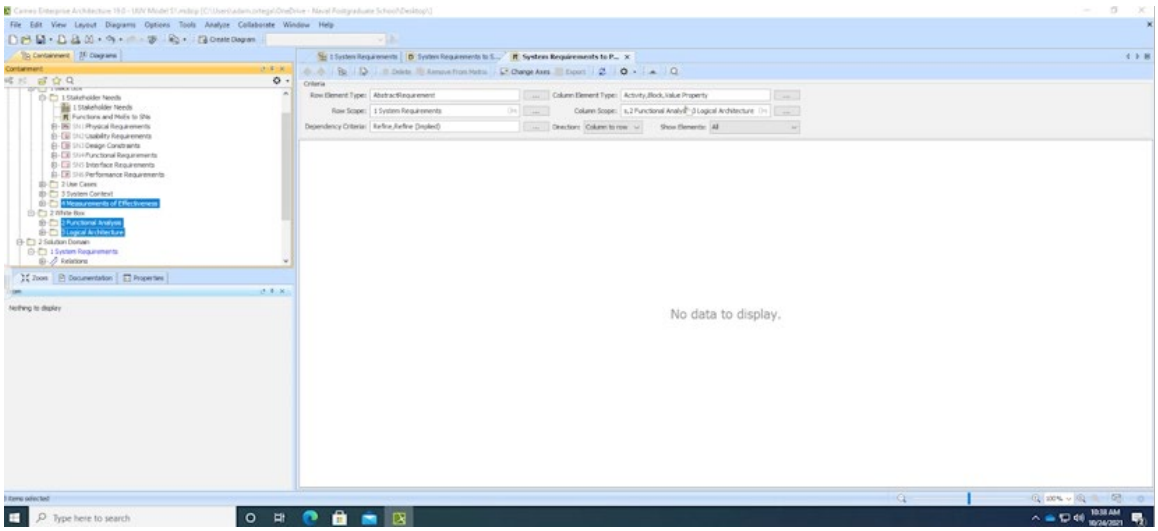
The we will type *block* in the search bar and from the result list we will select the *Block* option.



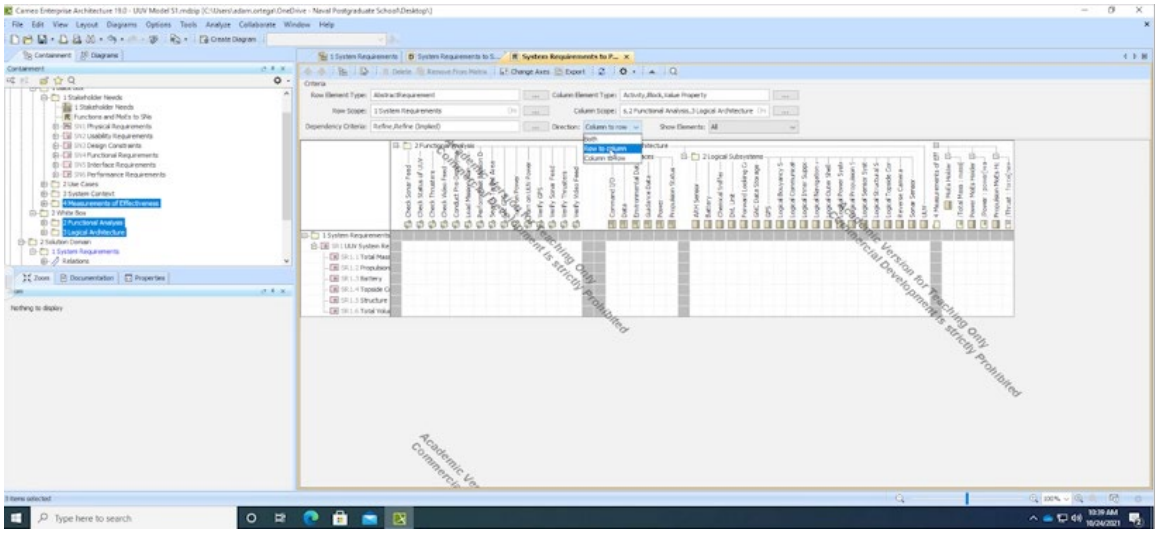
The we will type *value pr* in the search bar and from the result list we will select the *Value Property* option and then click on the *Ok* button to close down the *Select Element Type* popup window.



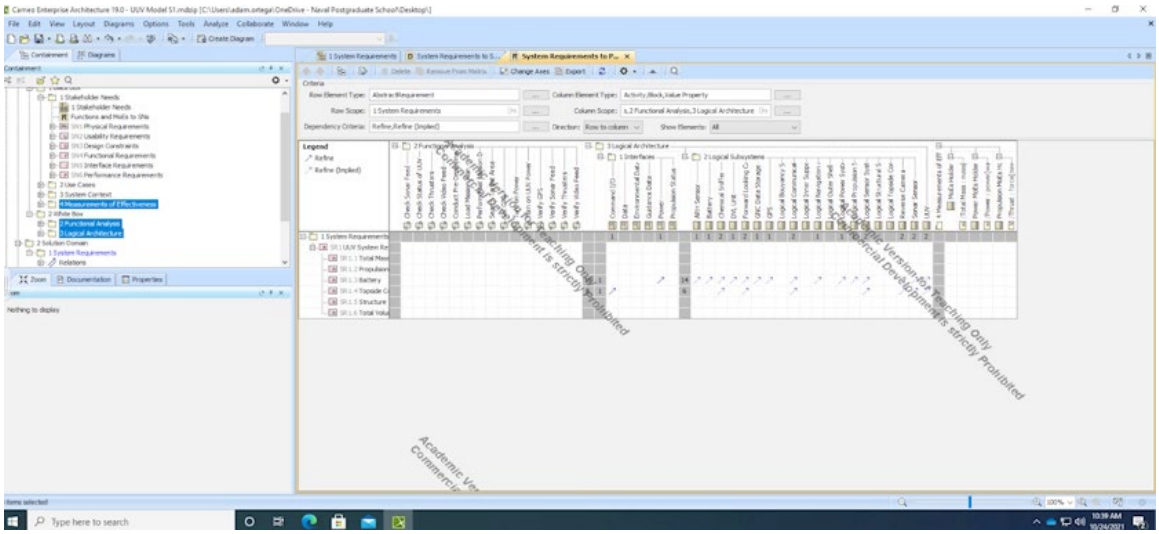
Now in the model browser we will be selecting a few items to drag to the *Column Scope*. The first will be the *4 Measure of Effectiveness* in the *Black Box*. And while holding *CTRL* we will select the *Functional Analysis* and the *Logical Architecture* in the *White Box*. We can expand the *White Box* package and select both *Functional Analysis* and the *Logical Architecture* and we will drag these to the *Column Scope*.



And we will change the direction by clicking the *Direction* drop-down menu and selecting *Row to Column* option. Then we will click the *Refresh* hyperlink to update the matrix.



And the same way in the previous matrix we can double click the intersections to establish the relationships between the two. For example, we can select and double click on the intersection between *Topside Control Station* and the *Command I/O*. We also can select and double click on the intersection between *Battery* and *Power*.



Continuing this method, we can continue to establish every relationship between the system requirements, the activities, and value properties. And this completes the tutorial for the initial system requirements. Thank you.

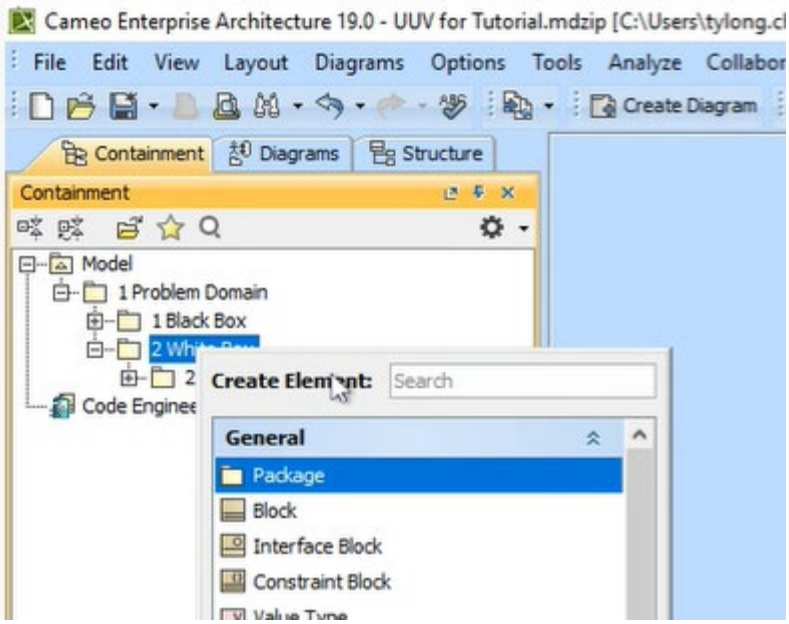
W3: Logical Subsystem Communication Tutorial

Background:

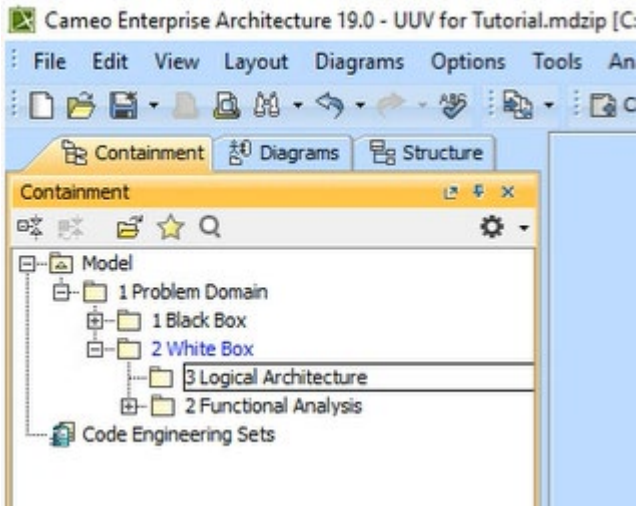
The previous section began to identify the logical subsystems while this section aims to address the interfaces and interactions between the different parts of the system of interest. A BDD is created to start to define inputs and outputs of the system using proxy ports of the system blocks. The proxy ports are typed by interface blocks and assigned flow properties that describe the kind of interface.

Step-By-Step Guide:

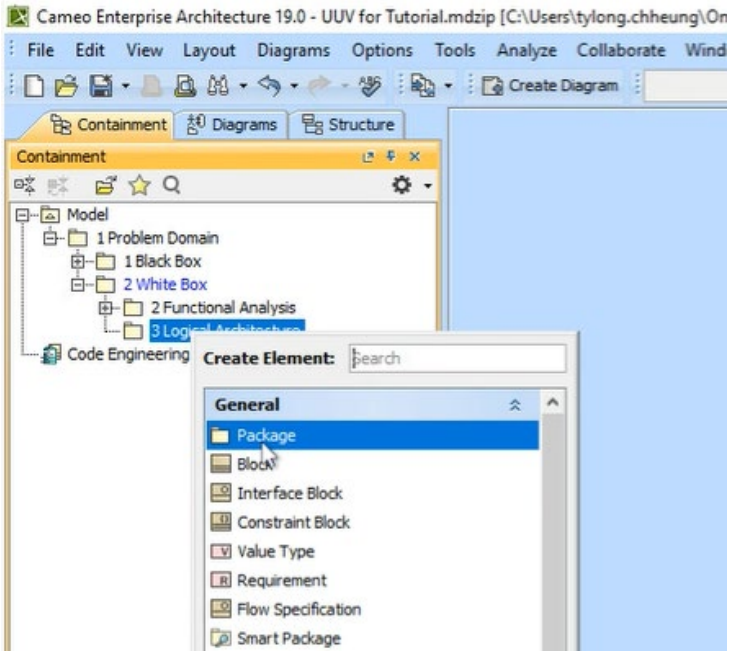
In model browser expand 1 Problem Domain, if not already. And locate >2 White Box. Right click 2 White Box package and select Create Element.



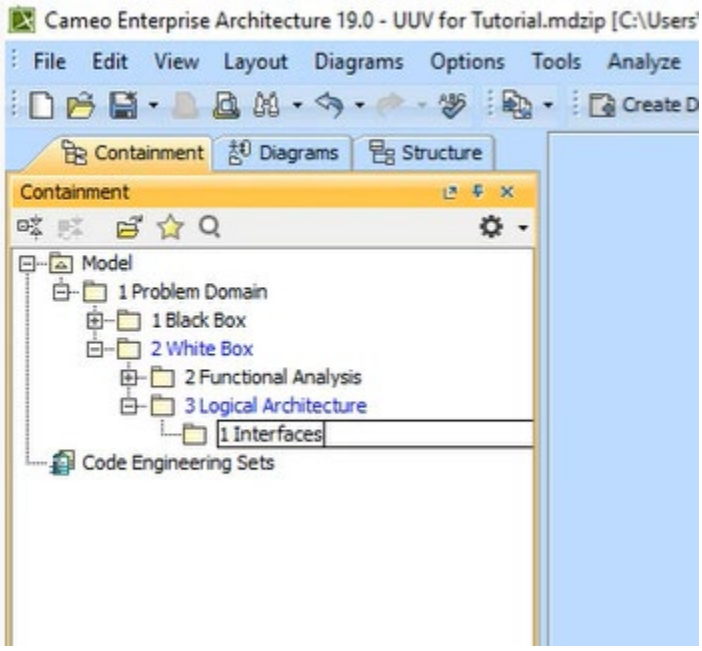
Select Package and type 3 Logical Architecture, then Enter.



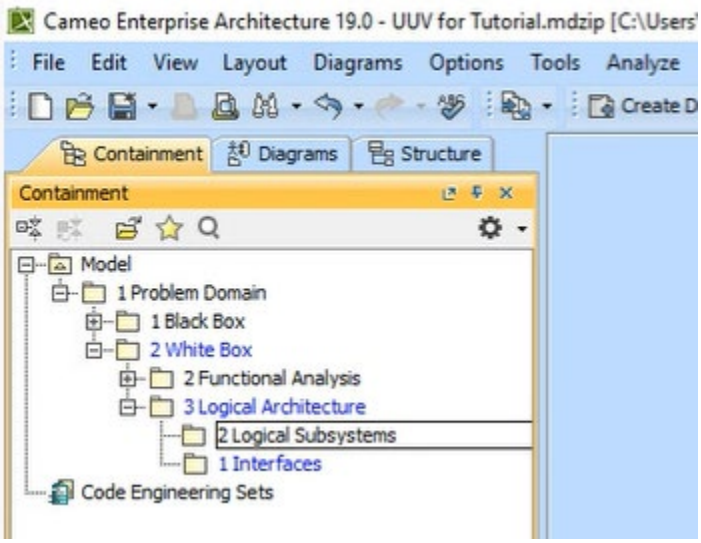
Right click 3 Logical Architecture and select Create Element.



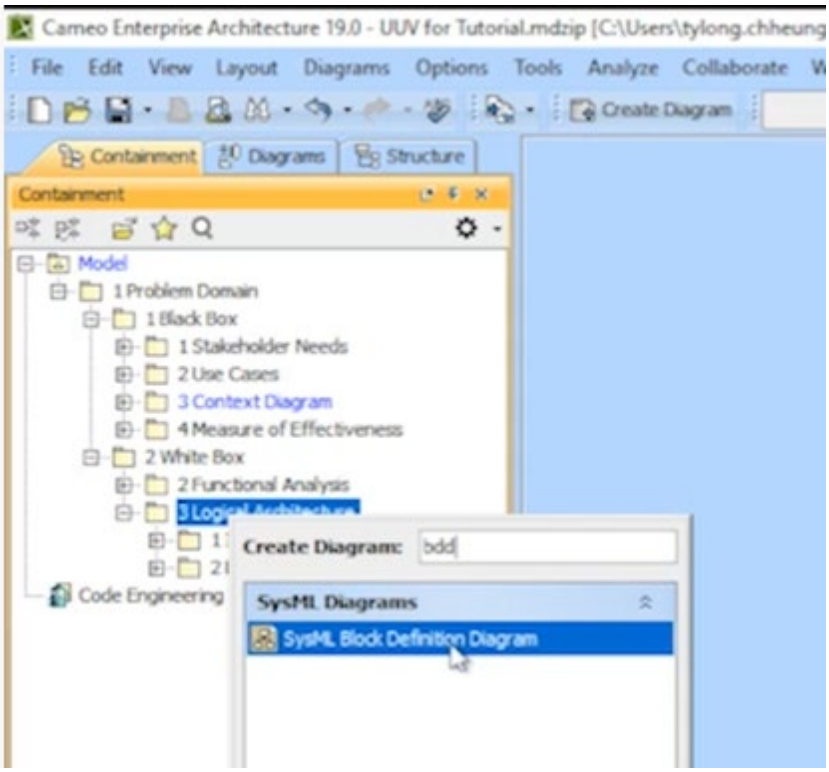
Select Package, then type 1 Interfaces, followed by Enter.

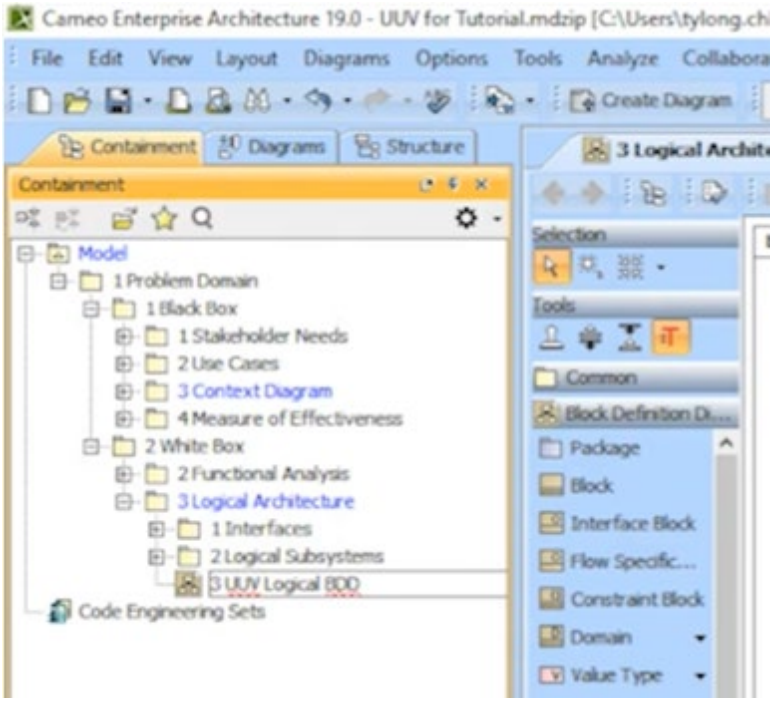


Repeat the previous steps and enter 2 logical Subsystems for the second package.

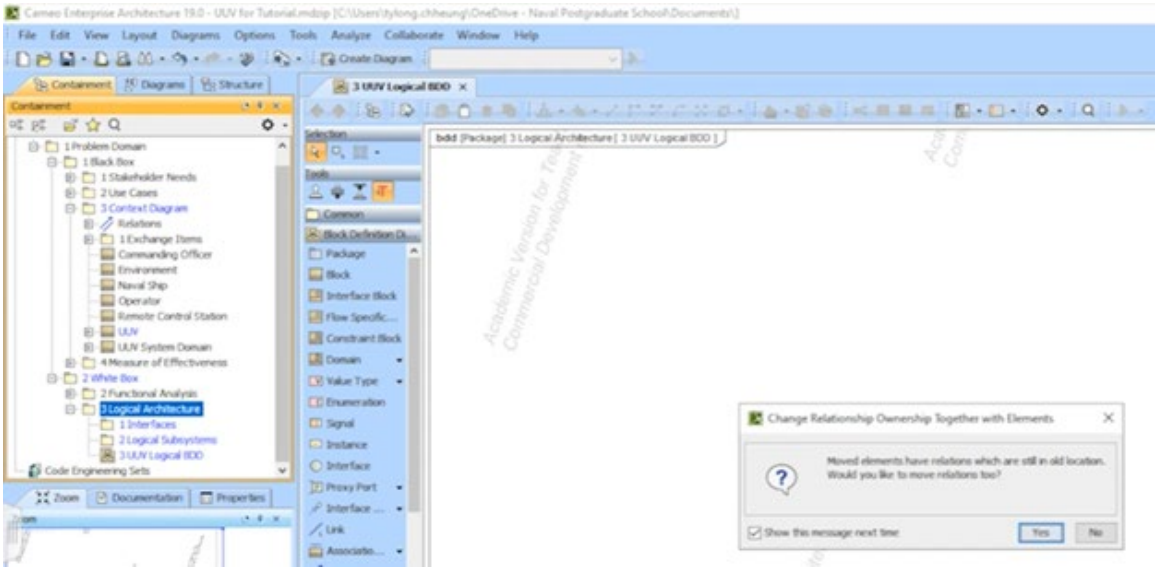


Next, we will create the Block Definition Diagram. Right click the 3 Logical Architecture and select Create Diagram. Type bdd for block definition diagram, type 3 UUV Logical BDD, press enter.

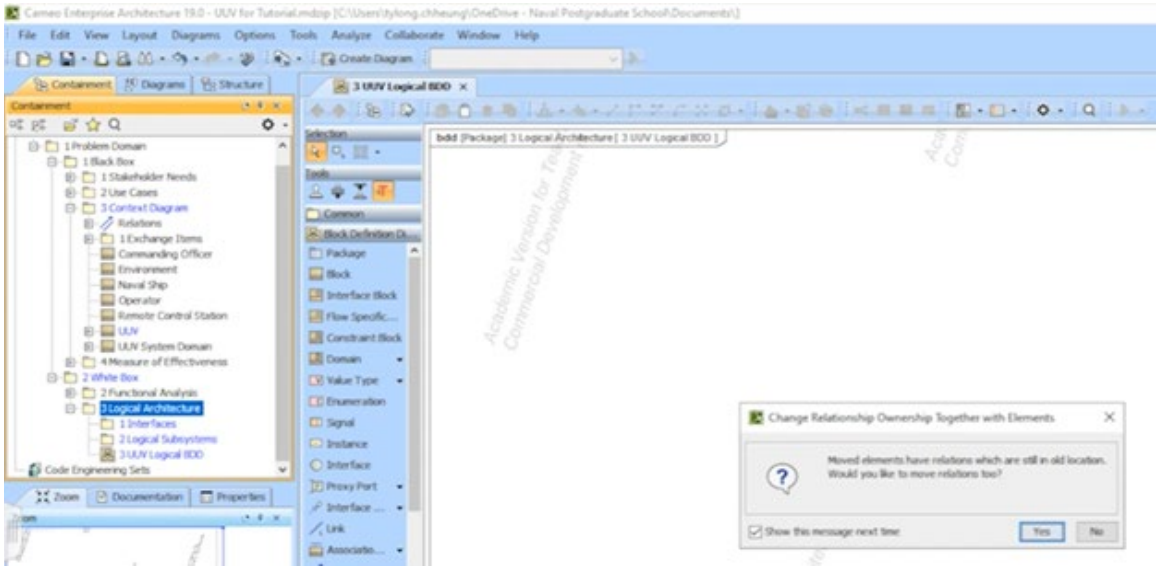




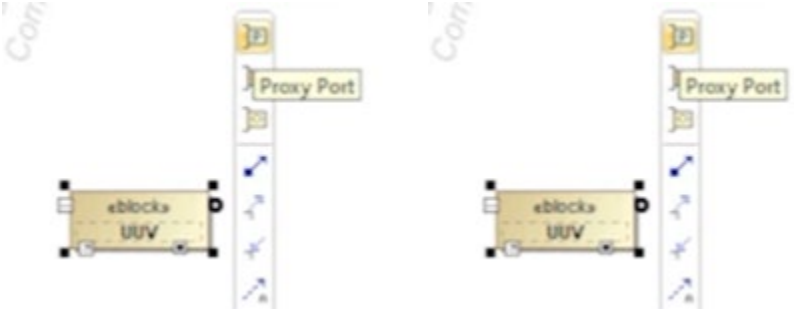
Locate the previous UUV bdd from the Problem Domain, Black Box, 3 Context Diagram select and drag to 3 logical Architecture. A pop-up window should appear, select Yes to move all relations to.



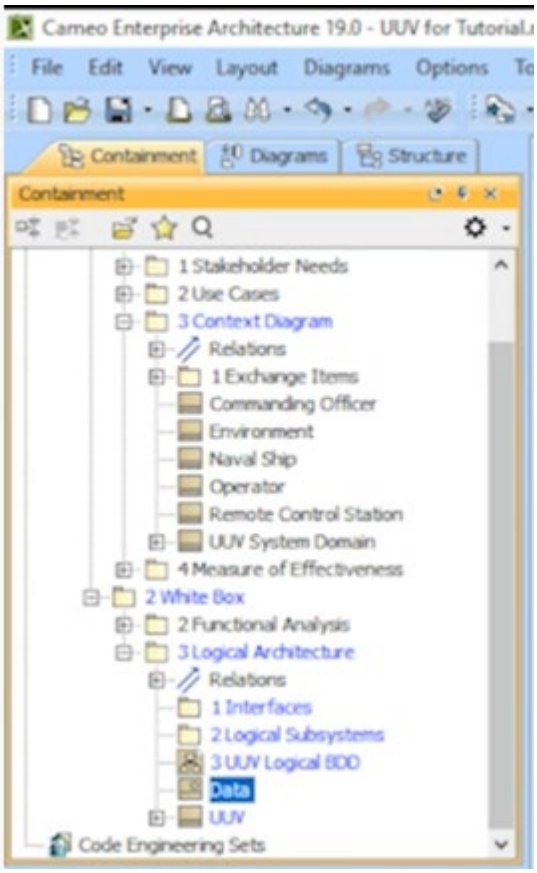
After successfully moving the bdd, select and drag the UUV onto the diagram pane for the logical bdd.



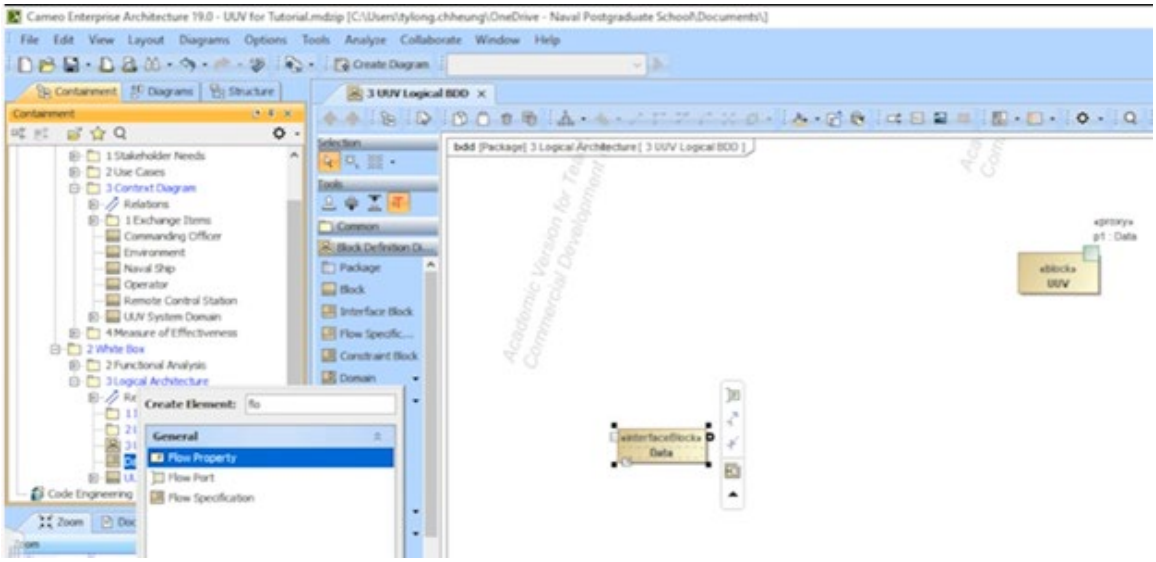
Utilizing the Smart Manipulate Tool Bar select Proxy Port, type Data then enter.



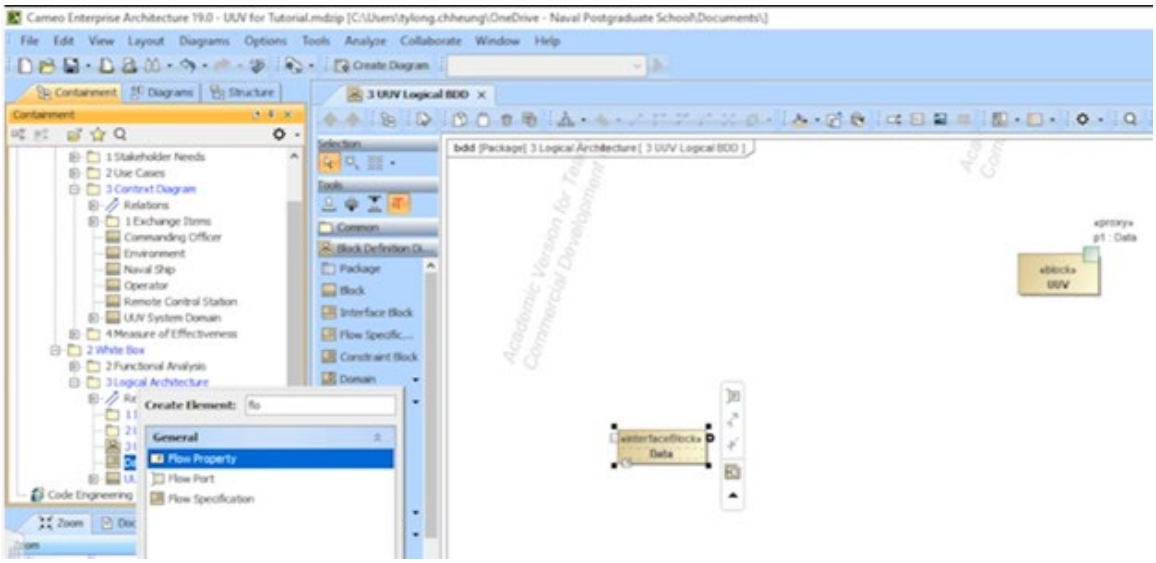
After hitting enter, Data should appear in the containment tree.



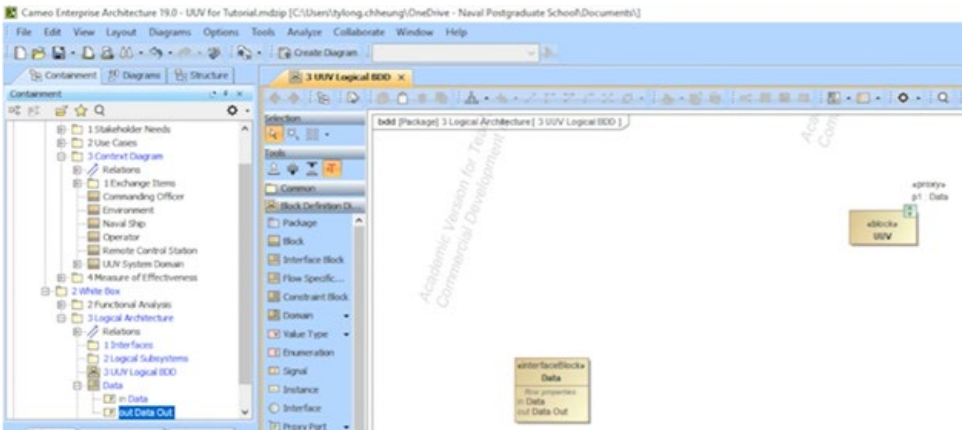
Select Data from the containment tree and drag onto the diagram pane. Right click Data from the containment tree, select Create Element, select Flow Property.



For the UUV, the Flow Property will be In and Out, type Data for the label.

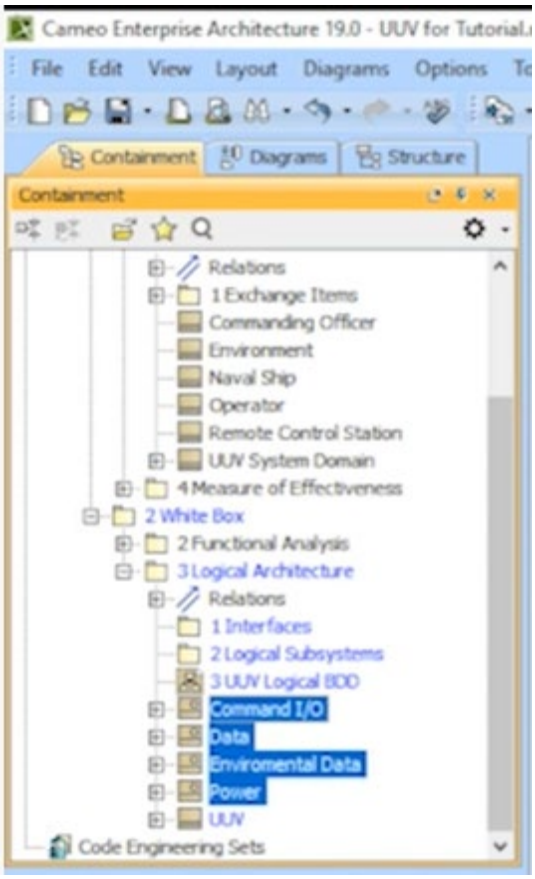


Optionally separate data Flow Properties can be created by adding an additional Flow Property by right clicking Data from the containment tree select Flow Property and label as Data Out with respect to the out prefix, the out prefix for Data must be removed.

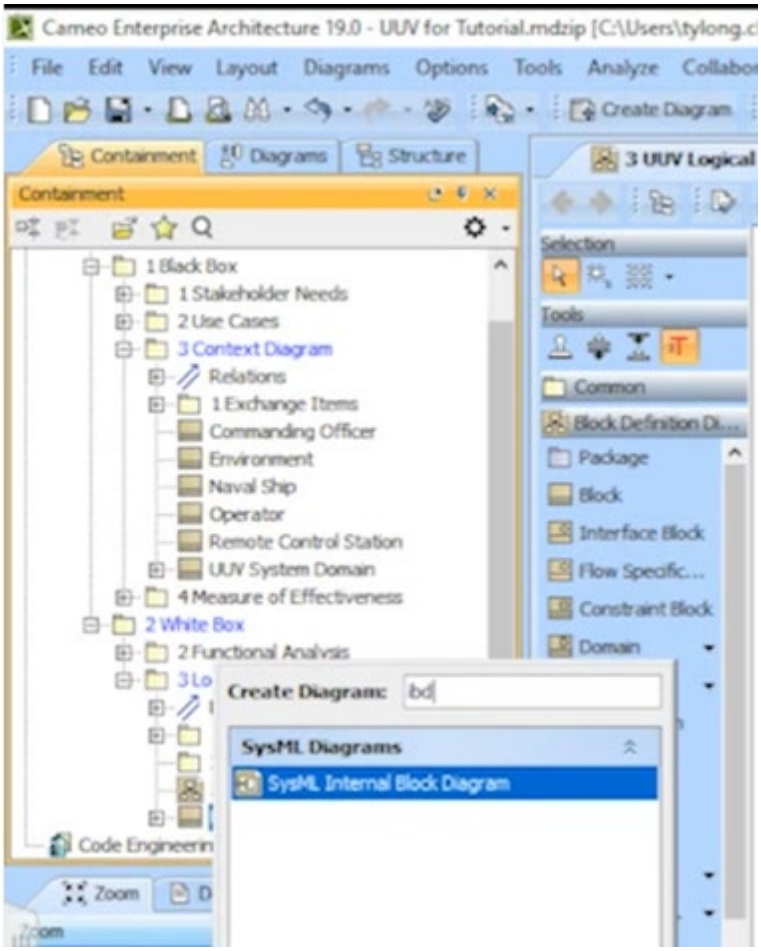


The remaining interfaces should be created following the previous few steps as desired.

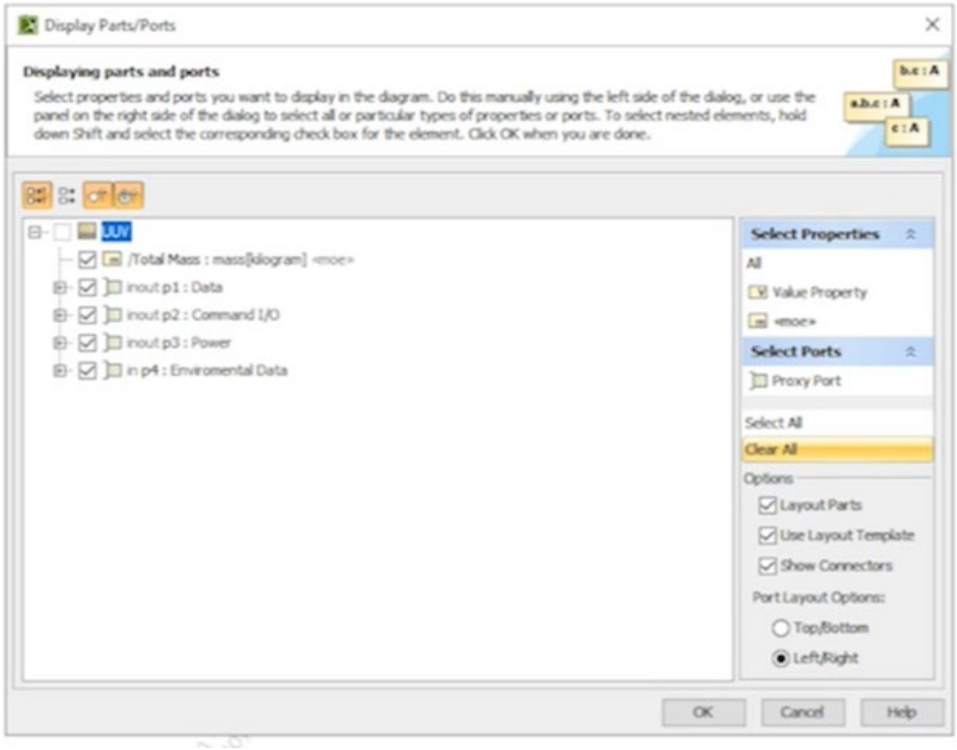
We can clean up the containment tree by selecting and highlighting the additional proxies created (Environmental Data, Power, Command input/output). Move the selected proxies into interfaces by dragging them into the Interfaces package.


Cameo Enterprise Architecture 19.0 - UUV for Tutorial
File Edit View Layout Diagrams Options
Containment Diagrams Structure
Containment
Relations
1 Exchange Items
Commanding Officer
Environment
Naval Ship
Operator
Remote Control Station
UUV System Domain
4 Measure of Effectiveness
2 White Box
2 Functional Analysis
3 Logical Architecture
Relations
1 Interfaces
2 Logical Subsystems
3 UUV Logical BDD
Command I/O
Data
Enviromental Data
Power
UUV
Code Engineering Sets

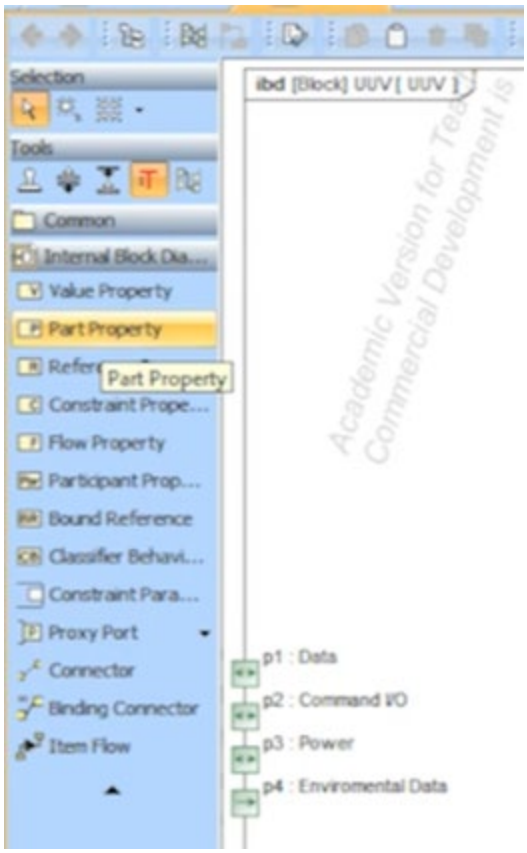
Next, we will be creating the Internal Block Diagram. Locate and right click the UVV in the Logical Architecture within the containment tree. Select Create Diagram and type IBD.

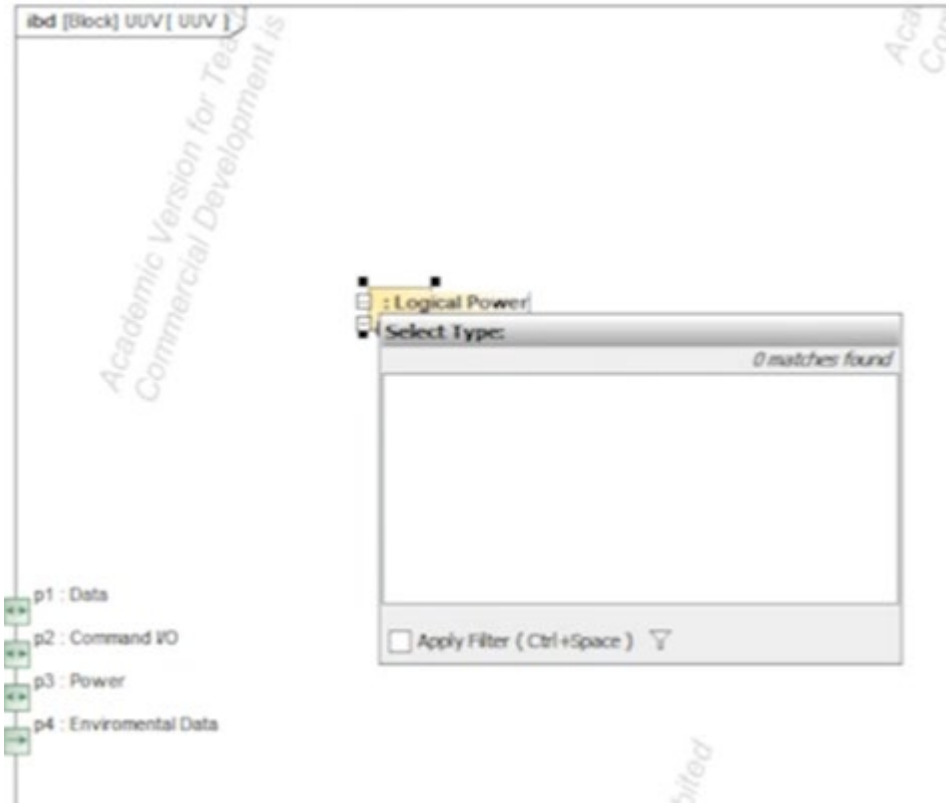


A pop-up window should appear for parts and ports. Click Clear All then Proxy and Ports (This will include all applicable proxies), click OK. This will produce the ibd in the diagram pane.

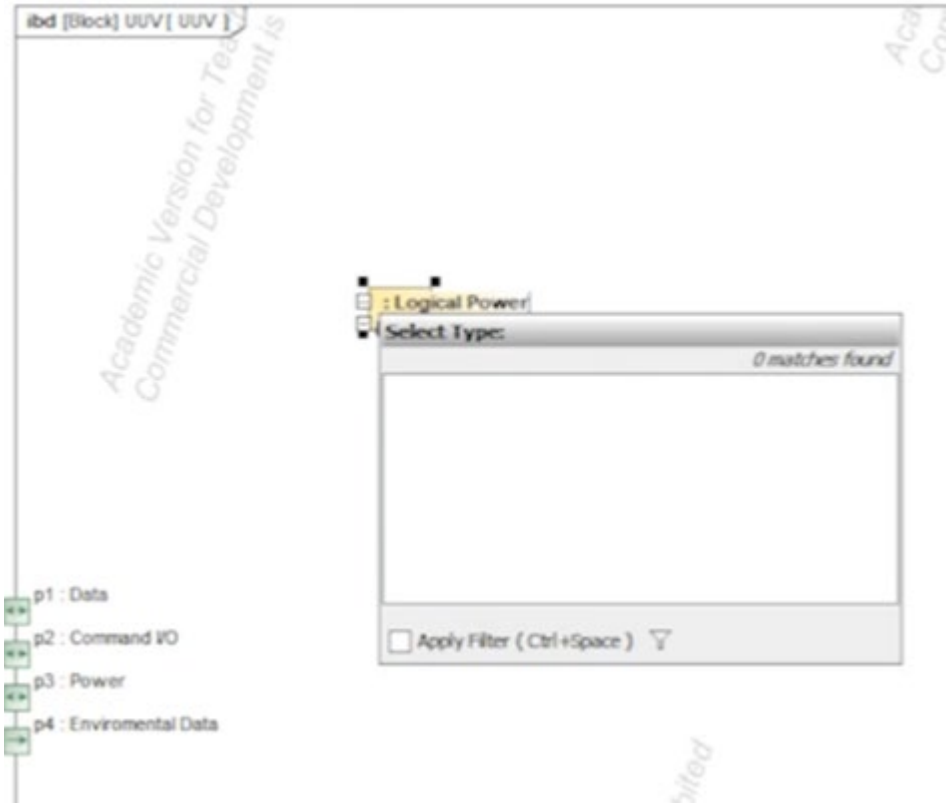


To Create Logical Subsystem, utilize the Part Property Function in the diagram palette, select and drag into the diagram pane. Label as Logical Power System, enter.


Selection
Tools
Common
Internal Block Dia...
Value Property
Part Property
Part Property
Constraint Prope...
Flow Property
Participant Prop...
Bound Reference
Classifier Behavi...
Constraint Para...
Proxy Port
Connector
Binding Connector
Item Flow
ibd [Block] UUV [UUV]
p1 : Data
p2 : Command I/O
p3 : Power
p4 : Enviromental Data


ibd [Block] UUV [UUV]
: Logical Power
Select Type:
0 matches found
Apply Filter (Ctrl+Space)
p1 : Data
p2 : Command I/O
p3 : Power
p4 : Enviromental Data

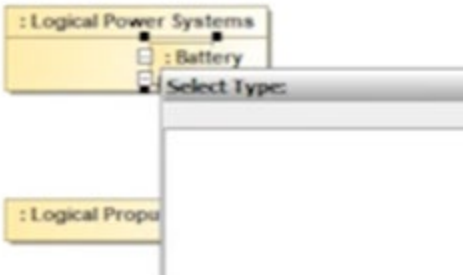
Repeat the previous steps for the following part properties (Logical Propulsion System, Logical Structure System, Logical Navigation and Positioning unit, Logical Communication System and Logical Sensor System)



Creating additional decomposed Subsystem's

Select Part Property Function then select Logical Power System.

Type Battery into the field below the logical Power System, this will create the subsystem for the Logical Power System.

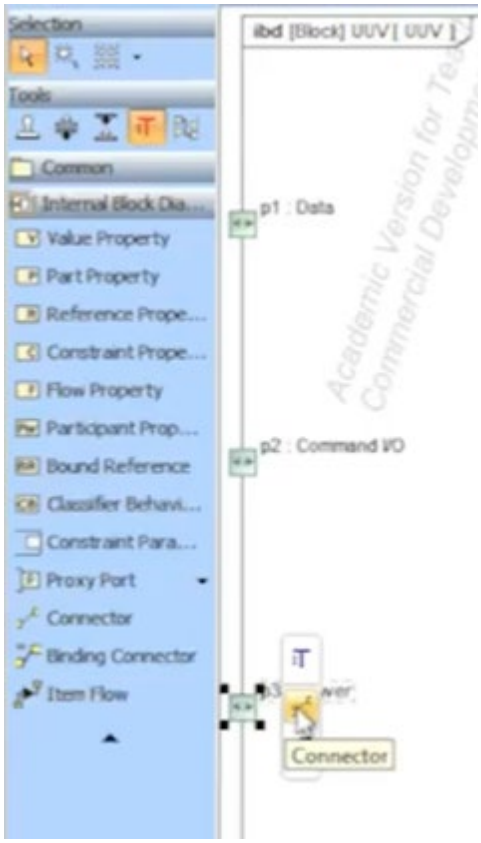


Repeat the previous steps for each additional subsystem respectively.

Interface Logical Subsystems to the IBD of the UUV

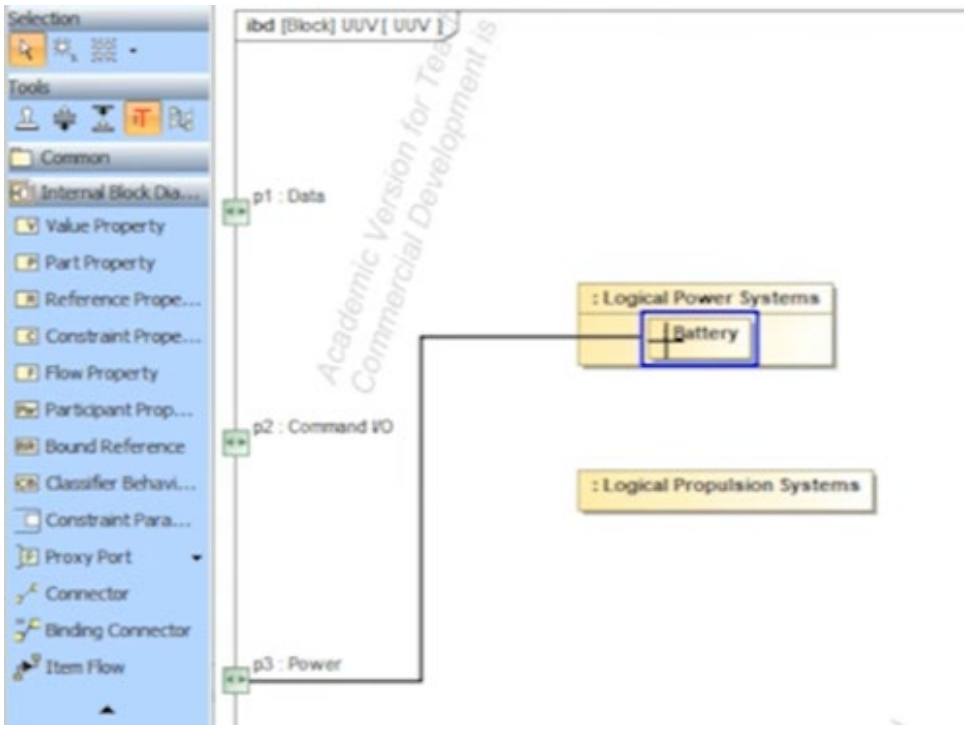
Click on Power, the smart manipulator tool bar should appear.

Select Connector Function.

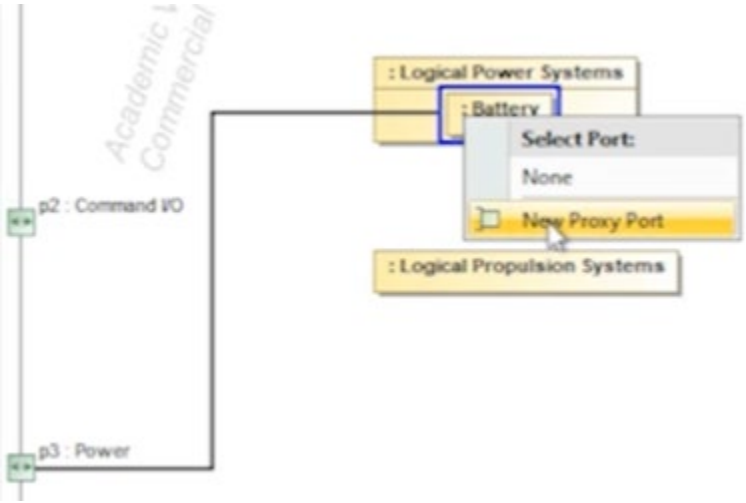


With the Connector Function active a trace line will appear and follow the cursor.

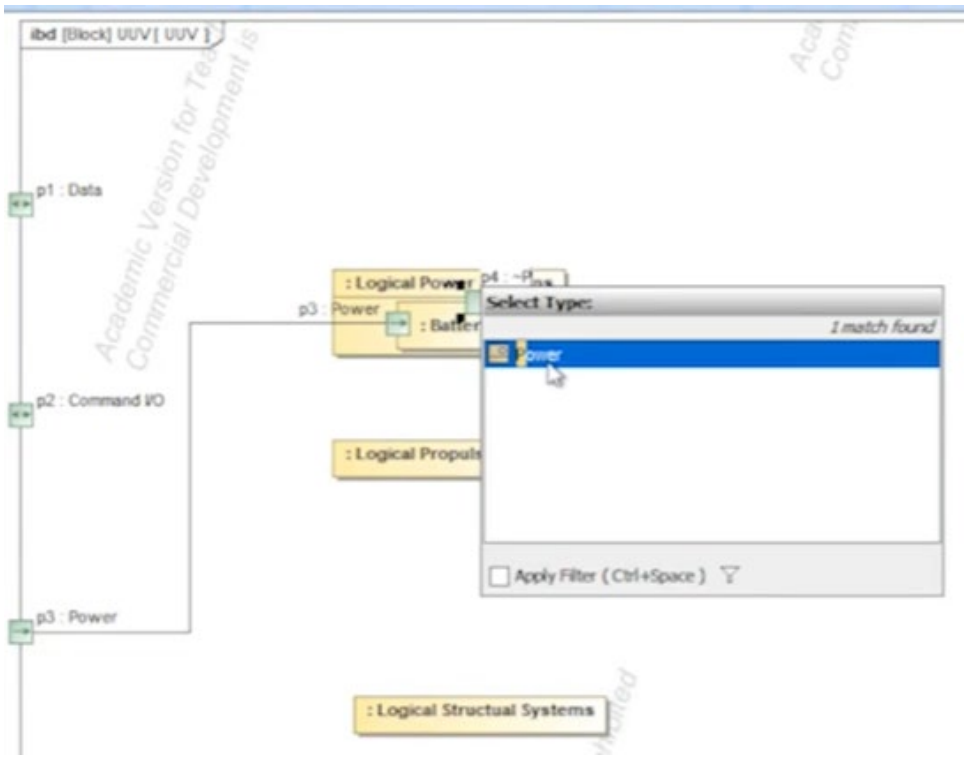
Move cursor over the Battery subsystem then click, a new window should appear.



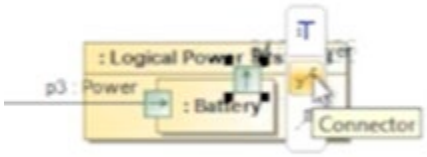
Select New Proxy Port function, this will create a new proxy input Power for the Battery subsystem.



Click the Battery subsystem, using the Tool bar select Proxy Port then create a tilda and label it as Power. This will be used to create the output.



Use the Smart Manipulator Tool bar select Connector Function.

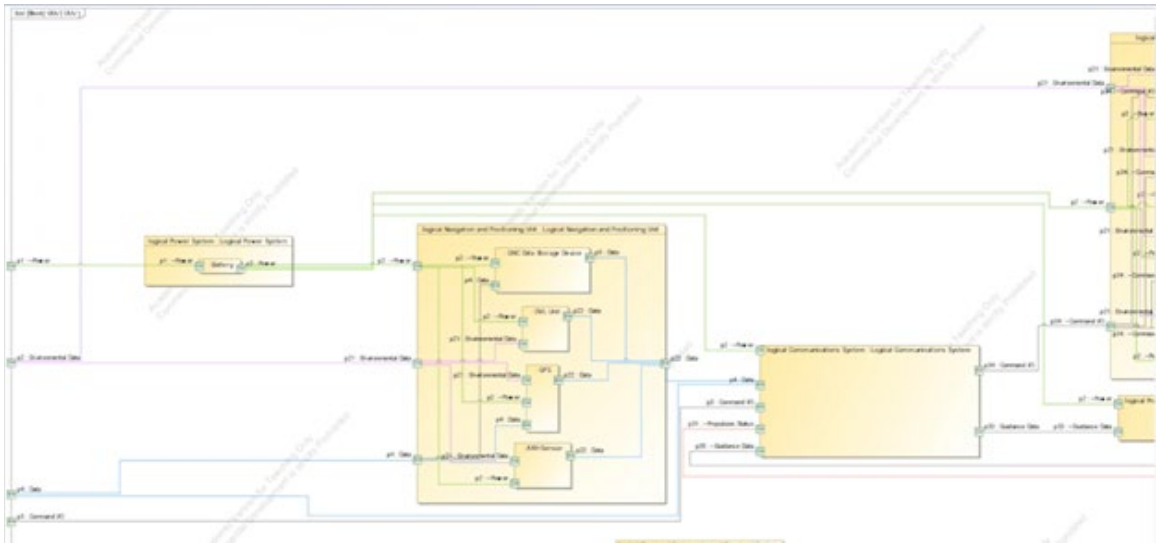


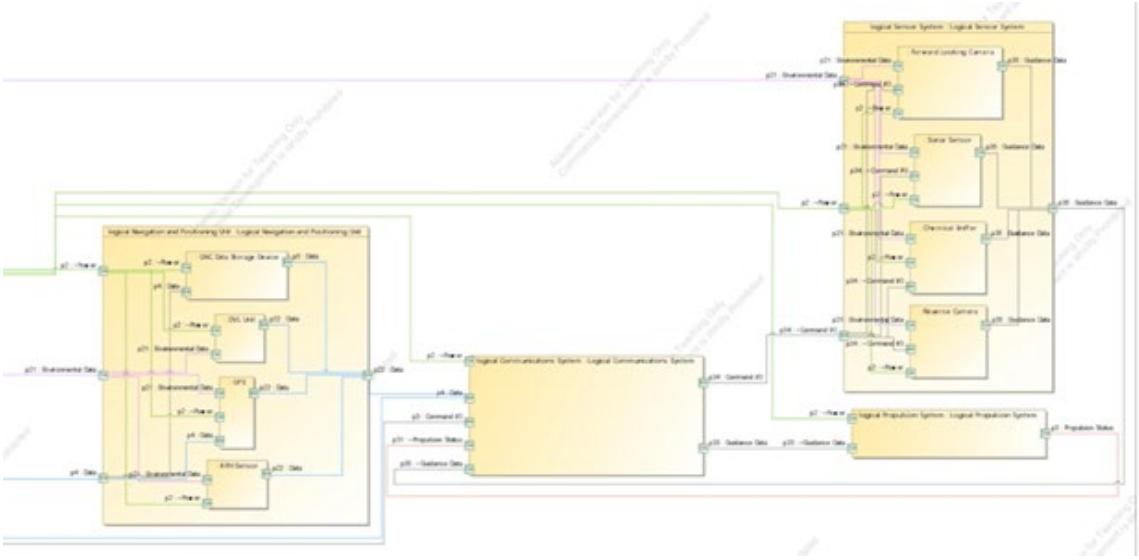
With the Connector Function active a trace line will appear and follow the cursor.

Move cursor over the logical Communication System then click, a new window should appear.

Select New Proxy Port function, this will create a new proxy input Power from the Battery subsystem to the Logical Communication System.

Utilizing the last few steps, the connect should be applied to each Logical Subsystem respectively.





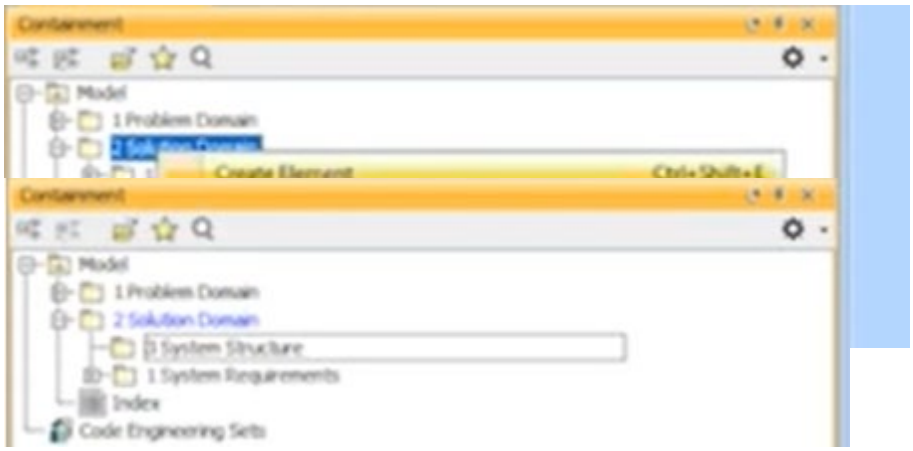
S3: Initial System Structure Tutorial

Background:

The initial system structure can be developed by using a SysML BDD to design a High-Level Solution Architecture (HLSA) which specifies all the subsystems of the system and their interfaces. This will specify how the subsystems interact and integrate as a whole system. In a standard setting, the HLSA helps to identify the different work packages of the subsystems and allocates to the respective engineering groups. Once the initial system structure is designed, specifics of the subsystems can be drawn out in more detail and integrated to a final solution architecture.

Step-By-Step Guide:

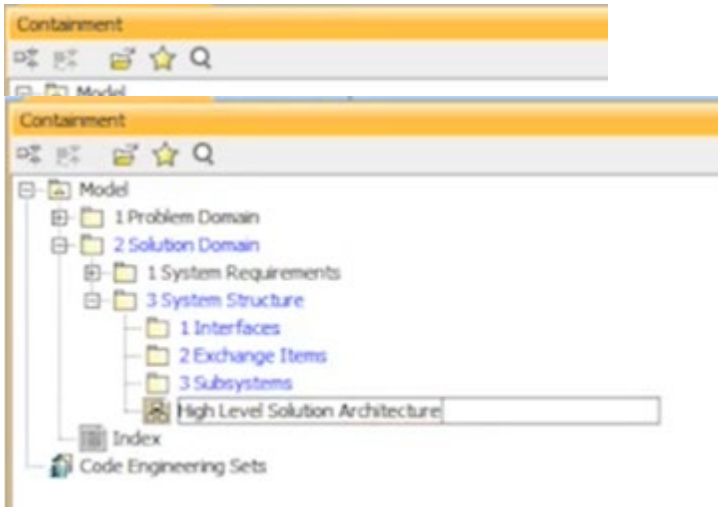
First, we start off by organizing the model for S3. Right-click the 2 Solution Domain package and select Create Element.



In the search box, type pa, the first two letters of the element type Package, and press Enter. Type 3 System Structure to specify the name of the new package and press Enter.

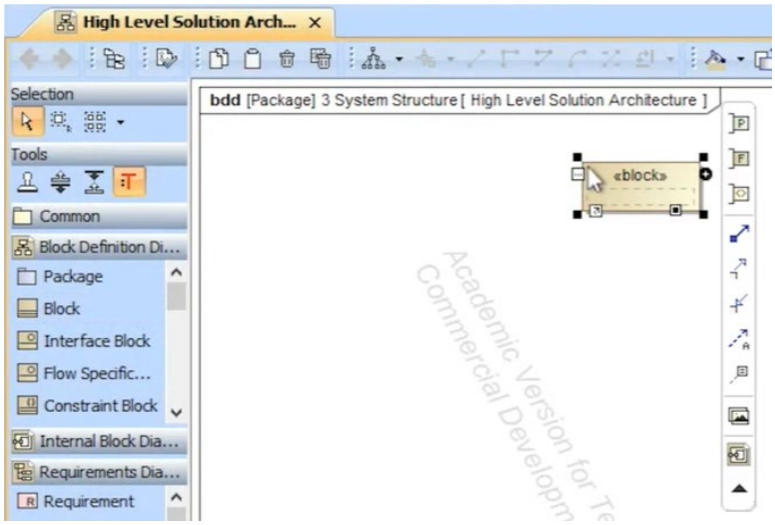
Next, we have to create three other sub packages for the System Structure Package. The Interfaces, the Exchange Items, and the Subsystems. Right-click the 3 System Structure package and select Create Element. In the search box, type pa, the first two letters of the element type Package, and press Enter. Type 1 Interfaces to specify the name of the new package and press Enter. Follow the previous steps to create the other 2 sub packages.

We will be creating a BDD for capture the HLSA. Right-click the 3 System Structure package and select Create Diagram. In the search box, type bdd, the acronym for

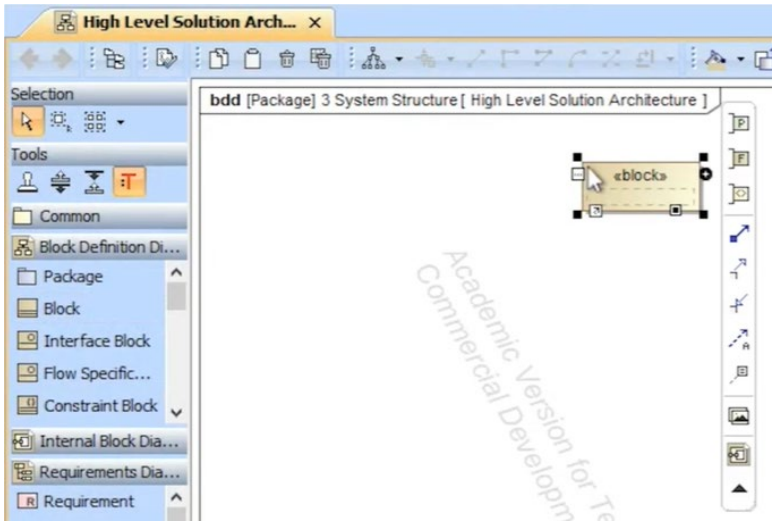


SysML block definition diagram, and then press Enter. The diagram is created. Type High-Level Solution Architecture to specify the name of the new diagram and press Enter again.

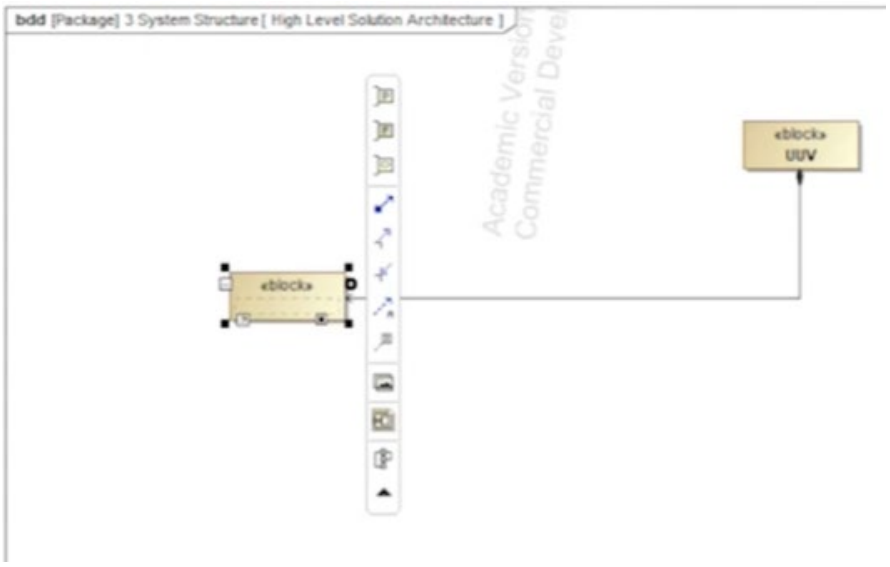
To capture subsystems of the system under design at HLSA, start by clicking the Block button on the diagram palette and then click on the diagram pane.

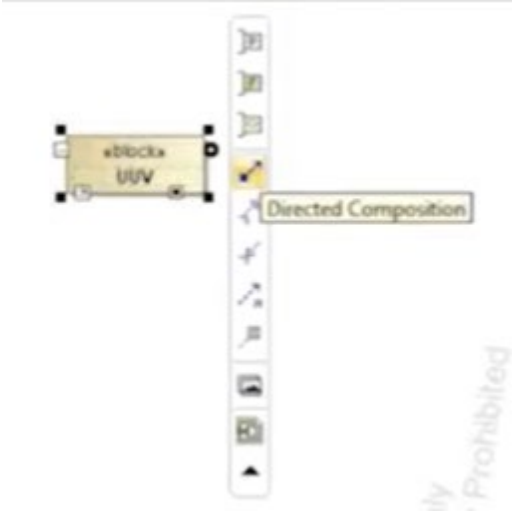


Type UUV to the name the block and press Enter.

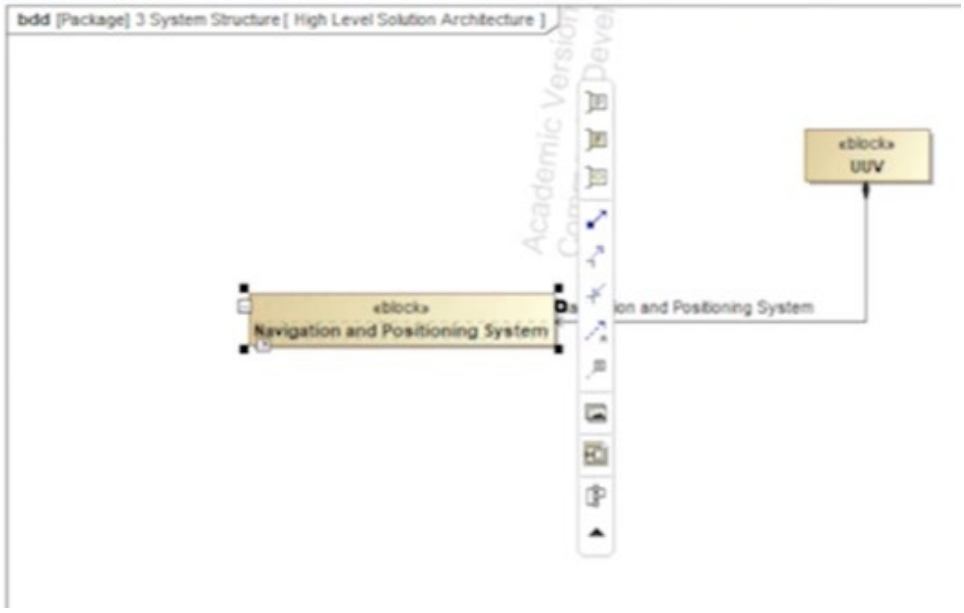


Click the Direct Composition button on the smart manipulator toolbar of the UUV block and then click an empty place on the diagram pane.

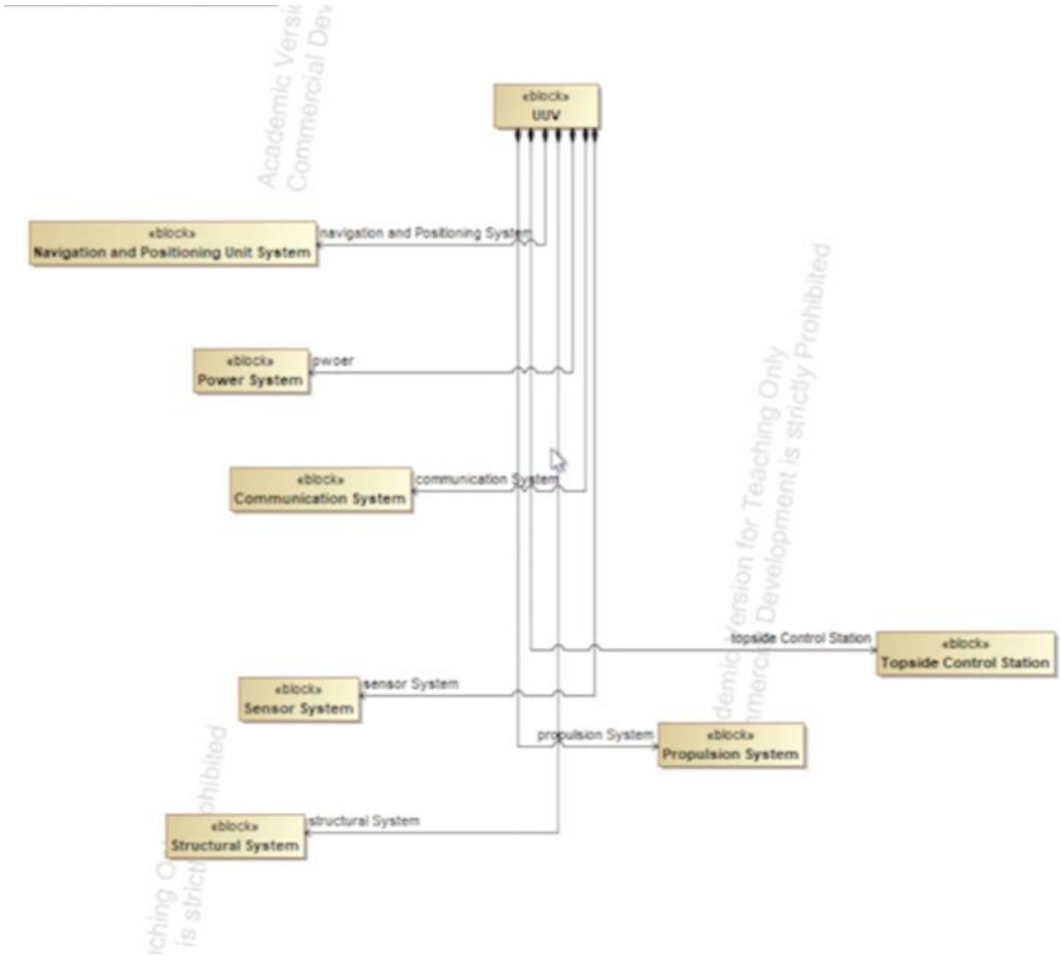




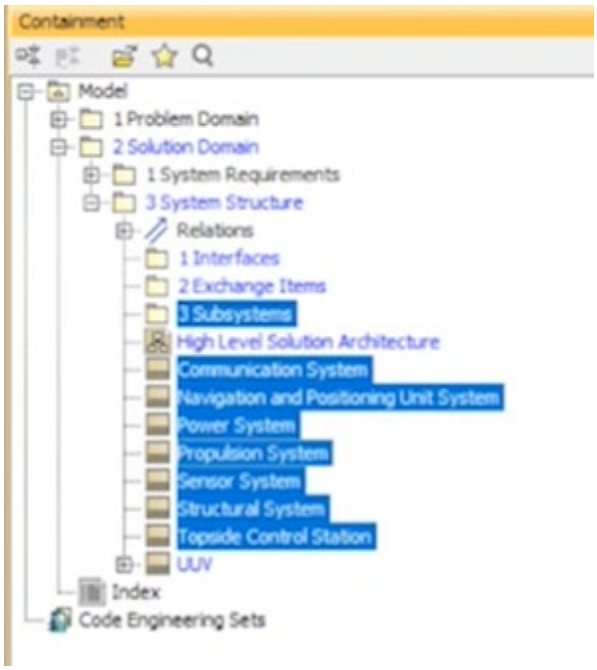
Type Navigation and Positioning System to name the block and press Enter.

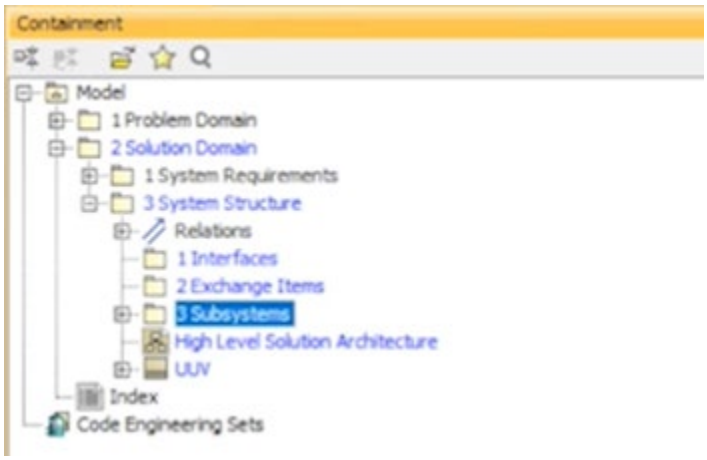


Repeat the previous steps to capture other subsystems: Power System, Communication System, Sensor System, Structural System, Propulsion System, and Topside Control System


«block»
UUV
«block»
Navigation and Positioning Unit System
navigation and Positioning System
«block»
Power System
pwoer
«block»
Communication System
communication System
topside Control Station
«block»
Topside Control Station
«block»
Sensor System
sensor System
propulsion System
«block»
Propulsion System
«block»
Structural System
structural System

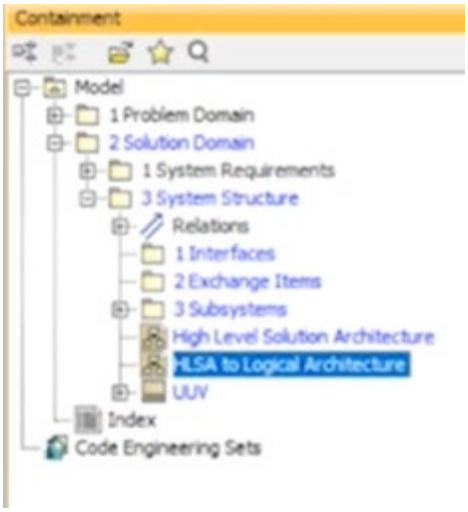
In the Model Browser, select all the blocks of subsystems and drag them to the 3 Subsystems package you created. The blocks are moved to the 3 Subsystems package.



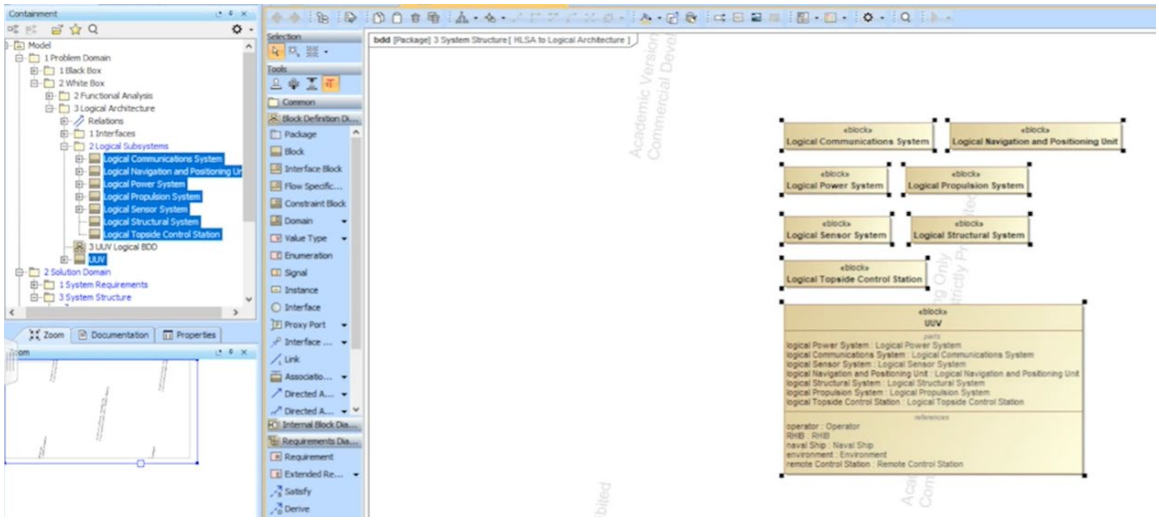


In order to establish traceability to W3, we have to create a separate BDD that displays structure blocks from both domains. Right-click the 3 System Structure package and select Create Diagram. In the search box, type BDD, the acronym for SysML block definition diagram, and then press Enter. The diagram is created. Type HLSA to Logical Architecture to specify the name of the new diagram and press Enter again.

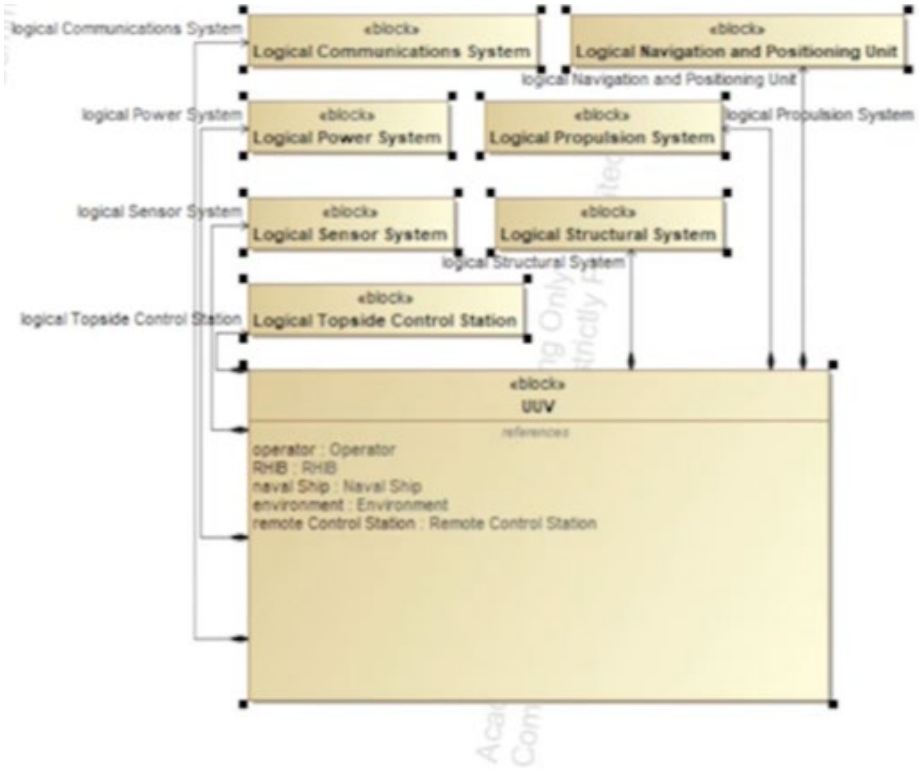
In the Model Browser, expand the contents of packages in the following sequence: 1 Problem Domain > 2 White Box > 3 Logical Architecture > 2 Logical Subsystems. Select



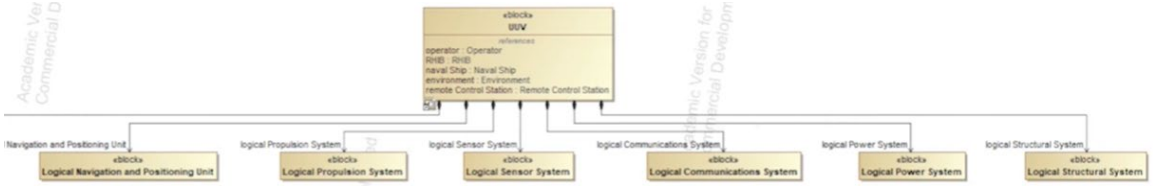
UUV and all of its Subsystem in the Subsystem Package. Drag the selection to the diagram pane. The shapes of the selected blocks are displayed on the diagram.



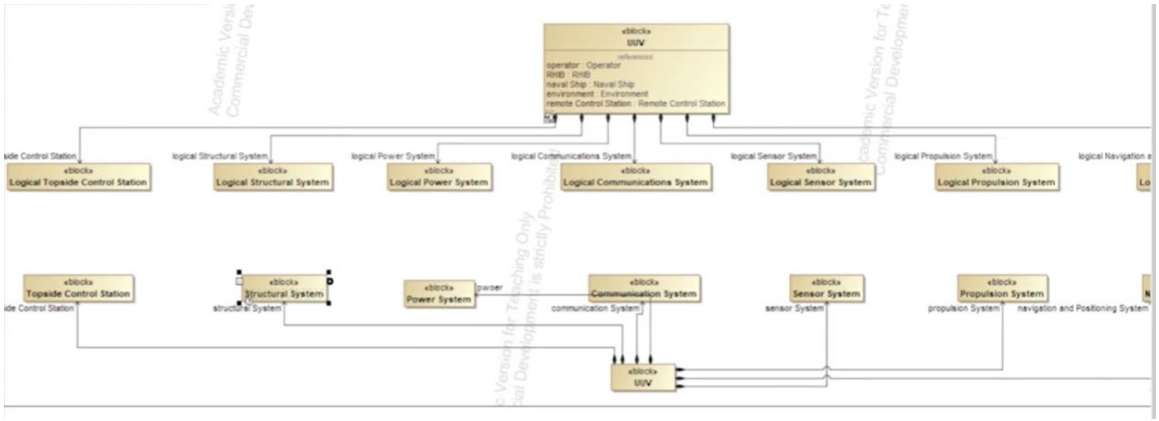
Right-click these shapes and select Display > Display Paths. Confirm that you want to display paths between selected symbols only. The paths of direct composition relationships are displayed between the shapes.



Click an empty space on the diagram pane to unselect the shapes. On the diagram toolbar, click the Quick Diagram Layout button.

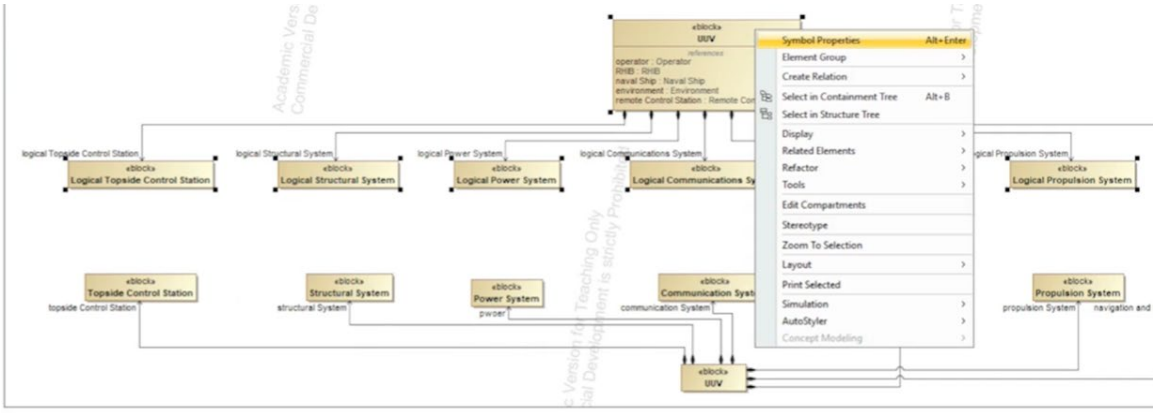


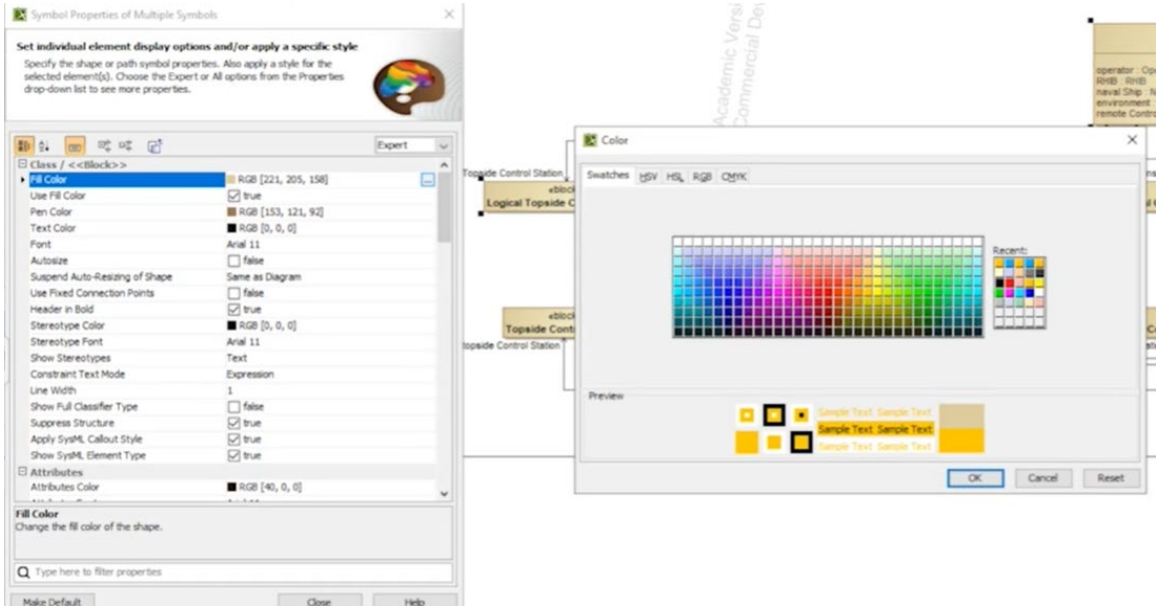
In the Model Browser, select all the blocks within the 3 Subsystems package for

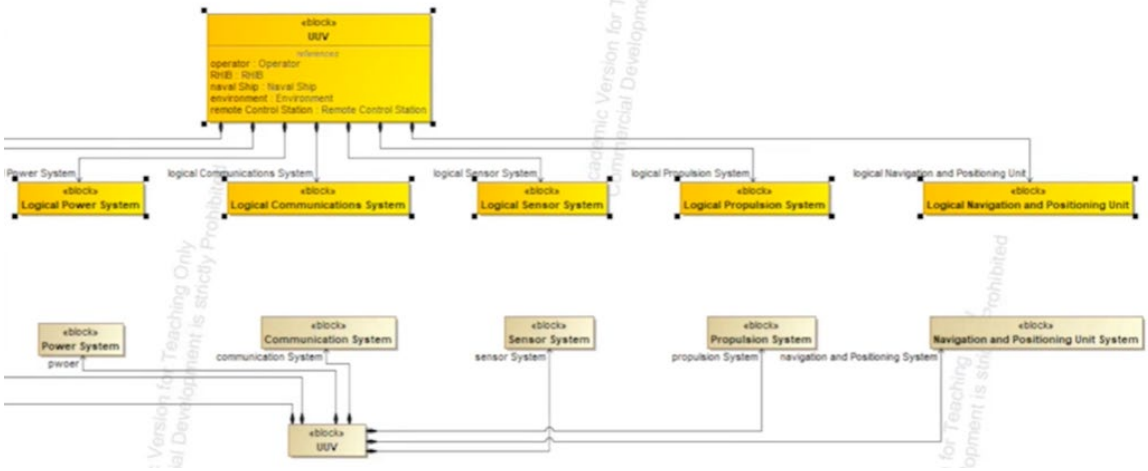


the solution domain and drag them onto the diagram pane to display them in a mirrored formation.

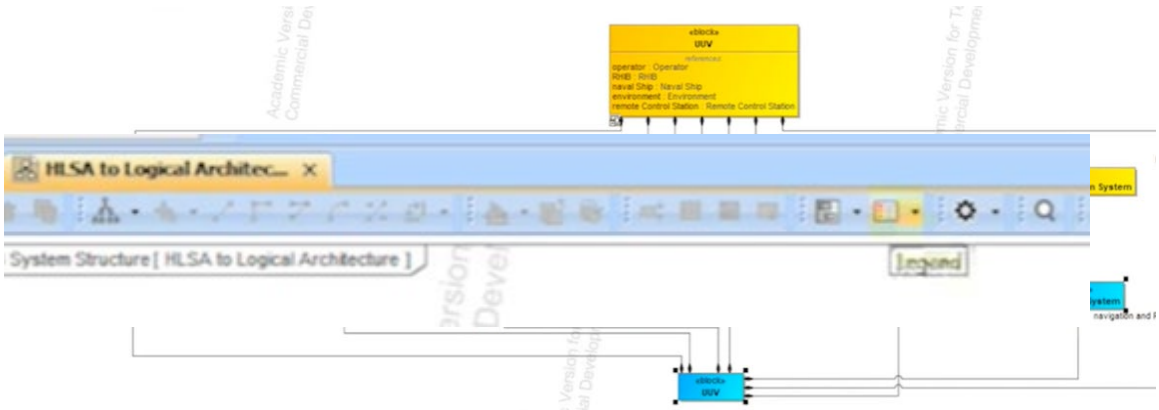
Select all system blocks from the 1 Problem Domain by holding the shift key and clicking each block. Right click to open the symbol properties and select a different color of your choice using the Fill Color option.


«block»
UUV
Symbol Properties Alt+Enter
Element Group
Create Relation
Select in Containment Tree Alt+B
Select in Structure Tree
Display
Related Elements
Refactor
Tools
Edit Compartments
Stereotype
Zoom To Selection
Layout
Print Selected
Simulation
AutoStyler
Concept Modeling
«block»
Logical Topside Control Station
«block»
Logical Structural System
«block»
Logical Power System
«block»
Logical Communications System
«block»
Logical Propulsion System
«block»
Topside Control Station
«block»
Structural System
«block»
Power System
«block»
Propulsion System
«block»
UUV


Symbol Properties of Multiple Symbols
Set individual element display options and/or apply a specific style
Specify the shape or path symbol properties. Also apply a style for the selected element(s). Choose the Expert or All options from the Properties drop-down list to see more properties.
Expert
Class / <<Block>>
Fill Color RGB [221, 205, 158]
Use Fill Color true
Pen Color RGB [153, 121, 92]
Text Color RGB [0, 0, 0]
Font Arial 11
Autosize false
Suspend Auto-Resizing of Shape Same as Diagram
Use Fixed Connection Points false
Header in Bold true
Stereotype Color RGB [0, 0, 0]
Stereotype Font Arial 11
Show Stereotypes Text
Constraint Text Mode Expression
Line Width 1
Show Full Classifier Type false
Suppress Structure true
Apply SysML Callout Style true
Show SysML Element Type true
Attributes
Attributes Color RGB [40, 0, 0]
Fill Color
Change the fill color of the shape.
Type here to filter properties
Make Default
Close
Help
Color
Swatches
HSV
HSL
RGB
CMYK
Recent:
Preview
Sample Text Sample Text
OK
Cancel
Reset

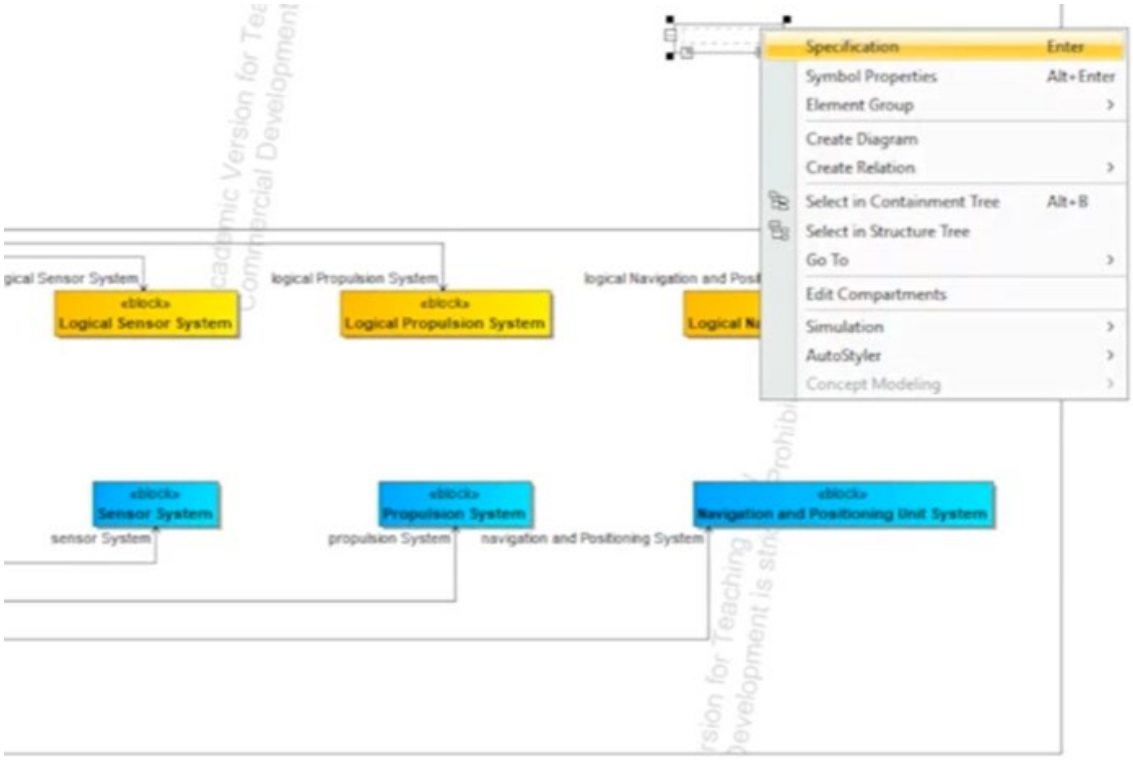

«block»
UUV
operator : Operator
RHIB : RHIB
naval Ship : Naval Ship
environment : Environment
remote Control Station : Remote Control Station
logical Communications System
logical Sensor System
logical Propulsion System
logical Navigation and Positioning Unit
«block»
Logical Power System
«block»
Logical Communications System
«block»
Logical Sensor System
«block»
Logical Propulsion System
«block»
Logical Navigation and Positioning Unit
«block»
Power System
«block»
Communication System
«block»
Sensor System
«block»
Propulsion System
«block»
Navigation and Positioning Unit System
«block»
UUV

Repeat the previous step for the 2 Solution Domain system blocks.

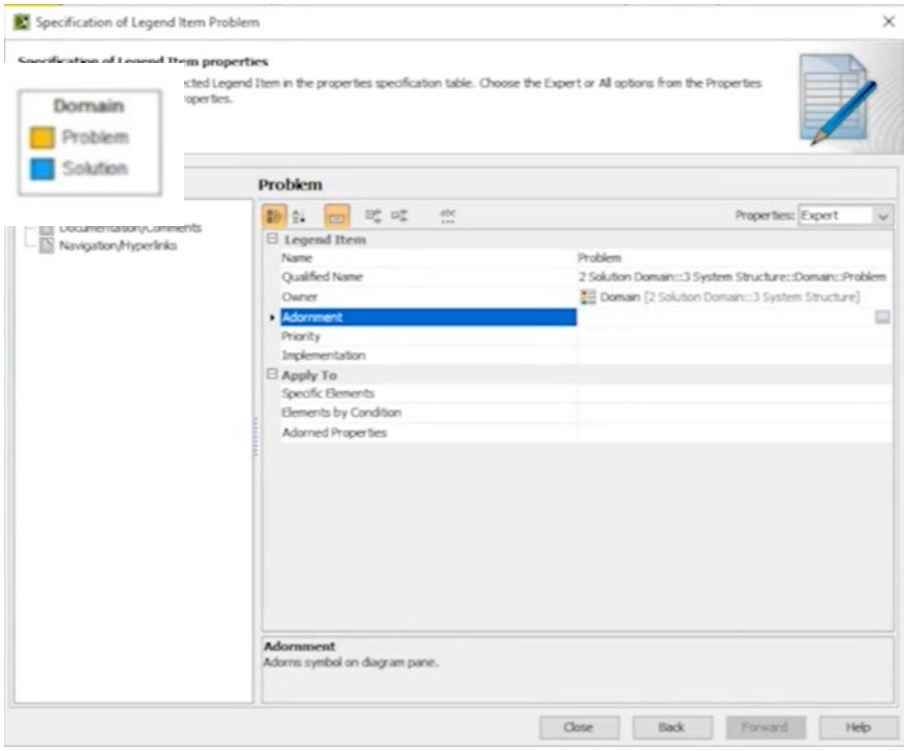


Optionally a legend can be added by selecting the Legend option on the tool bar.

Right click legend box and select Specification.

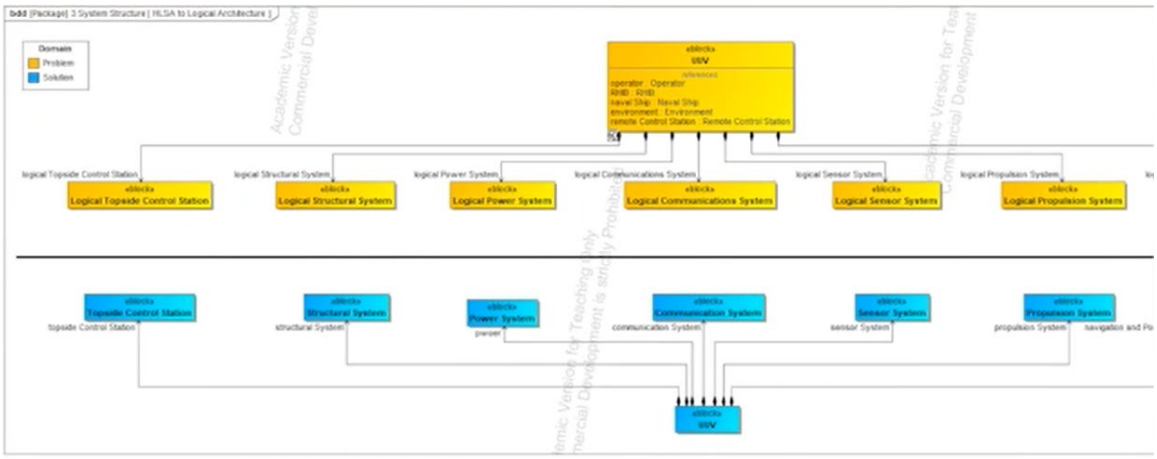


Type Domain as the Name. Click Legend Items. Type Problem as the name. And select the matching color to reflect the color chosen for the Problem Domain system blocks by selecting Adornment.

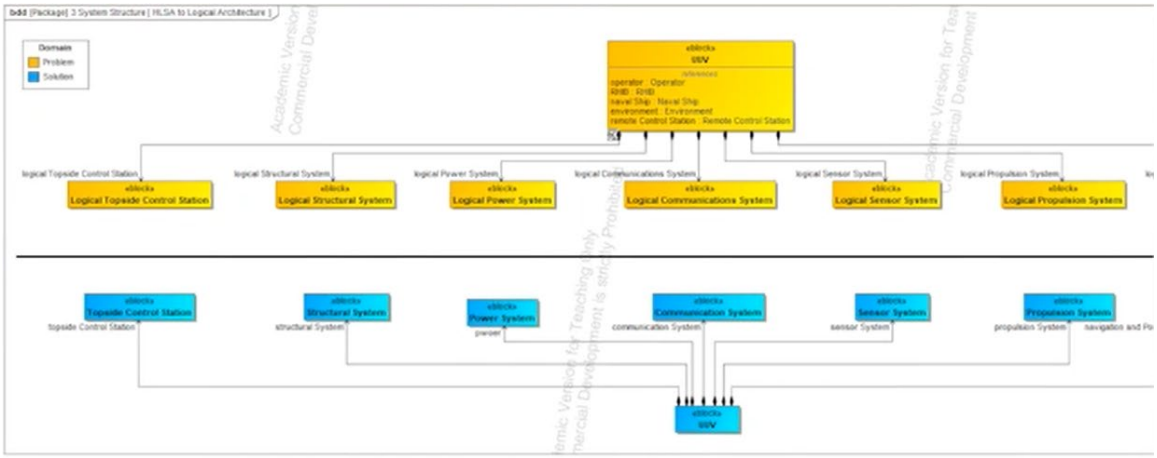


Follow the same steps to create the Solution Legend.

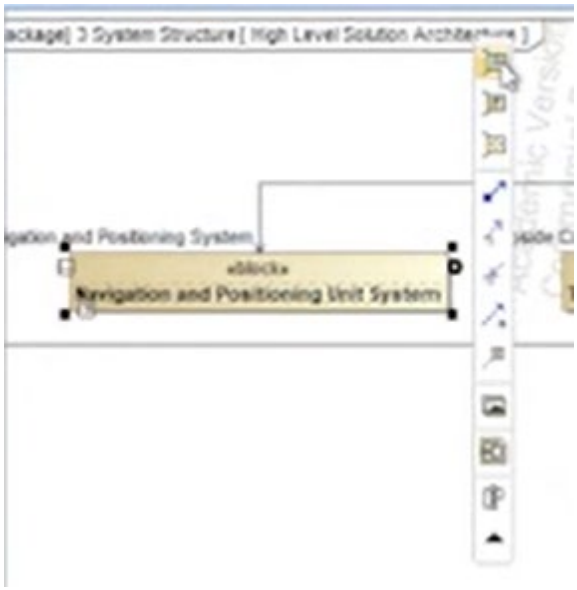
Next, we will create a separation between the two domains. Under Common in the Diagram Palette, select the Horizontal Separator and draw the separator into the diagram. Right click the separator to change to solid line and the line width to 3.



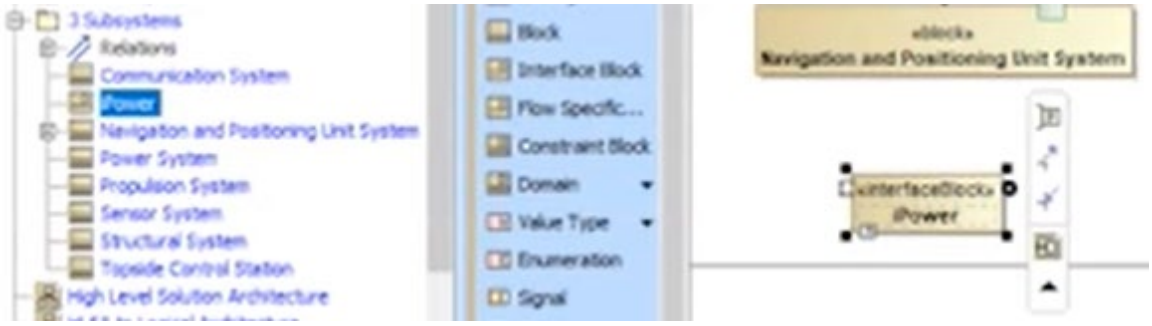
To specify Abstraction Relationship, click the Abstraction button on the diagram palette. Select the shapes of Problem Domain block and the Solution Domain block that need to be linked. Repeat for all the links.



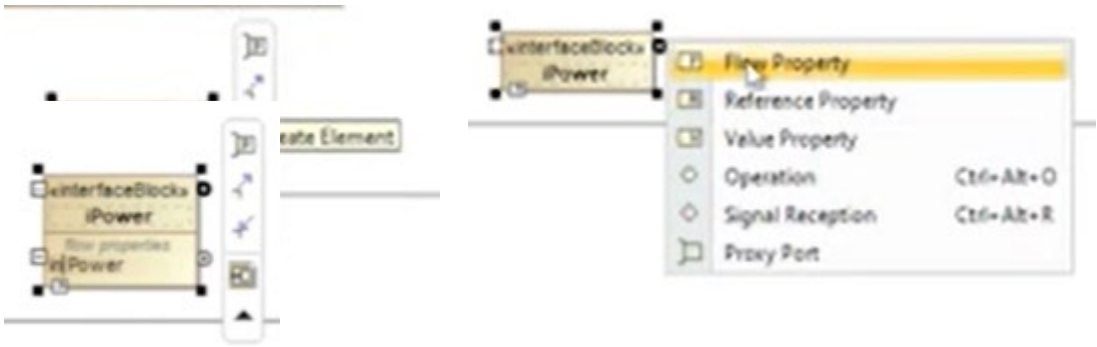
In order to capture Interfaces at the HLSA, make sure the Type Selection Mode is enabled in the High-Level Solution Architecture diagram. Select the shape of the Navigation and Positioning System and on its smart manipulator, click Proxy Port.



Type iPower next to the colon and press Enter. Select the iPower interface block in the containment tree and drag it onto the diagram Pane.



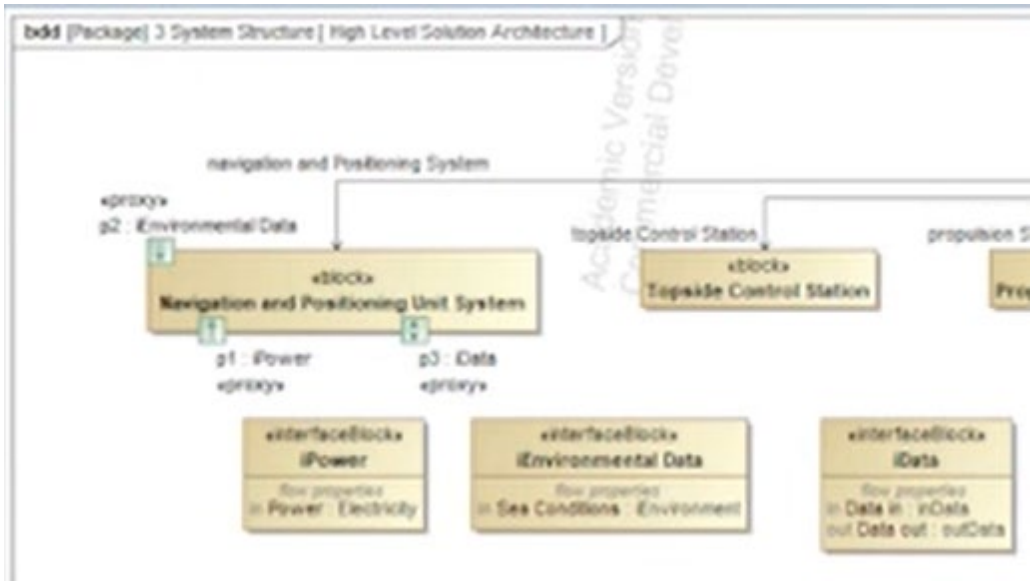
Click the create element button on the shape and select Flow Property.



Remove the out from the flow property direction and type Power to name the flow

On the smart manipulator toolbar, select the type function to type the type of the flow as Electricity.

Repeat the previous steps to create the interfaces as shown.



The last thing we have to do is to organize the Model. Expand the Data, Environmental Data, and Power interfaces. Highlight the signals created per interface. Drag the signals into the 2 Exchange Items package. Highlight the remaining interfaces and drag into 1 Interfaces package. This concludes the Initial System Structure build process.

S3: Final System Structure Tutorial

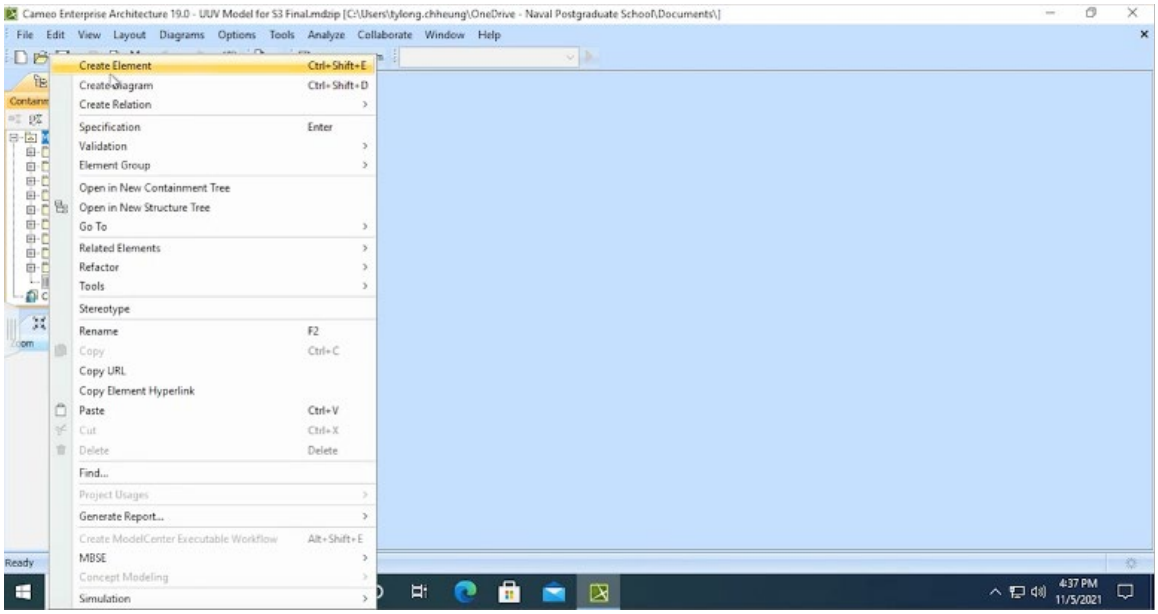
Background:

In the final phase of the system structure, the solution architectures of the subsystems are integrated into a single model. In a standard setting, there may be different solution alternatives that can be analyzed. For this tutorial, only one solution was developed. With the final system structure, it can be identified whether each subsystem can communicate successfully with each other and that interfaces between the subsystems are compatible. Using a Structure Decomposition Map and an internal block diagram, hierarchies are relationships can be illustrated and analyzed for compatibility, thus completion the final system structure.

Step-By-Step Guide:

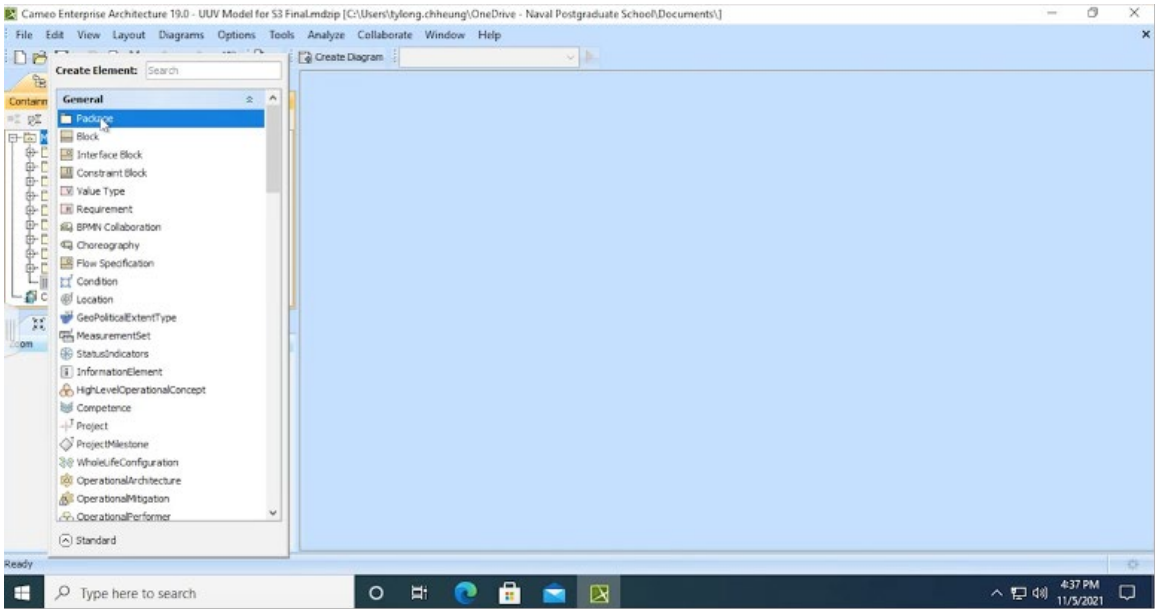
In this video we will cover how to do the UUV configuration structure in order to verify that the subsystem of the UUV system can communicate with each other and successfully integrate into one unified model. We can start off by organizing the model for the UUV configurations. Head over to the containment tree on the left-hand side and right

click on the *Model* package and from the drop-down menu select the *Create Element* option.

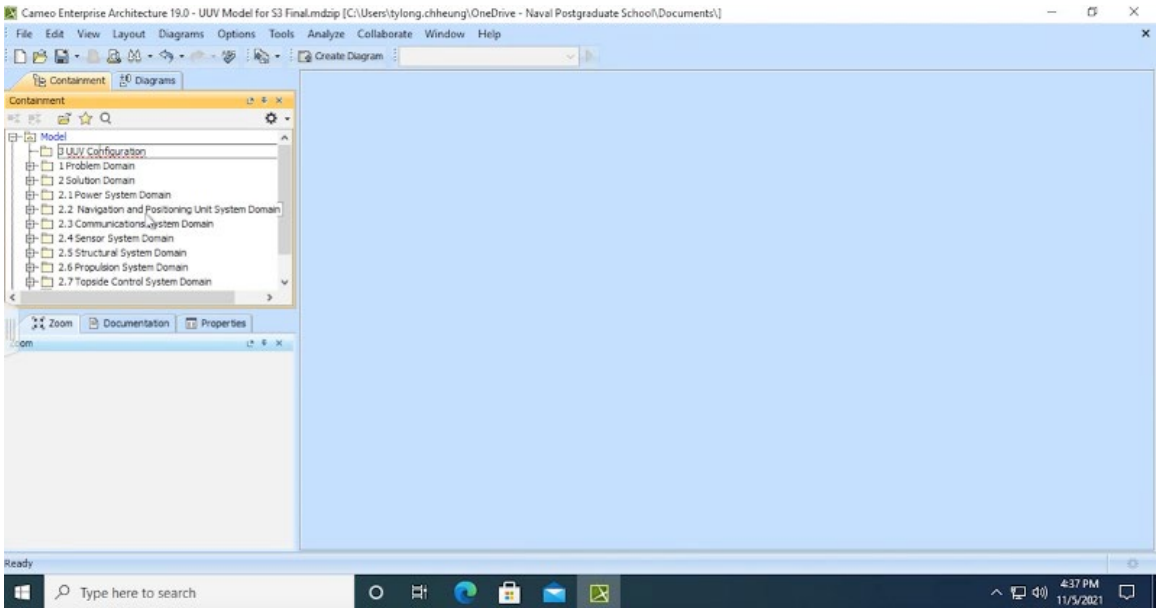


Then the *Create Element* popup menu will appear and select the *Package* option.

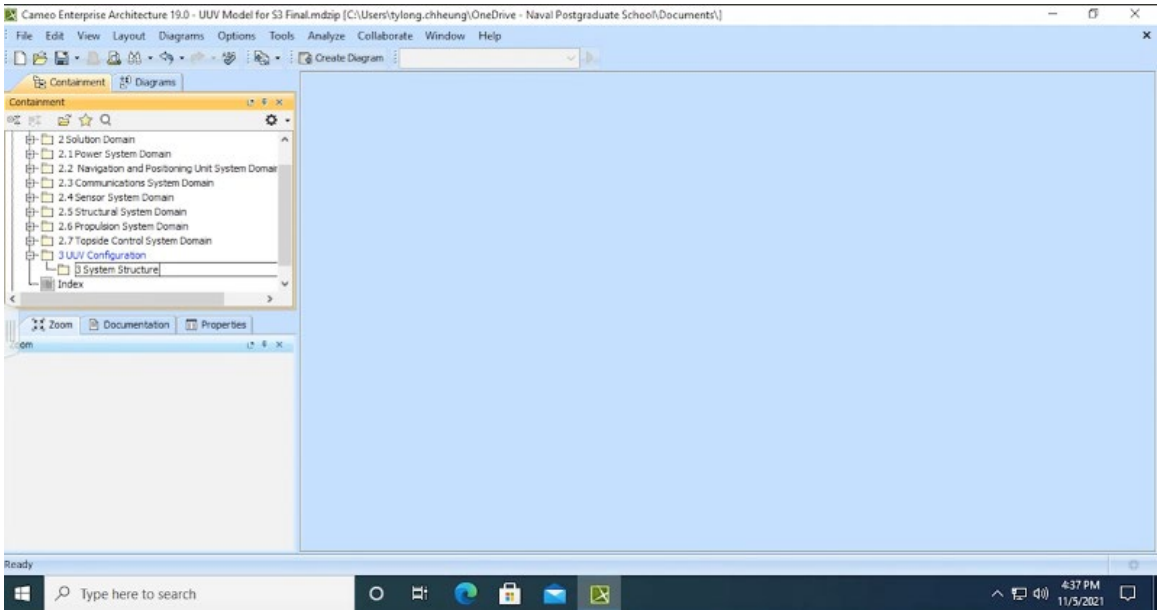
We will label the newly created package as *3 UUV Configuration*. Then we will



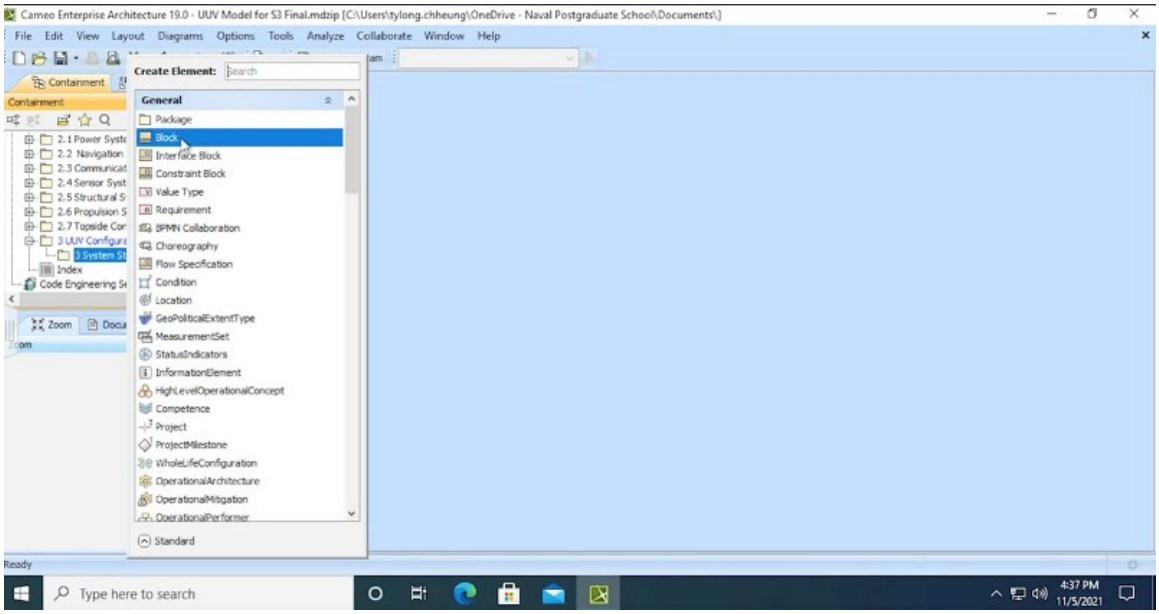
right click the *3 UUV Configuration* package and from the drop-down menu select the *Create Element* option. Then the *Create Element* popup menu will appear and select the *Package* option.



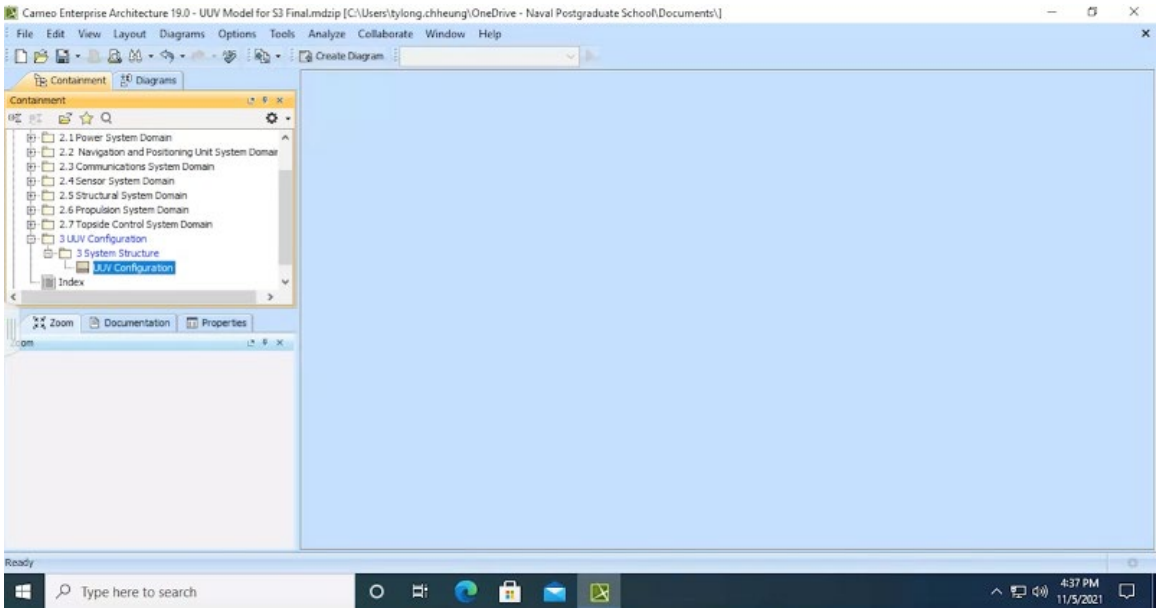
We will label the new package *3 System Structure*. Then we will right click the *3 System Structure* package and from the drop-down menu select the *Create Element* option.



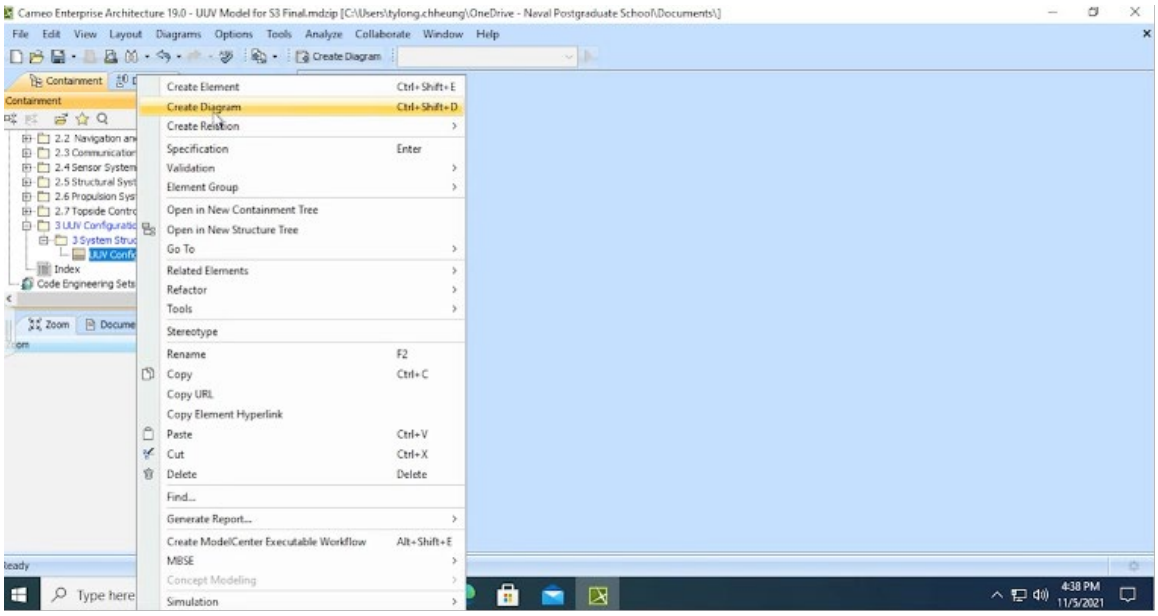
Then the *Create Element* popup menu will appear and select the *Block* option.



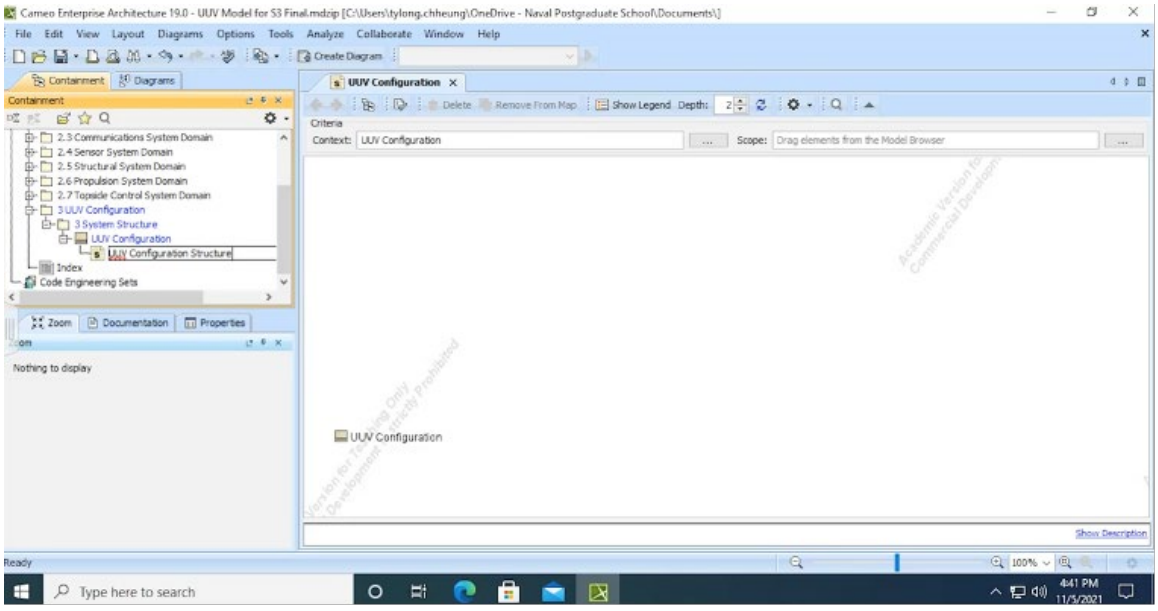
We will label the block as *UUV Configuration*.



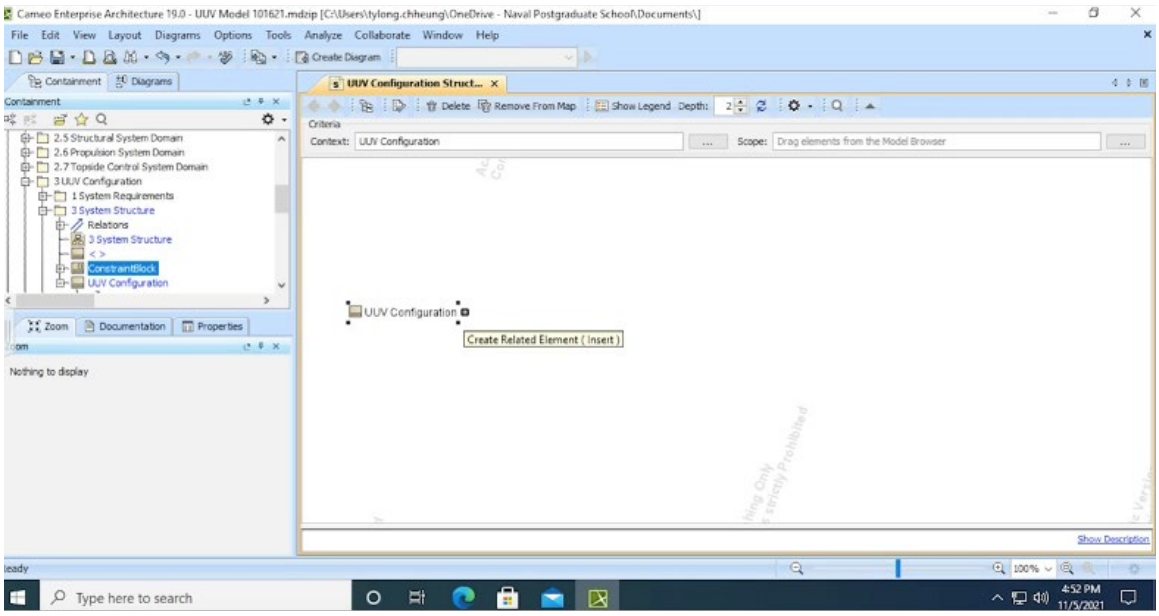
Then we will right click on the *UUV Configuration* block and from the drop-down menu select the *Create Diagram* option.



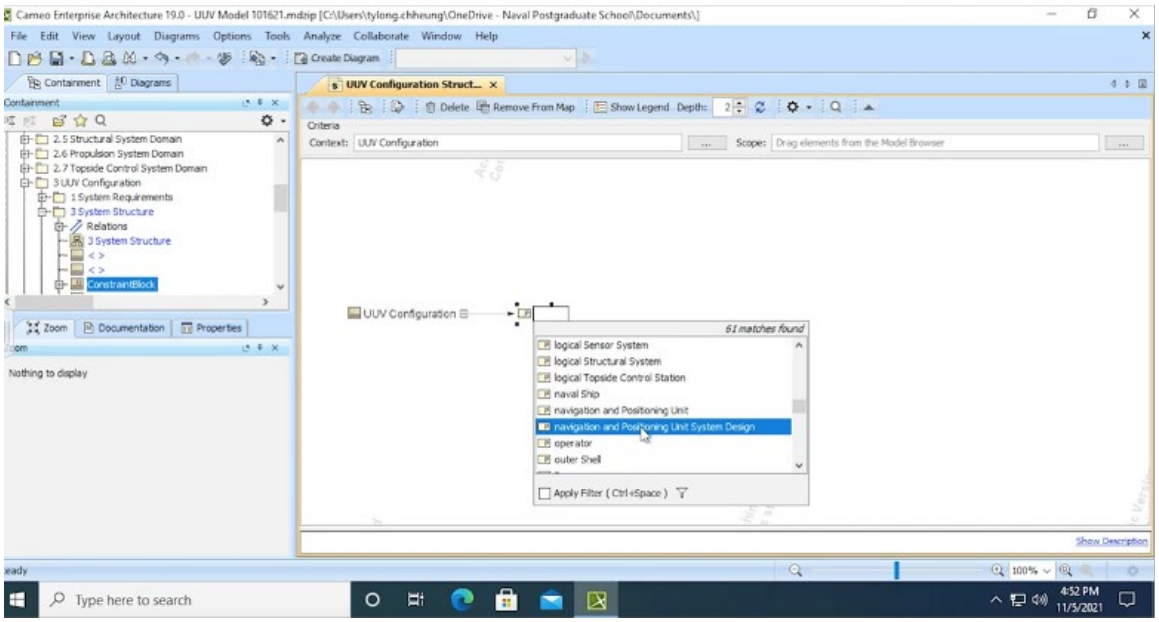
Then the *Create Diagram* popup menu will appear and on the search text box we will type *sdm* and select the *Structure Decomposition Map* option. And then label it as *UUV Configuration Structure*.



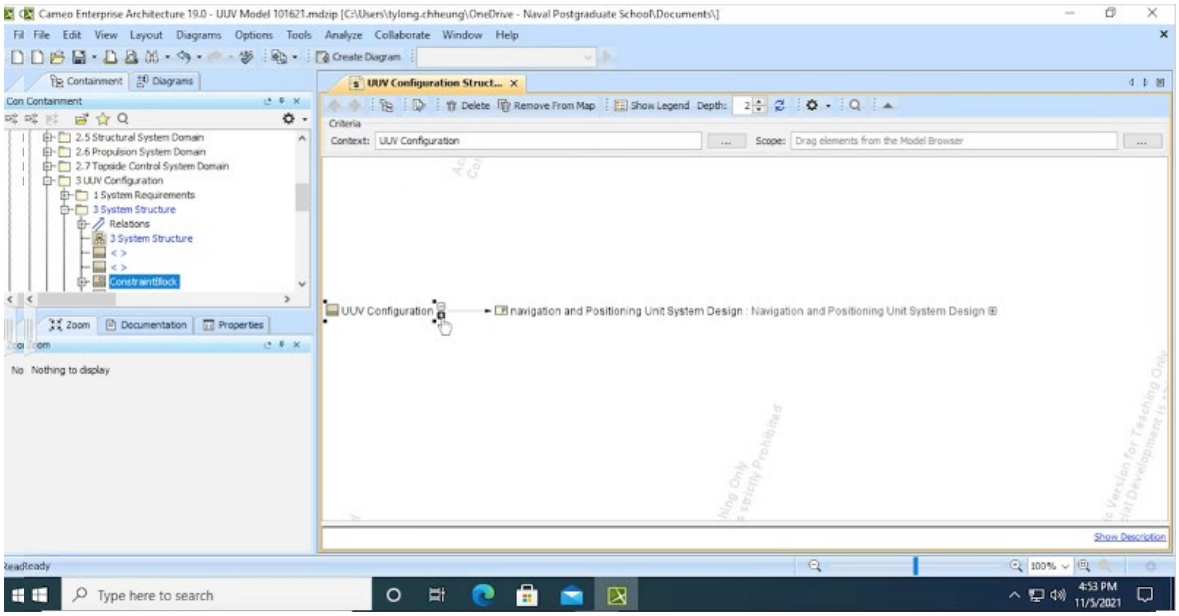
Inside the diagram plane the *UUV Configuration* block appears and then select the *Create Related Element (Insert)* option.



Then from the scroll down menu we are going to be selecting our first subsystem we are going to include which is the *Navigation and Positioning Unit System Design*.

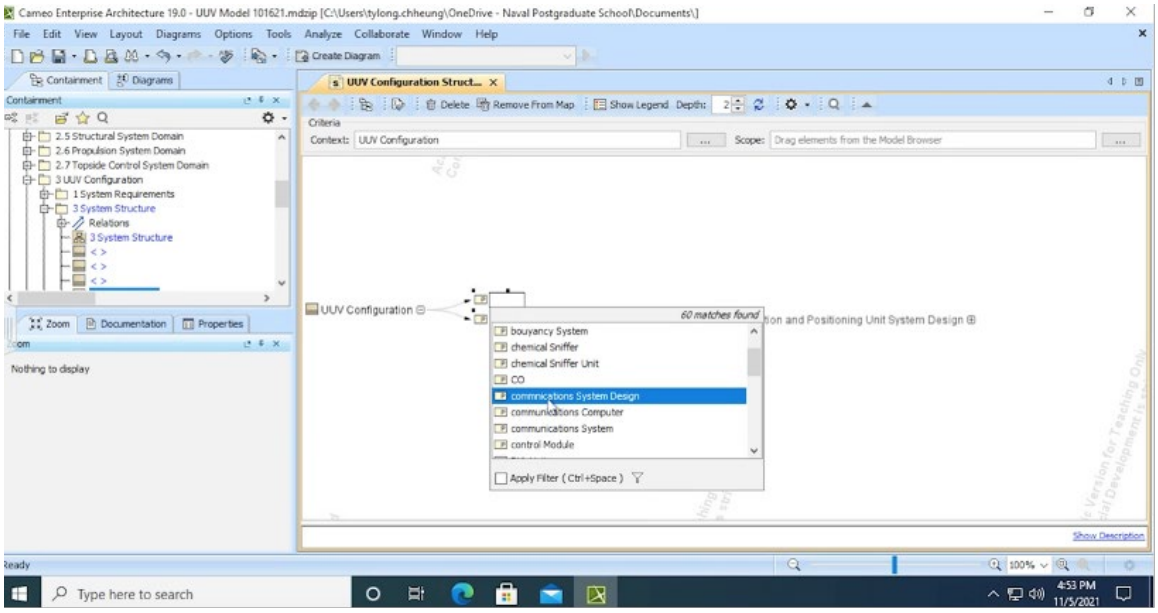


Once you have selected the *Navigation and Positioning Unit System Design* subsystem, then you can select the box with the plus sign inside to expand the subsystem and show all of its internal components.

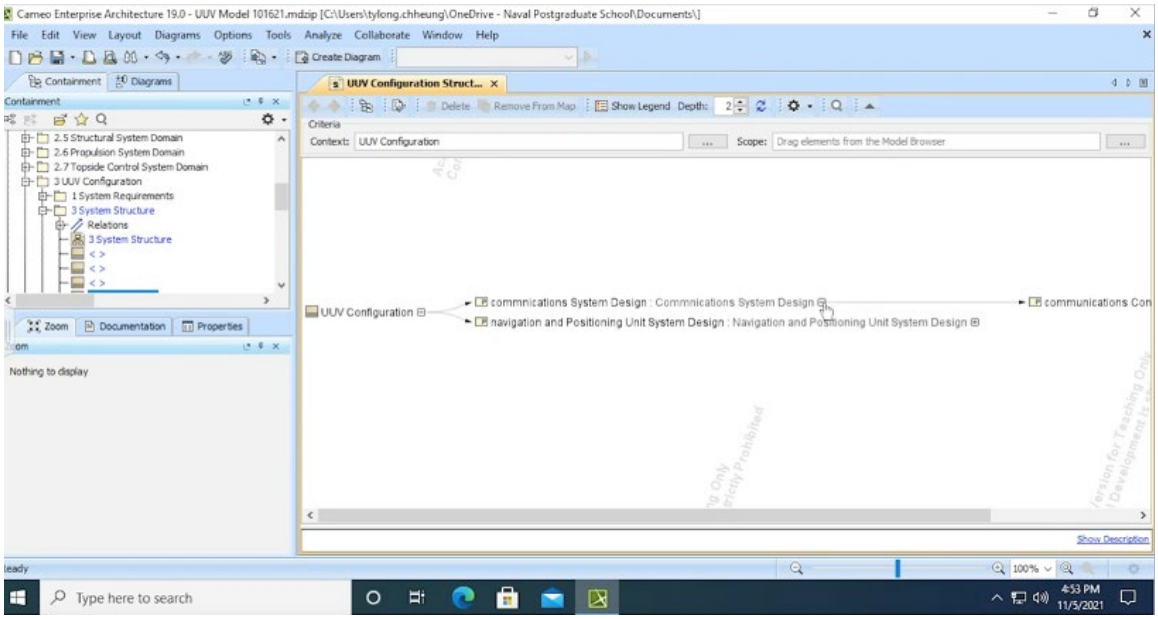


You are able to minimize the subsystem by pressing on the box with the minus sign inside. In order to add another subsystem, you can right click on the black box with the plus sign inside of it.

You can also right click on *UUV Configuration* and select *Create Related Element* option from the popup menu. For this related element we will be adding the *Communications System Design* subsystem.

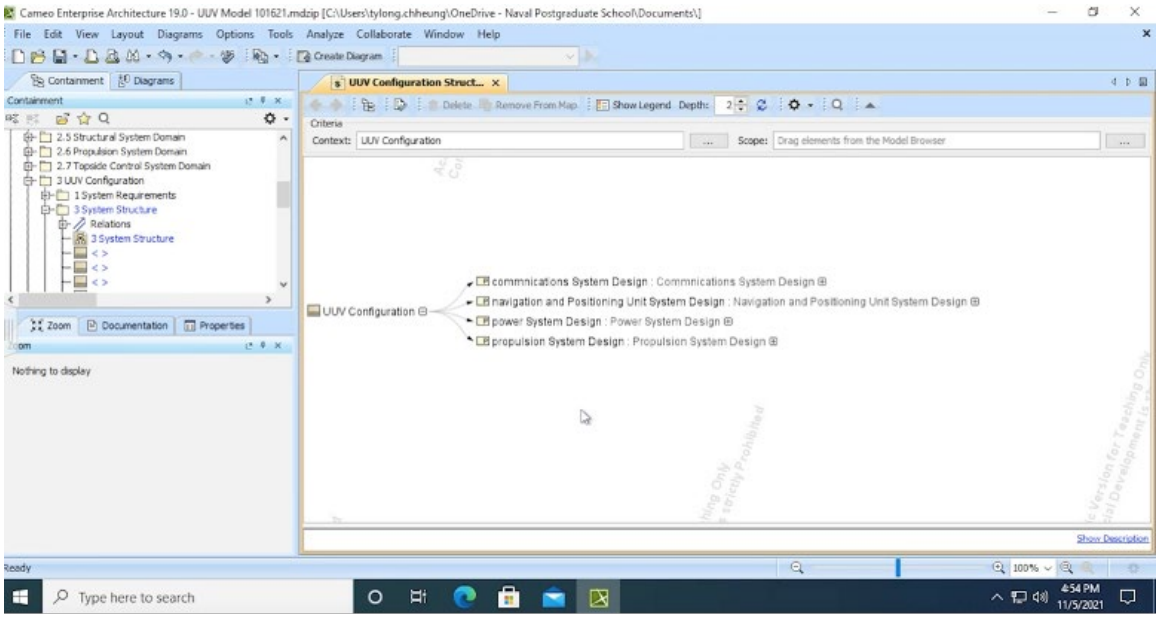


We are also able to expand the *Communications System Design* subsystem to show all of its components.

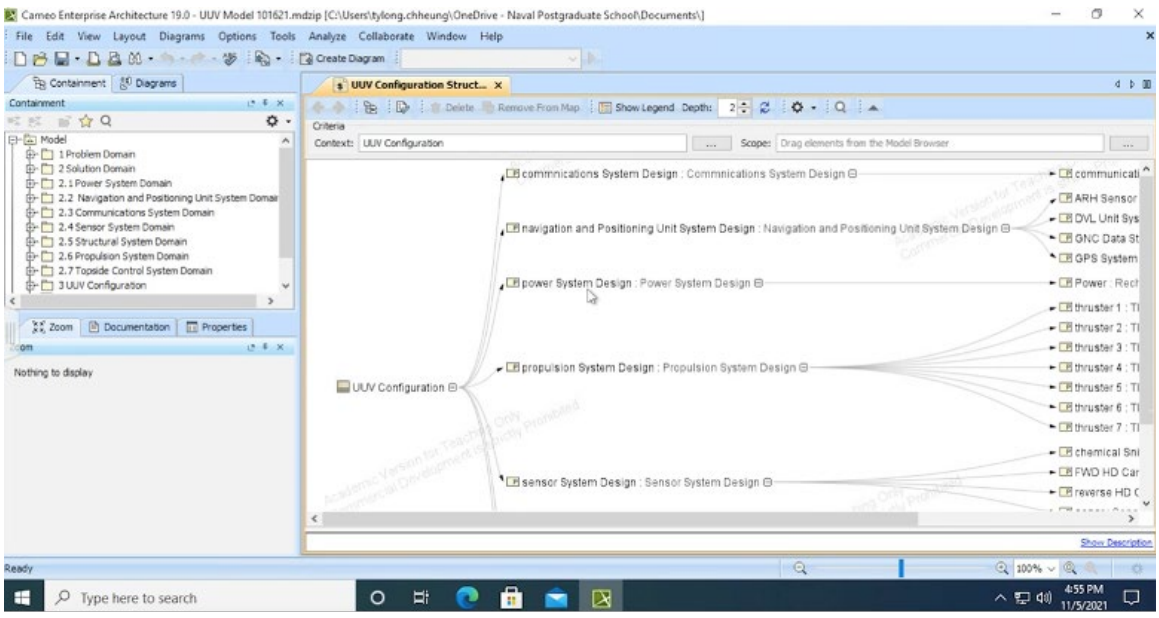


We will then go back and repeat the process to add the *Power System Design* subsystem and the *Propulsion System Design* subsystem.

If you just go in and put the rest of the subsystems in the diagram plane, the final diagram will look similar to the screenshot below.



Here is the *UUV Configuration Structure Decomposition Map* it includes all the subsystems of the UUV and along with their expanded subsystems. That concludes the tutorial on the final system structure. Thank you.



S4: System Parameters Tutorial

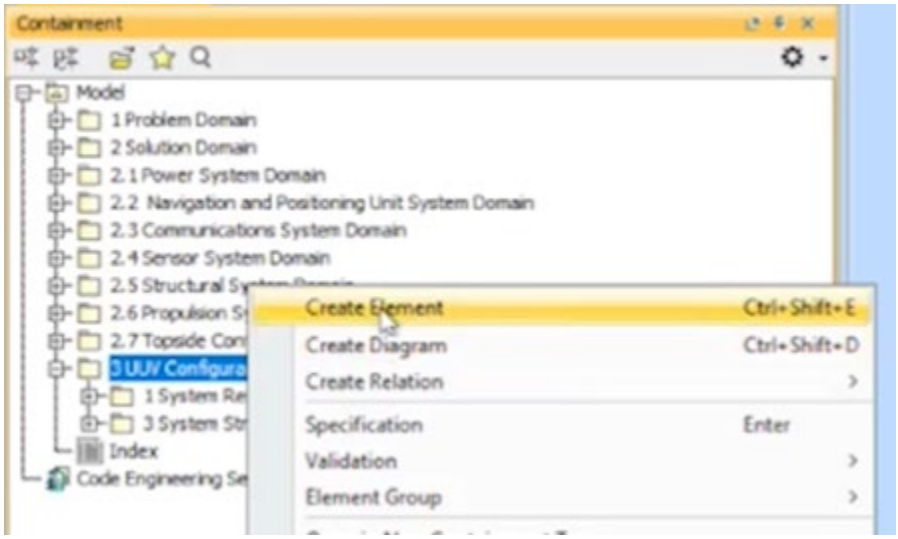
Background:

With the completion of the system structure, the system parameters can be captured to illustrate the quantitative characteristics of the system. The parameters satisfy the non-functional system requirements and can be derived from the measurements of effectiveness that were previously established. A SysML BDD is used to capture the system parameters and can display values and constraints that apply to the system.

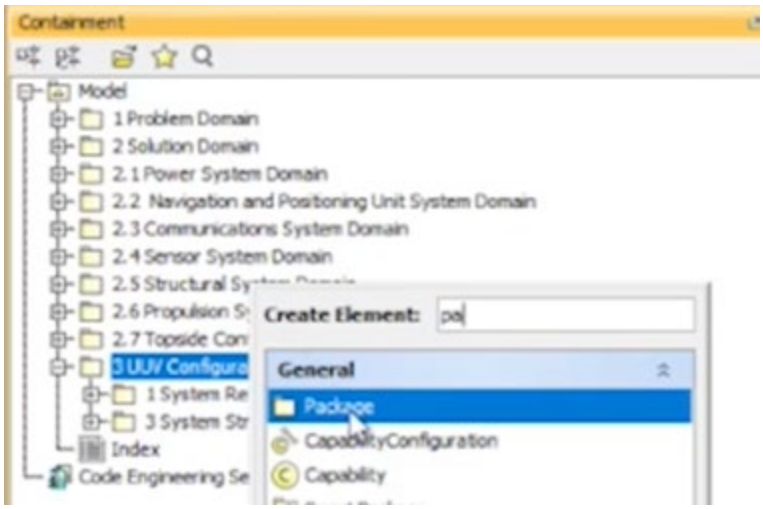
Step-By-Step Guide:

How to calculate the mass of a total system

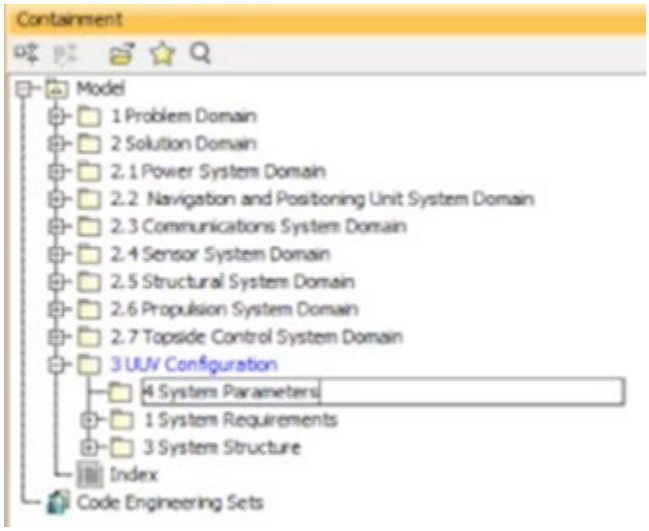
Create a new package by right clicking 3 UUV Configuration from the Containment tree, select Create Element.



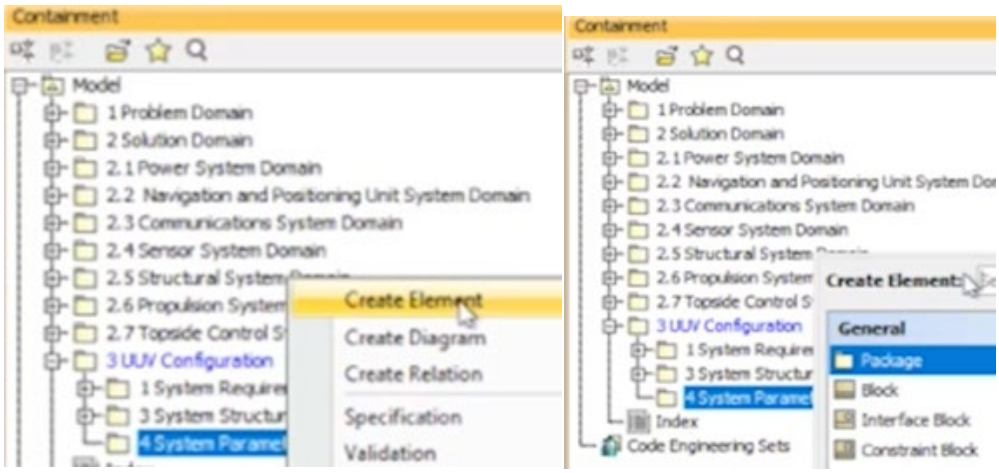
Type pa in the search window to quickly locate the Package element.



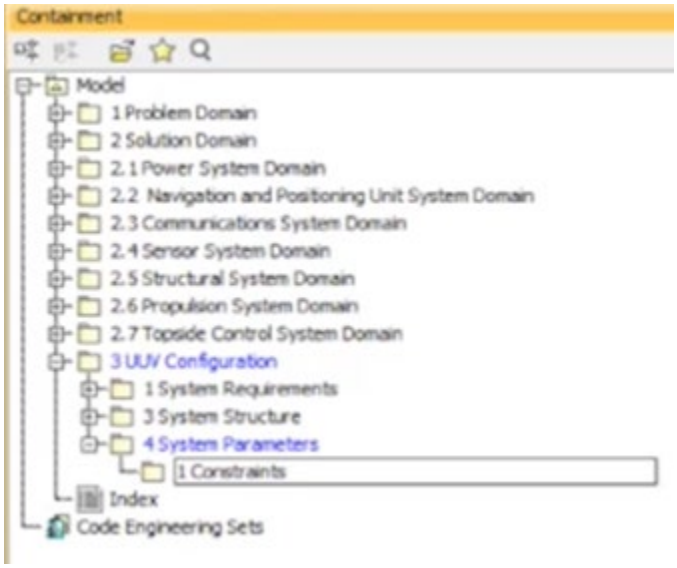
Type 4 System Parameters as the label for the package, then enter.



Create an additional package within the 4 Systems Parameters package by right clicking the 4 Systems Parameters, and selecting Create Element then selecting Package.

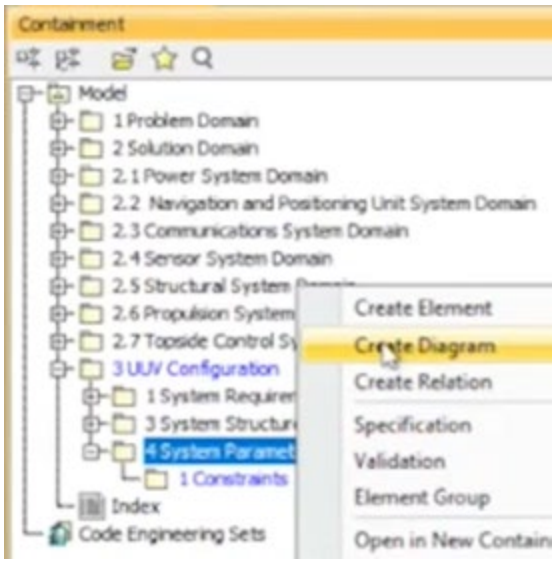


Type 1 Constraints as the label for the package, then enter.

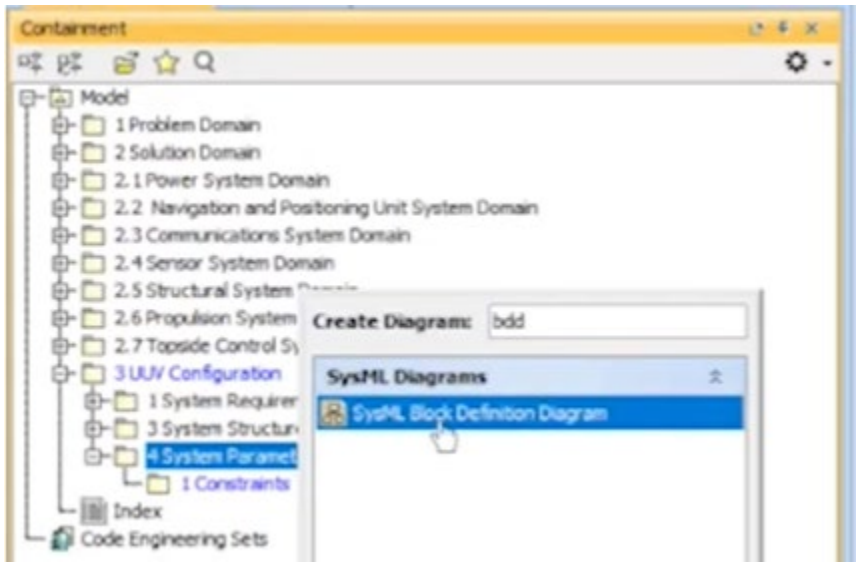


Create a Block Definition Diagram to capture system parameters.

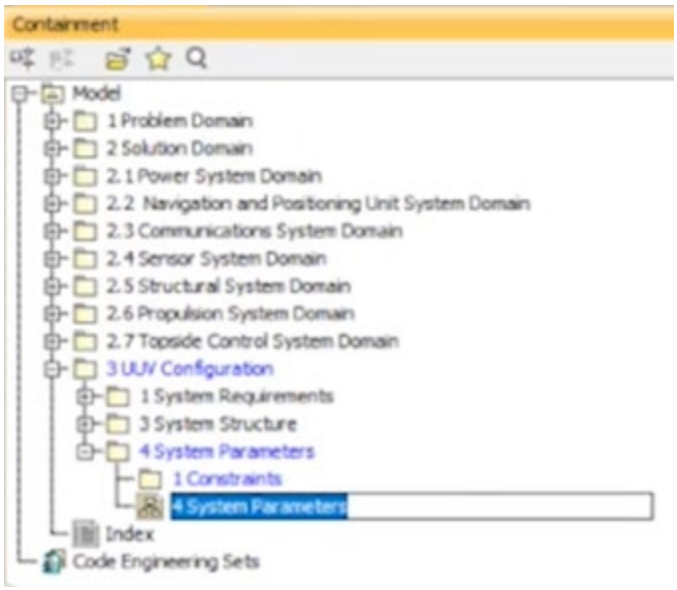
Right click 4 System Parameters, Select Create Diagram.



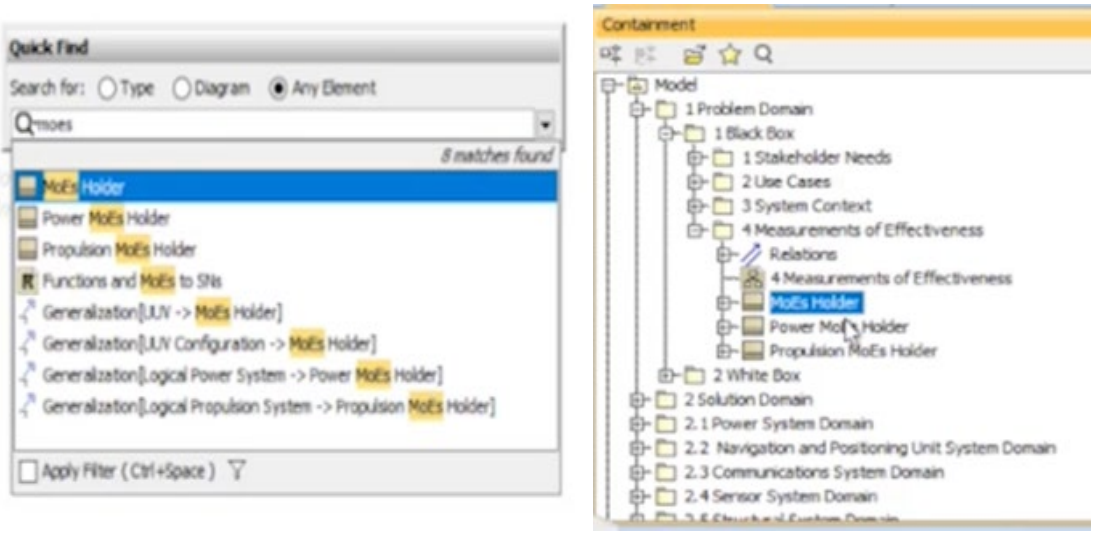
Type BDD in the search window, locate and select SysML Block Definition Diagram.



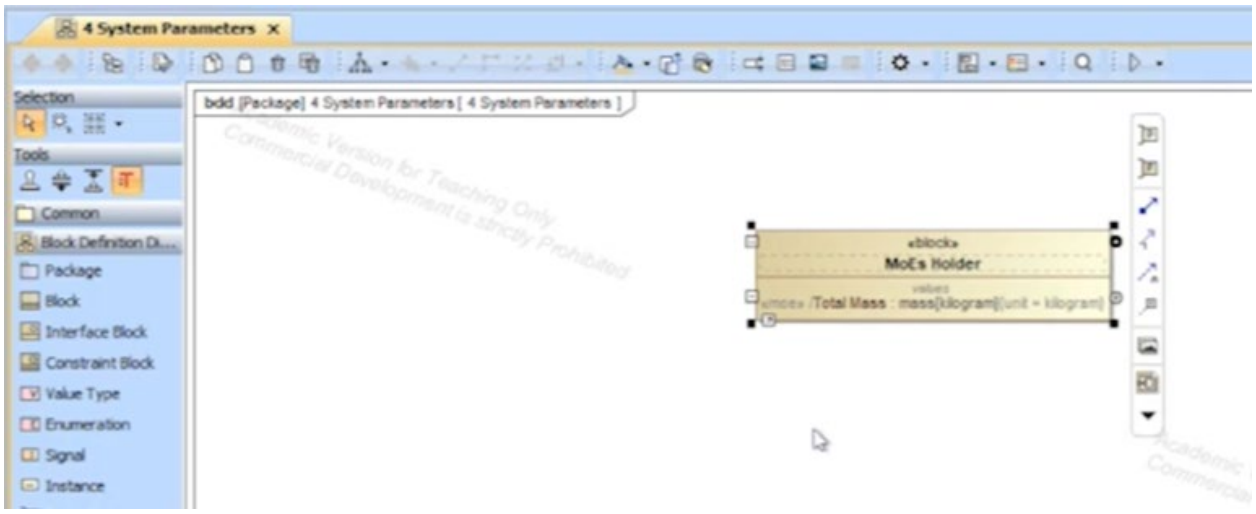
Notice the 4 Systems Parameters has appeared within the containment tree under the 4 System Parameters package. No need to change the name.



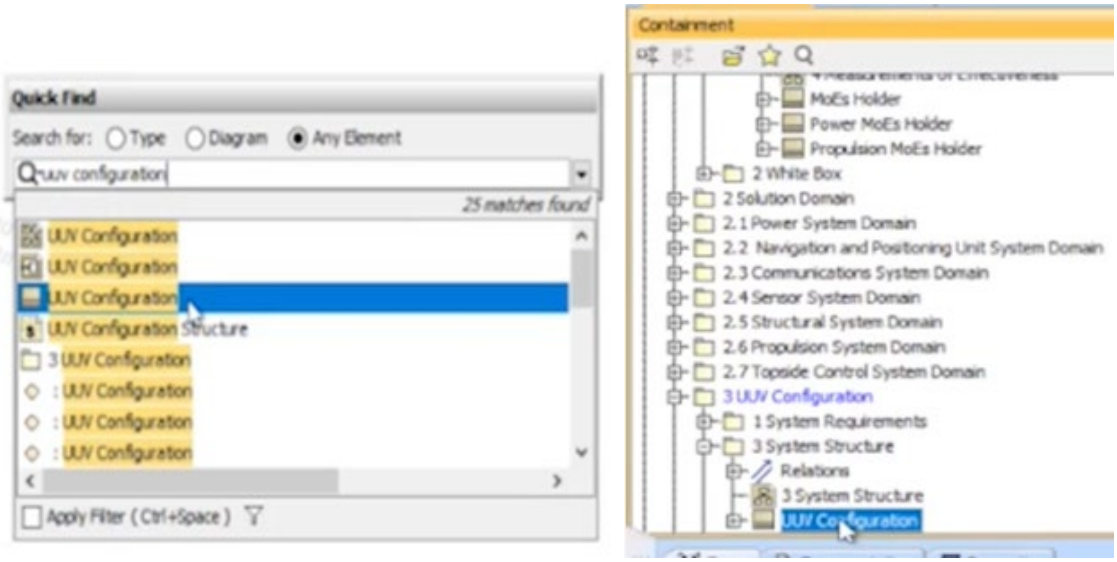
Locate the MOEs Holder block by using the keypad press and hold ctrl, alt, F. This will automatically display a search window called Quick Find. Verify the Any Element is selected then enter MOE in the search window. Select MOEs Holder, this will automatically highlight the MOEs Holder in the Containment tree.



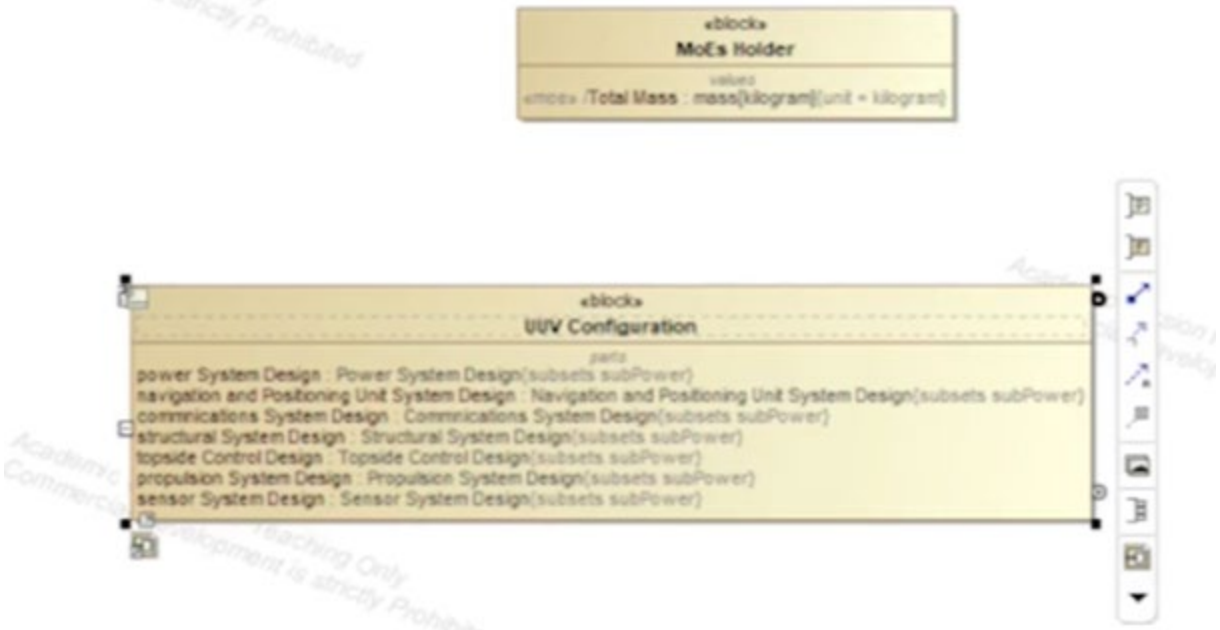
Select and drag this block into the diagram pane.



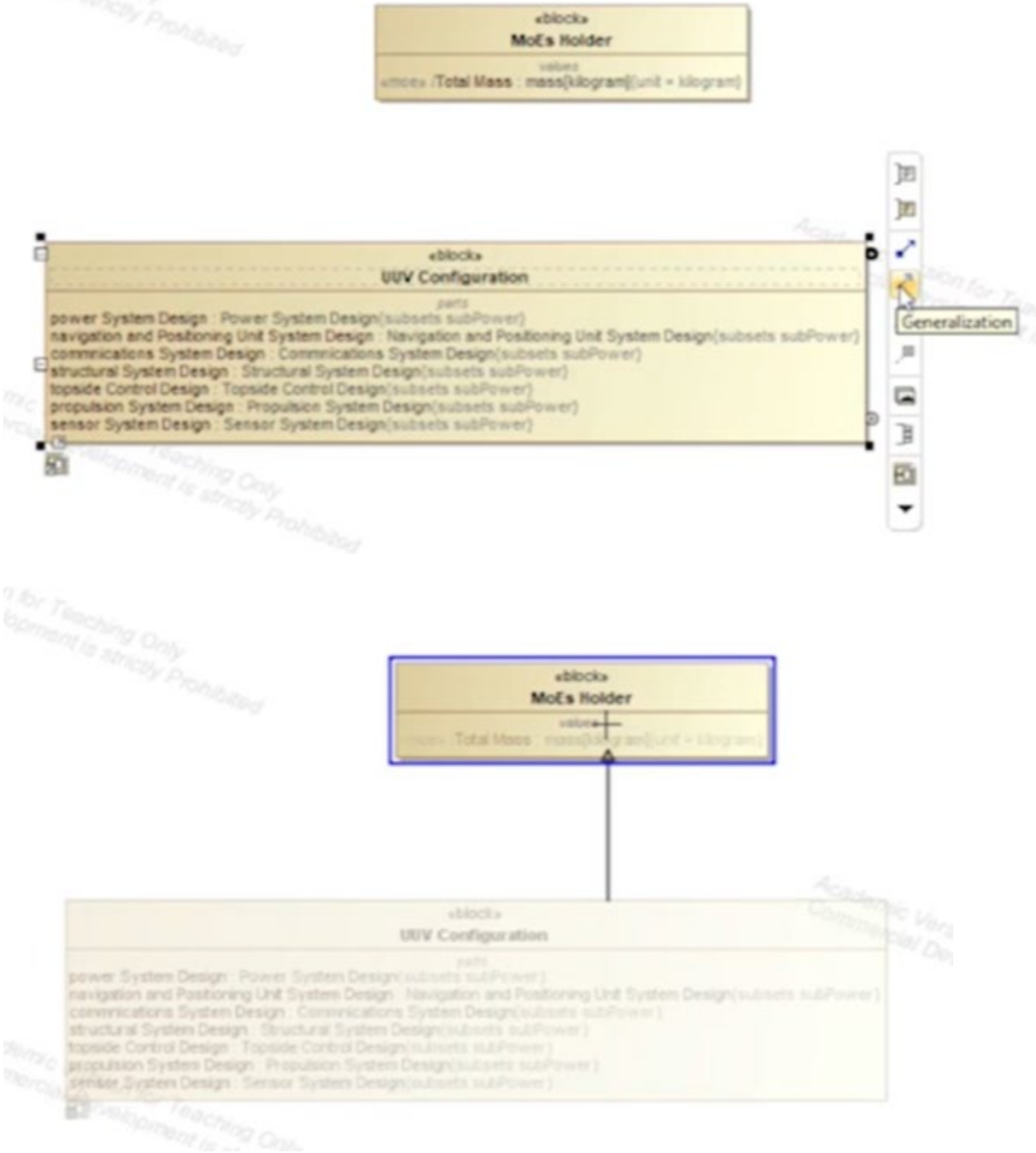
Locate the UUV Configuration block by using the keypad press and hold ctrl, alt, F. This will automatically display a search window called Quick Find. Verify the Any Element is selected then enter UUV Configuration in the search window. Select UUV Configuration block, this will automatically highlight the desired package in the Containment tree.



With UUV Configuration highlighted, drag to main window.

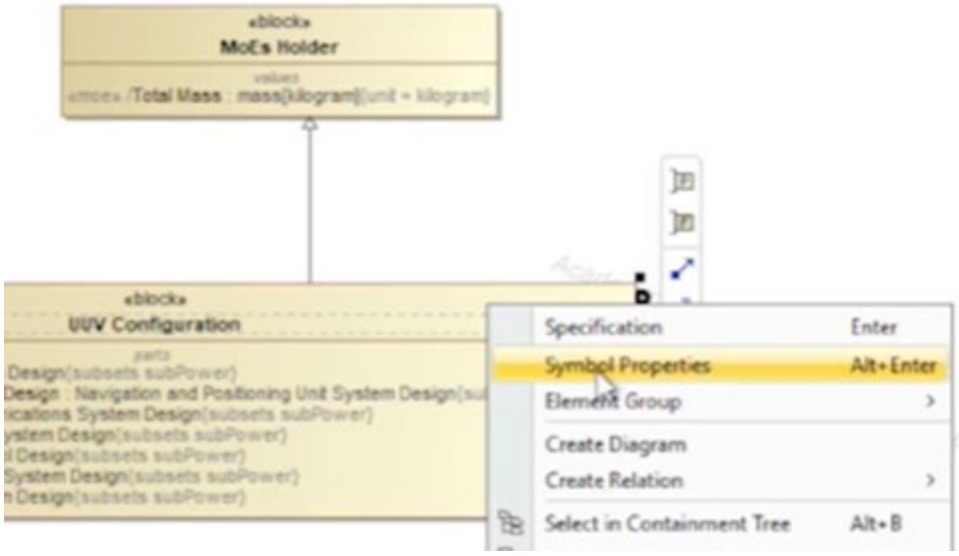


Utilizing the smart manipulator tool bar for the UUV Configuration select Generalization. A draggable arrow will appear, drag the arrow to the MOEs Holder block and select.

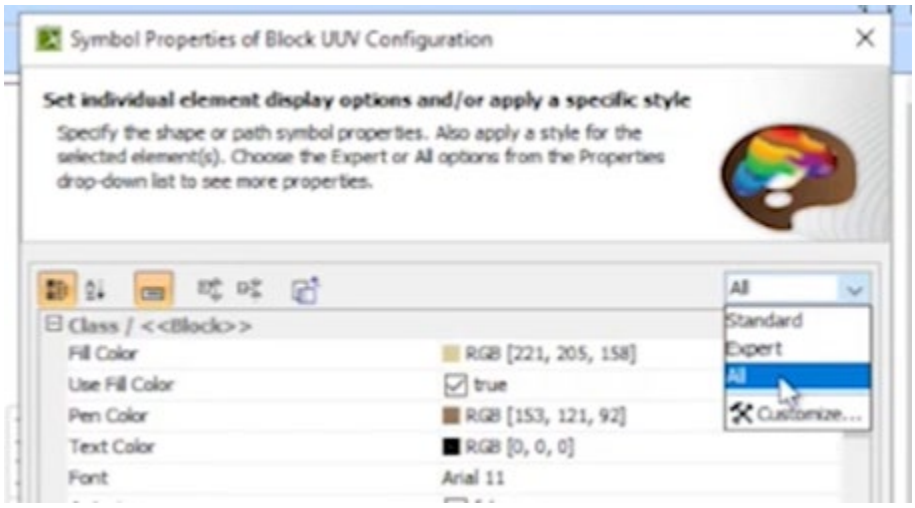


Show inherited properties

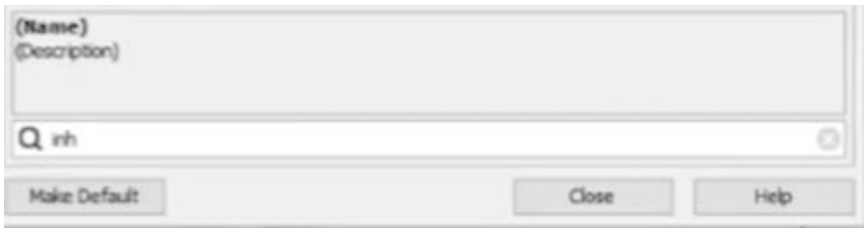
Right click UUV Configuration. Select Symbol Properties.



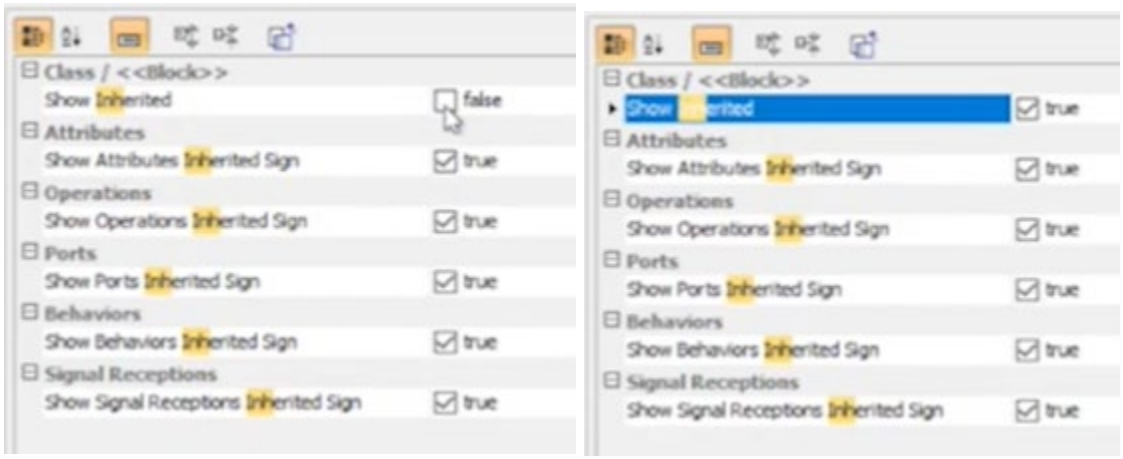
Verify the drop-down menu is set to All.



Locate search bar at the bottom of the Symbol Properties menu and type inh to find all items related to Inherited.

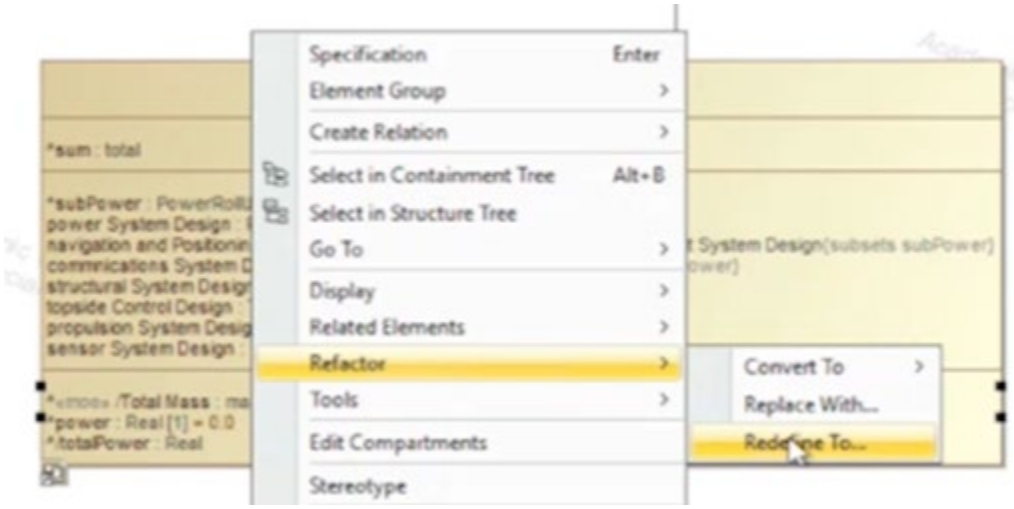


Locate the Class/<<Blocks>> field and change from false to true. Select Close when complete.

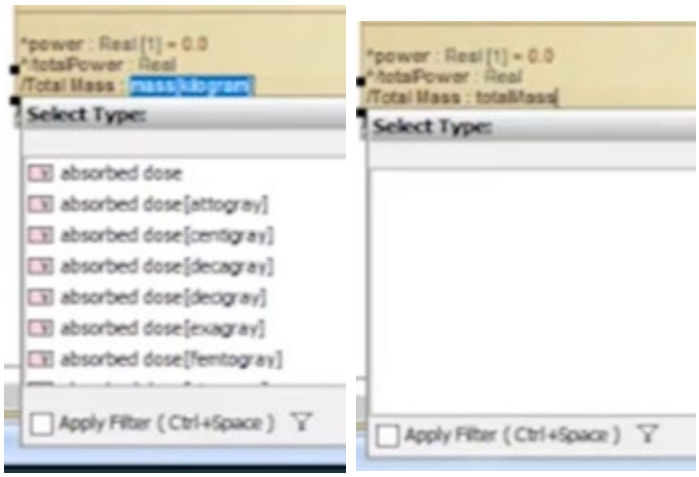


Redefine Total Mass property.

Right click TotalMass Property in the UUV Configuration block. Select Refactor, then select Redefine To…



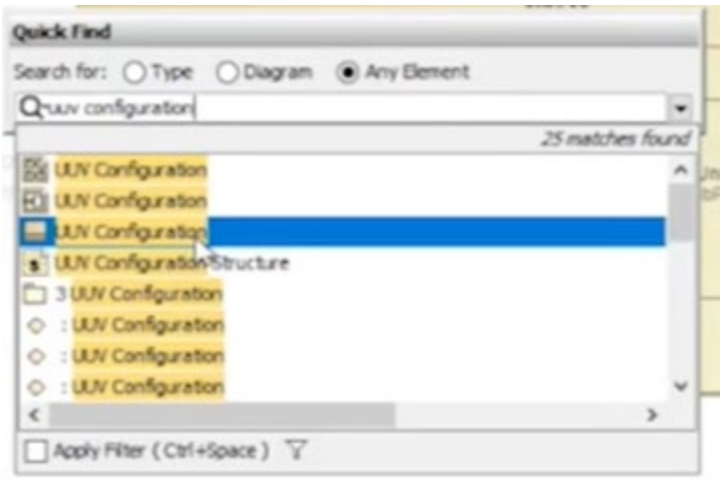
Change the name to totalMass, hit enter when complete.

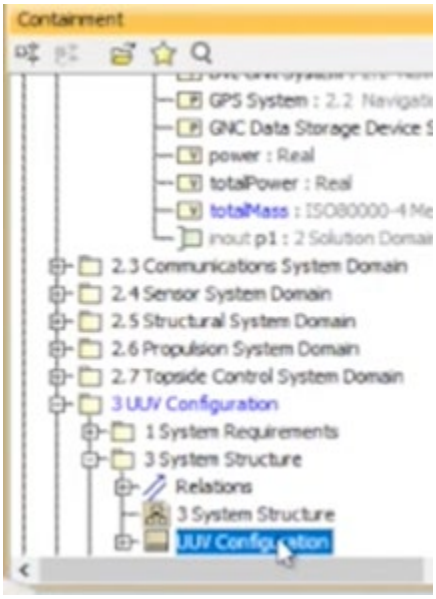


This should have changed to reflect totalMass defined by mass(kilogram).

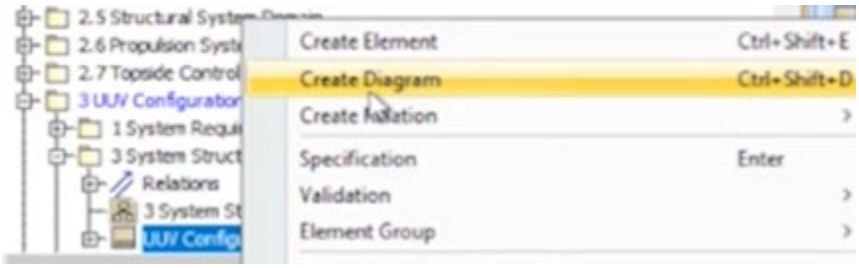
Create bind constraints parameters to the subsystem parameters

Locate the UUV Configuration block by using the keypad press and hold ctrl, alt, F. This will automatically display a search window called Quick Find. Verify the Any Element is selected then enter UUV Configuration in the search window. Select UUV Configuration block, this will automatically highlight the desired package in the Containment tree.

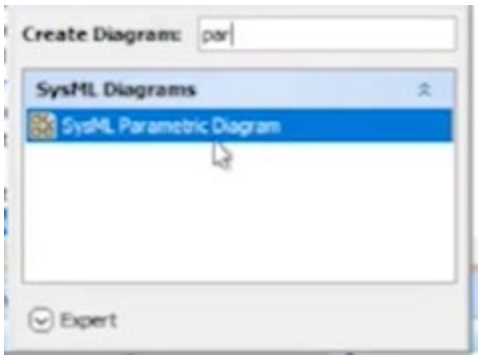




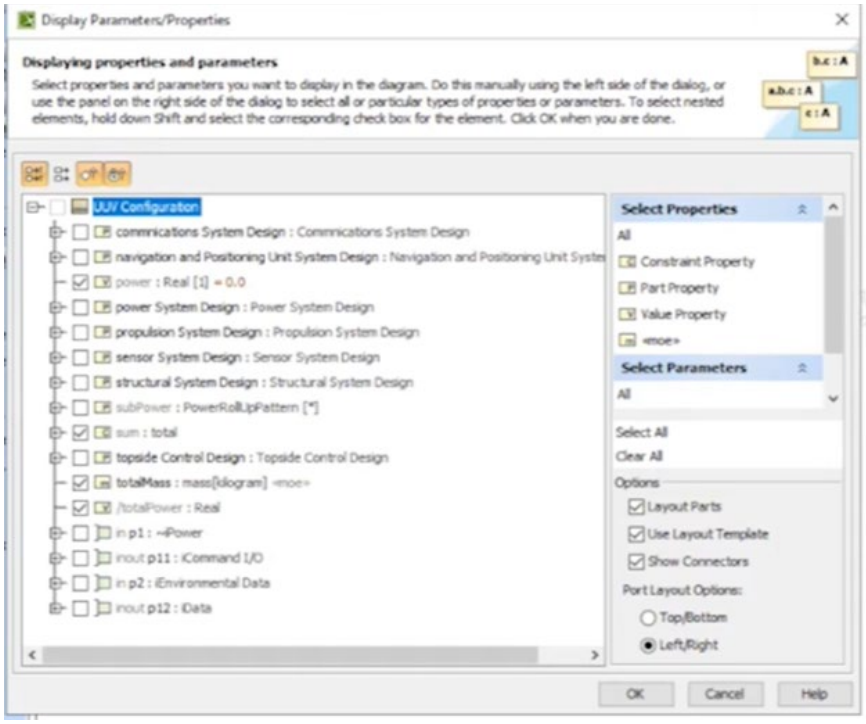
Right click UUV Configuration block select Create Diagram.



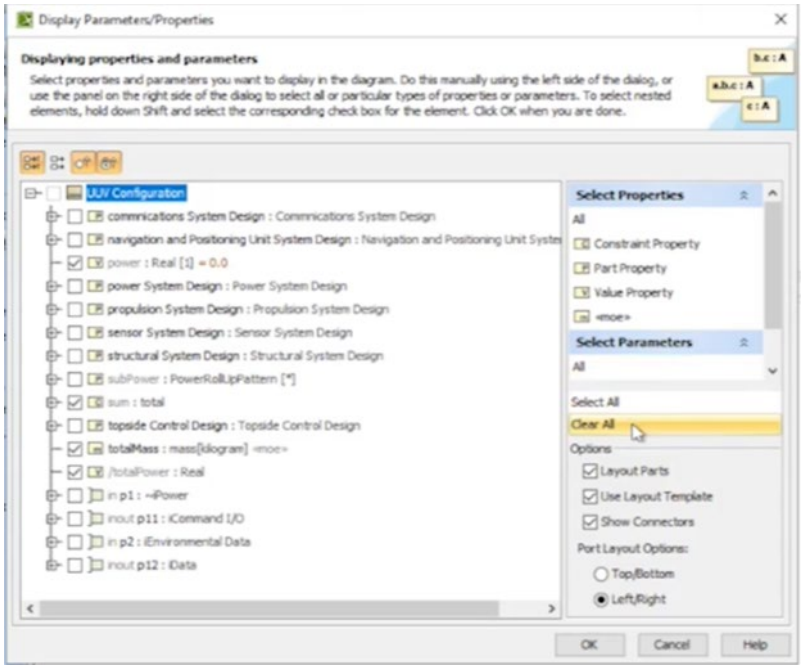
Use the search window in Create Diagram and type par to locate and select SysML Parametric Diagram.



The Display Parameters Properties window will automatically open.

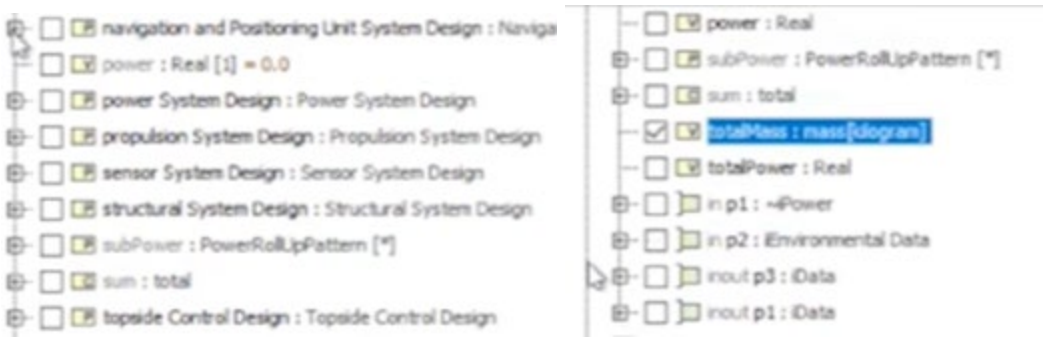


Select Clear All.

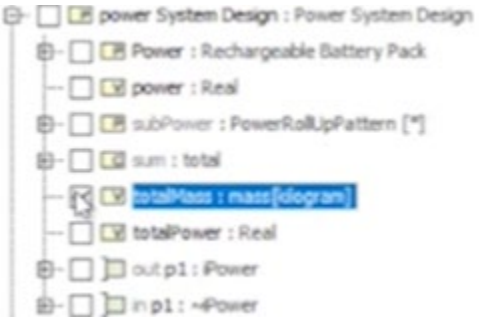


Expand Communications System Design. Locate and select totalMass

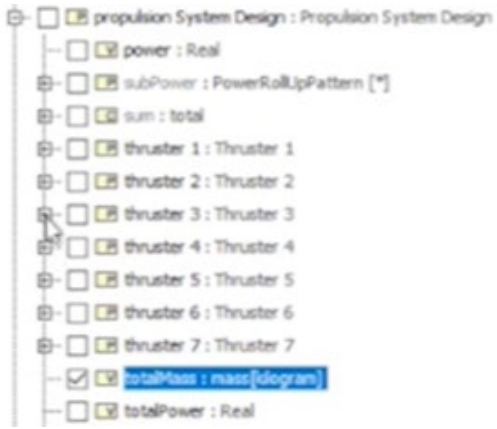
Expand navigation and Positionings Unit System Design. Locate and select totalMass.



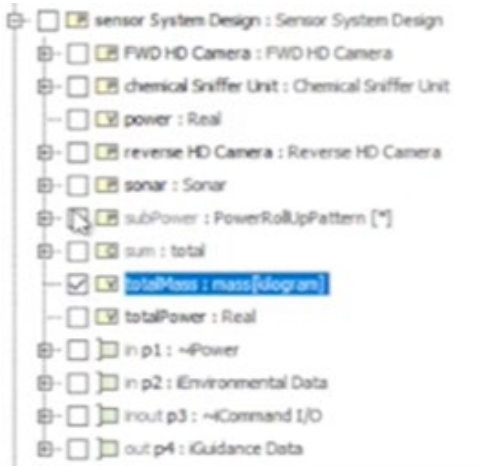
Expand power System Design. Locate and select totalMass.



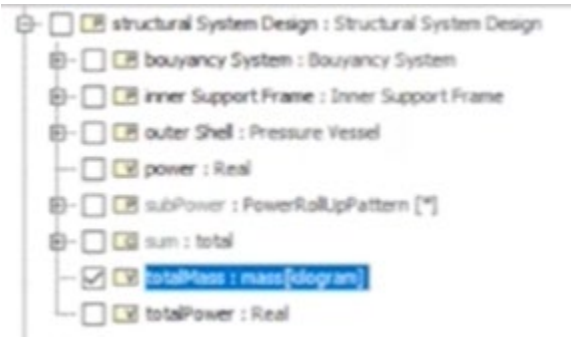
Expand propulsion System Design. Locate and select totalMass.



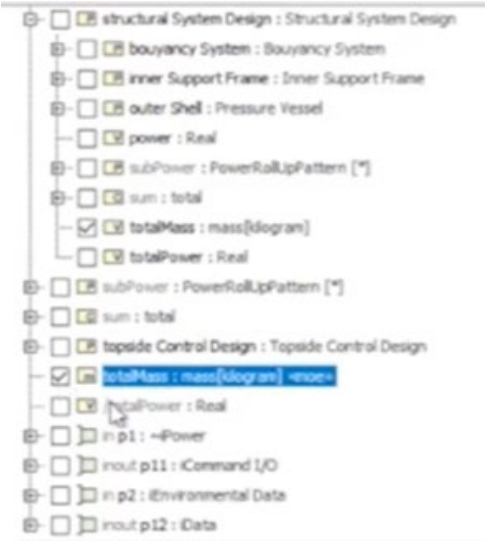
Expand sensor System Design. Locate and select totalMass.



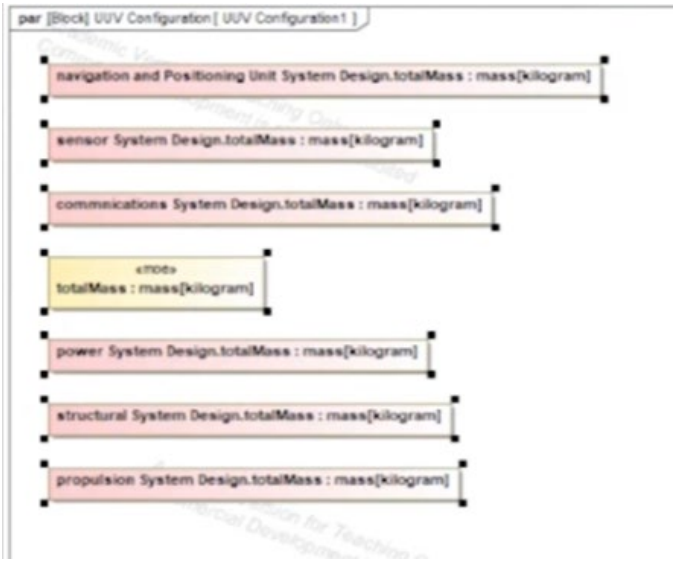
Expand structure System Design. Locate and select totalMass.



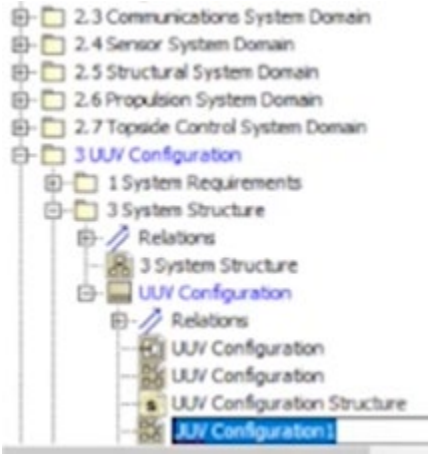
Locate and select additional totalMass property at the bottom of the MOE list, then select OK to close the window.

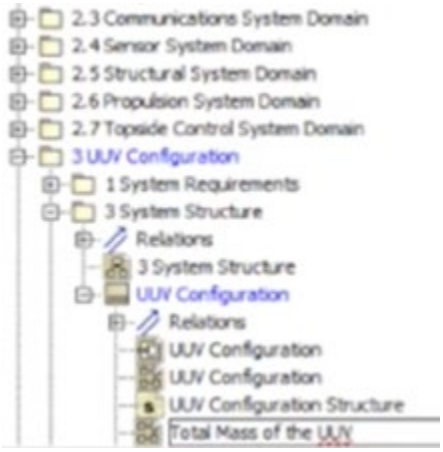


The following value property blocks should have appeared on the diagram pane for UUV Configuration1.

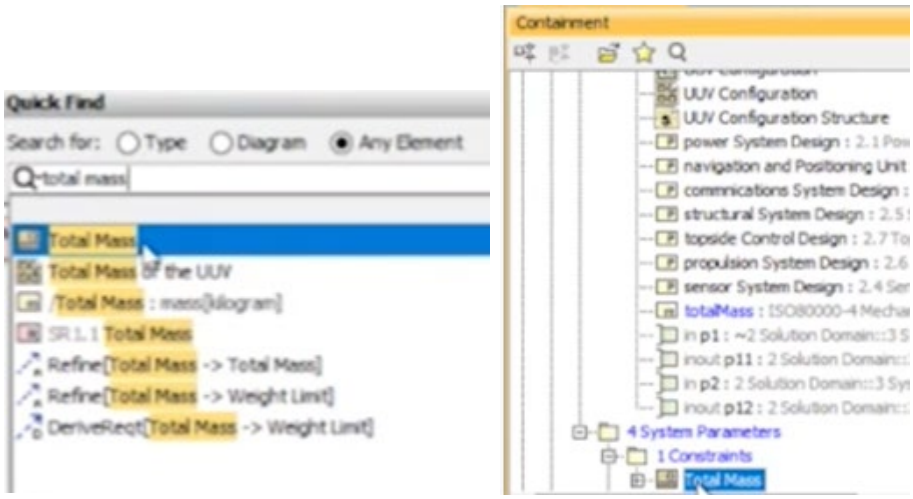


Rename the Diagram from UUV Configuration1 to Total Mass of the UUV

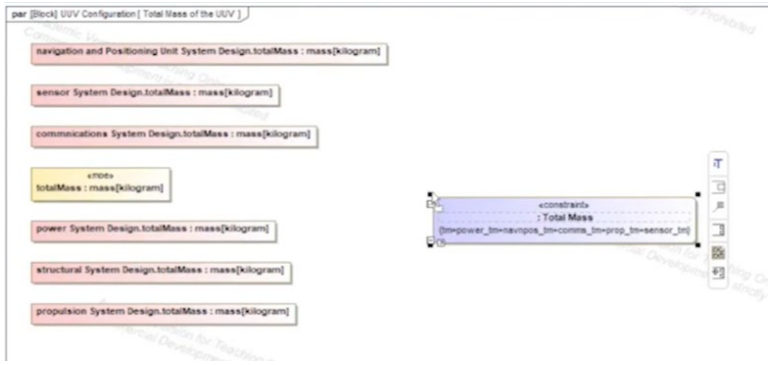




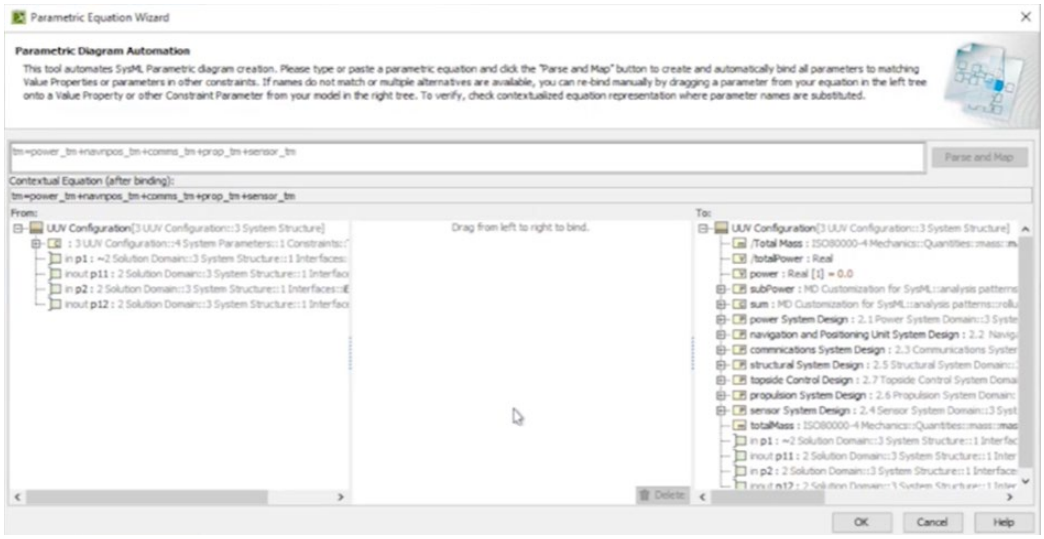
Locate Total Mass block by using the keypad press and hold ctrl, alt, F. This will automatically display a search window called Quick Find. Verify the Any Element is selected then enter total mass in the search window. Select Total Mass constraint block, this will automatically highlight the desired package in the Containment tree.



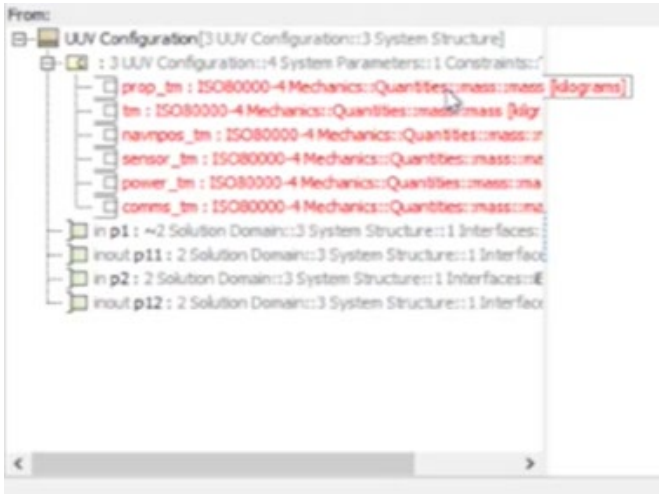
With Total Mass highlighted in the containment tree drag into the Total Mass of the UUV diagram pane.



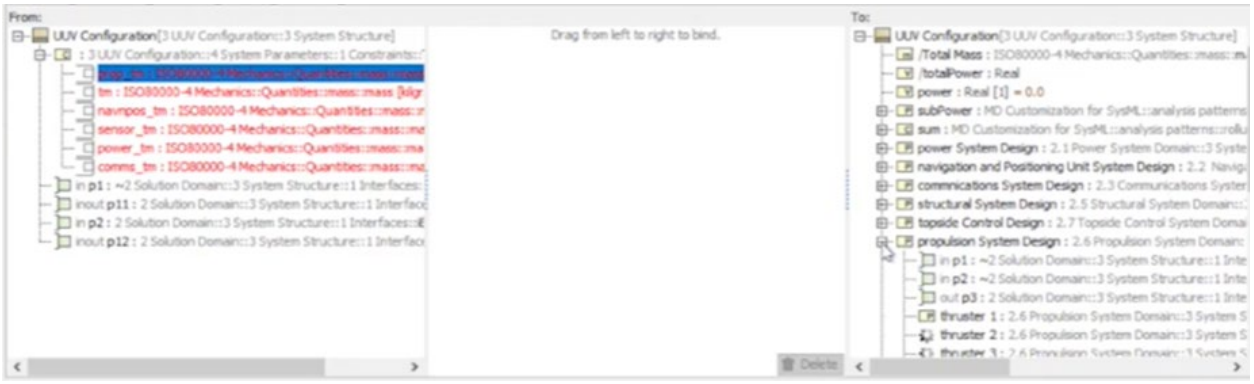
The Parametric Equation Wizard should automatically appear.



Expand 3 UUV Configuration.

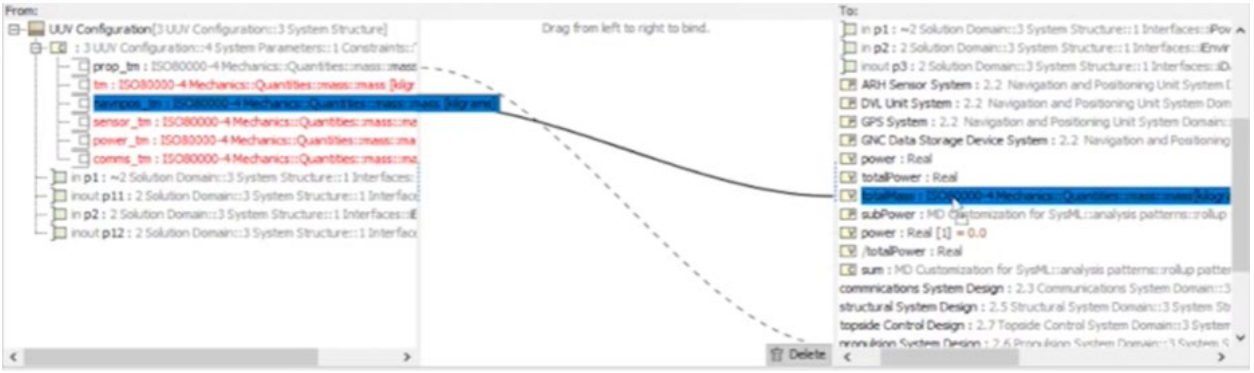


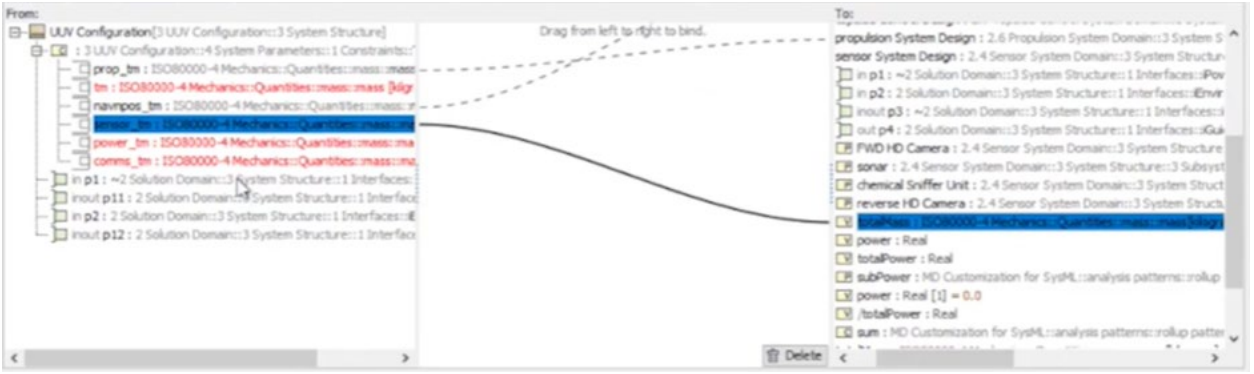
Each subsystem will be linked to their respected Total Mass parameter listed on the right side of the columns. Locate propulsion System Design on the right side and expand that section.

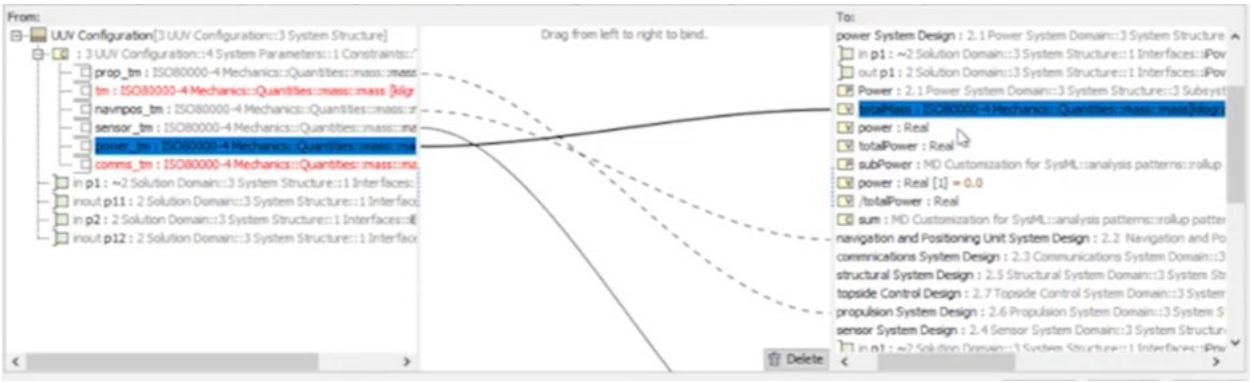


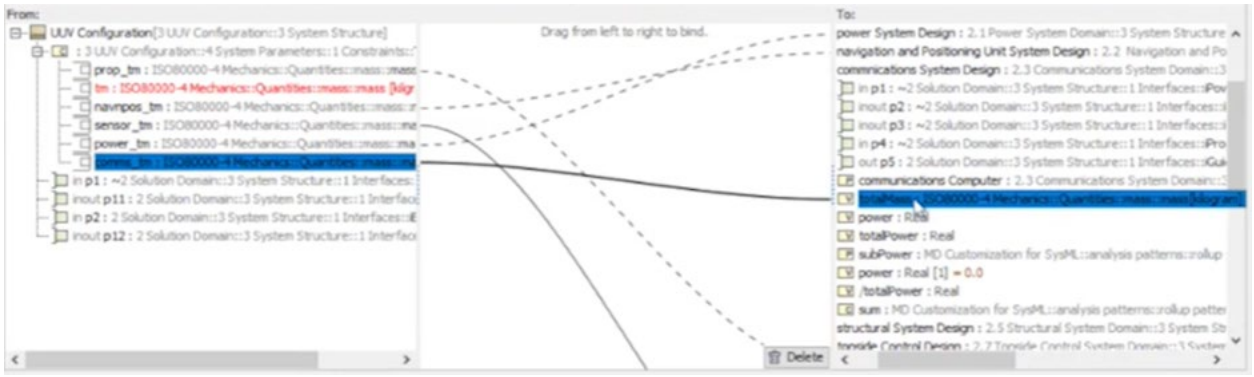
Locate totalMass, select prop_tm and drag from left to right to totalMass listed under propulsion System Design.

Repeat the last two steps for each remaining constraints (Navigation, Sensors, Power, and Communications). Then select OK.

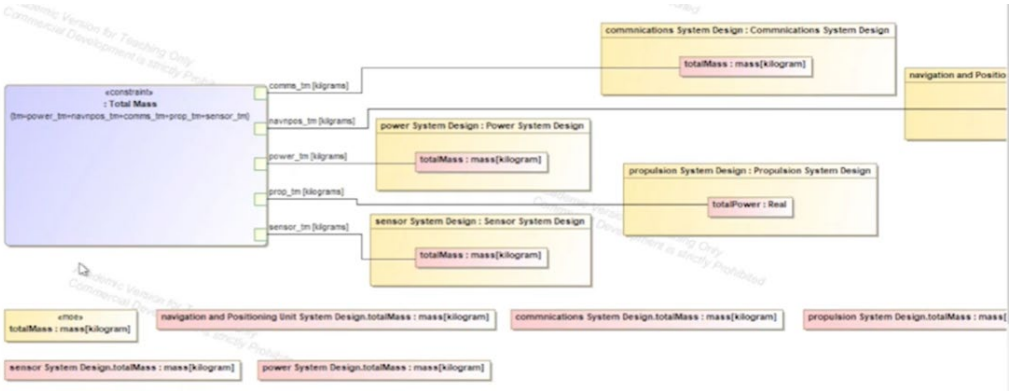








The following blocks should appear in the main diagram.



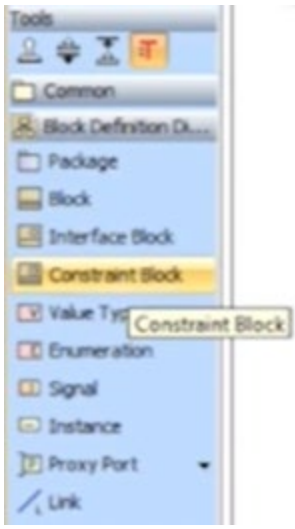
To link the totalMass constraint to the MOE remaining click the Total Mass constraint block to open the smart manipulator tool bar, select Display All Parameters.

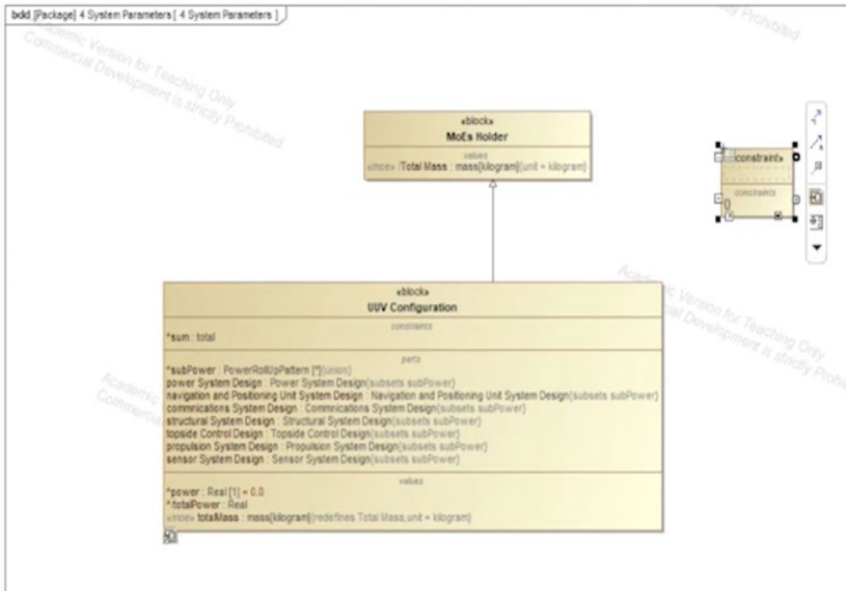
An additional port will now be available to click and open the tool bar, select Binding Connector.

With the Binding Connector tool mode active drag the line to the totalMass MOE then select.

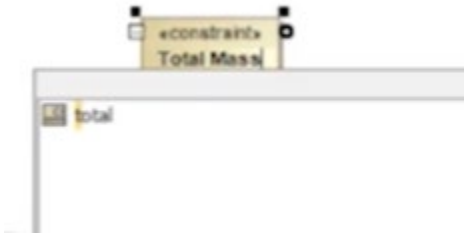
Creating the formula to calculate for Total Mass

Within 4 System Parameters, Select Constraint Block from the tool bar and click in an open area in the diagram to create the block.





Type Total Mass in the block, then hit enter.



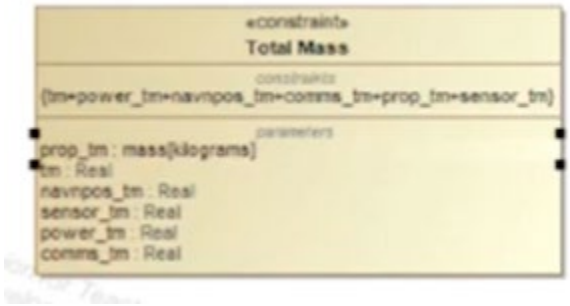
Click on the empty expression of the block.

Enter the formula total mass = power + navigation + communications + sensor (tm=power_tm+navnpos_tm+comms_tm+prop_tm+sensor_tm)

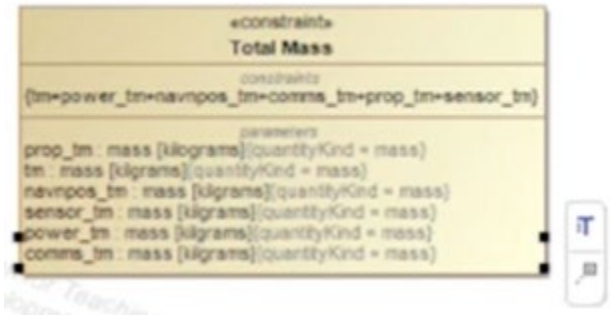
Specify constraint parameters

Click the Total Mass constraint block to open the tool bar. Select Parse and Create Parameters.

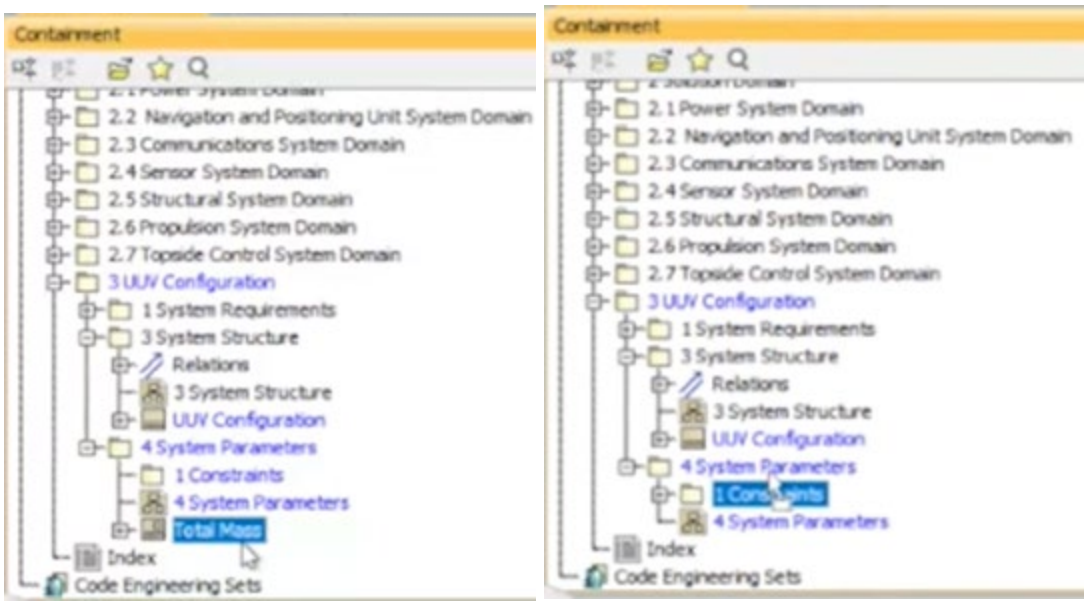
Each of the parameters listed will need to change from Real to mass kilograms. Click prop_tm and erase Real, then type mass [kilograms]. Then hit enter.



Repeat this step for each of the parameters.

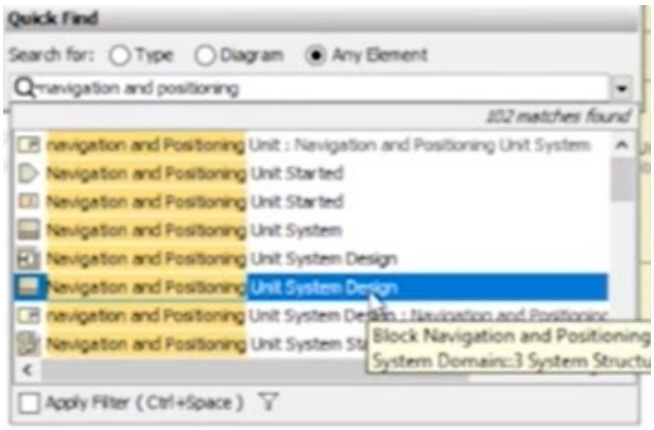


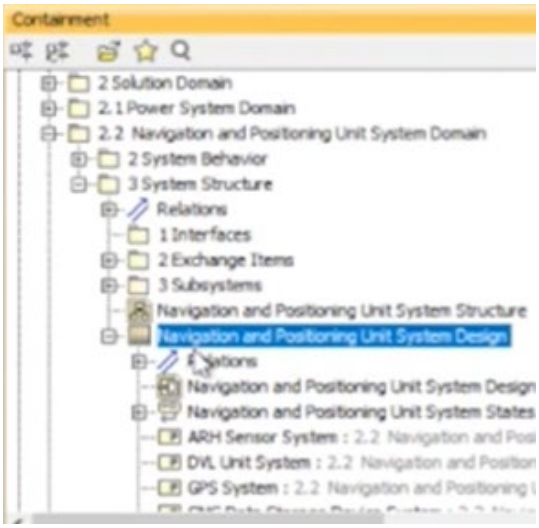
Click and drag Total Mass from the Containment tree to 1 Constraints to keep organized.



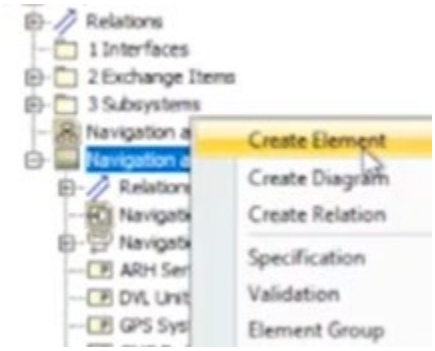
Specify the subsystem parameters that capture the total mass

Locate Navigation and Positioning block by using the keypad press and hold ctrl, alt, F. This will automatically display a search window called Quick Find. Verify the Any Element is selected then enter navigation and positioning in the search window. Select Navigation and Positioning Unit System Design constraint block, this will automatically highlight the desired package in the Containment tree.

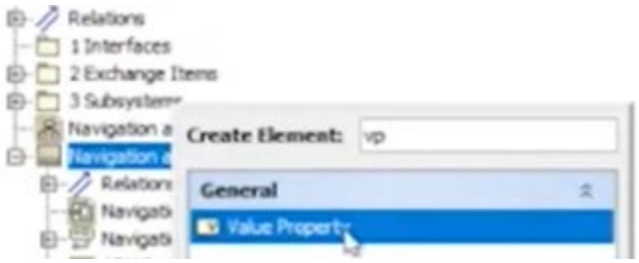




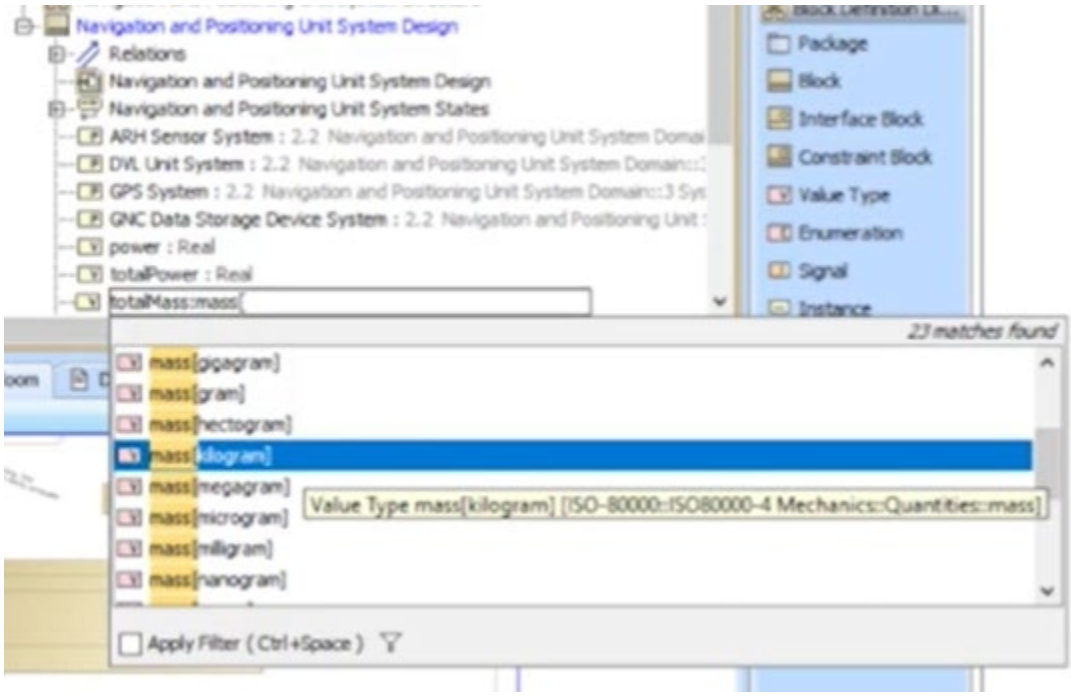
Right click Navigation and Positioning Unit System Design, select Create Element.



Type vp in the search bar, locate and select Value Property.



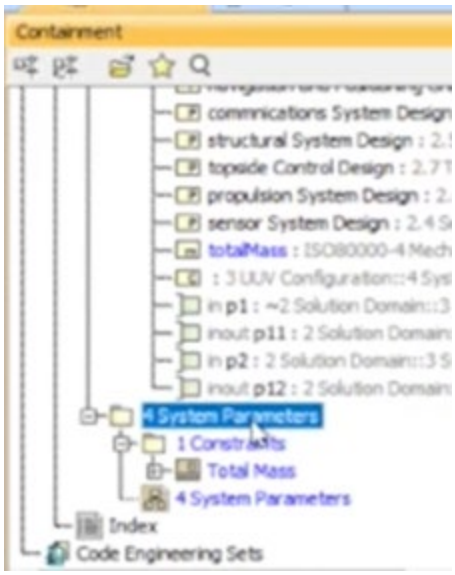
Type totalMass:mass [then ctrl + spacebar on the keyboard to bring up a selection list. Locate and select the mass kilogram option listed below, then hit enter.



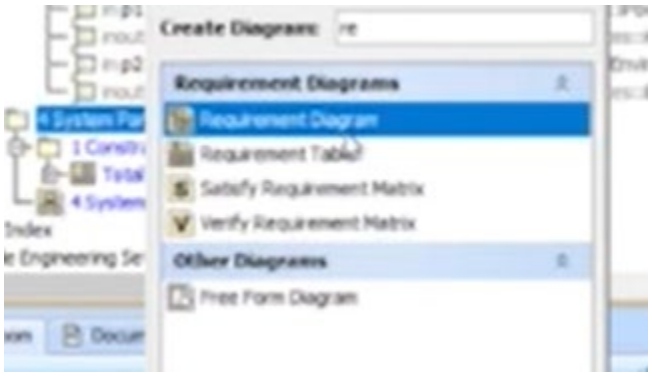
Repeat the last 4 steps for the remaining systems of the UUV.

Creating automated requirements verification

Locate in the Containment tree 4 Systems Parameters. Right click select Create Diagram.



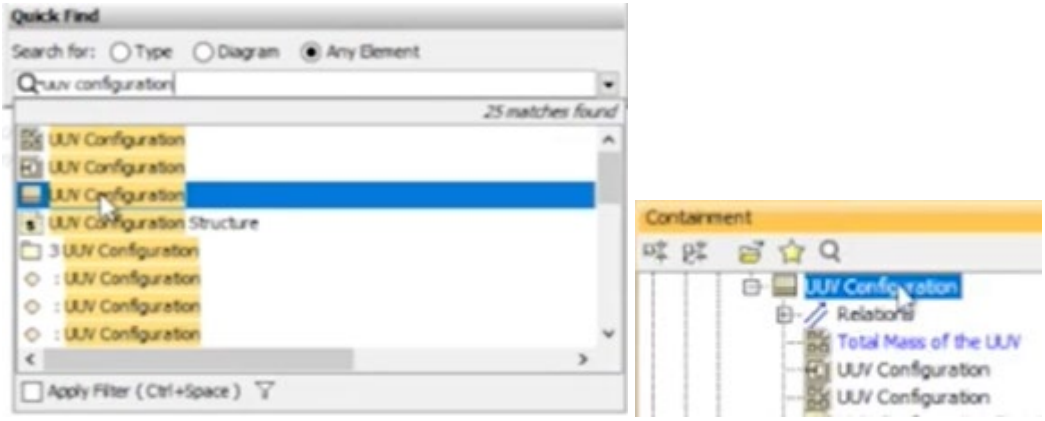
Type re in the search window to locate and select Requirement Diagram. Then hit enter.



Type Total Mass Verification, then hit enter.

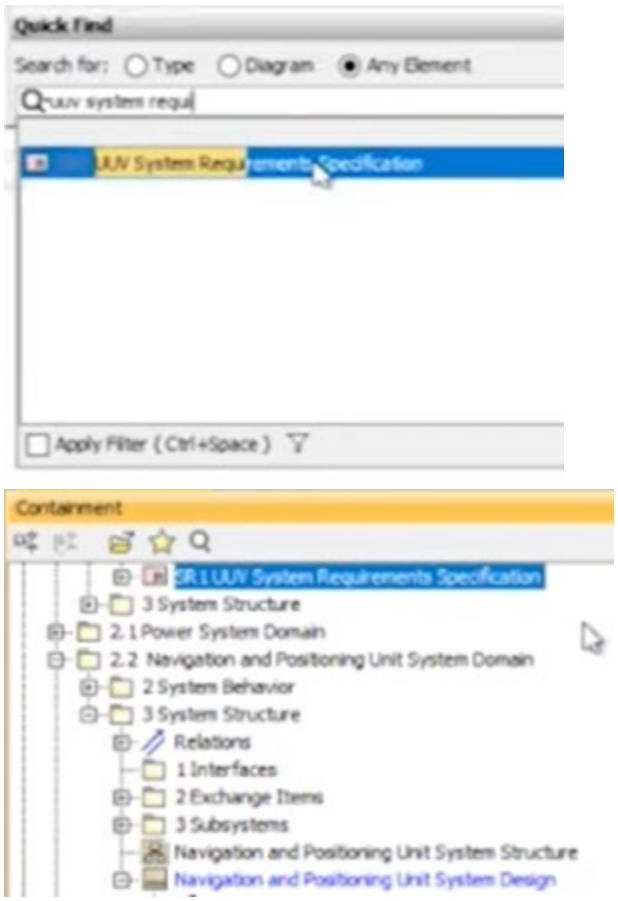
Locate UUV Configuration block by using the keypad press and hold ctrl, alt, F. This will automatically display a search window called Quick Find. Verify the Any Element is selected then enter UUV configuration in the search window. Select UUV

Configuration constraint block, this will automatically highlight the desired package in the Containment tree.



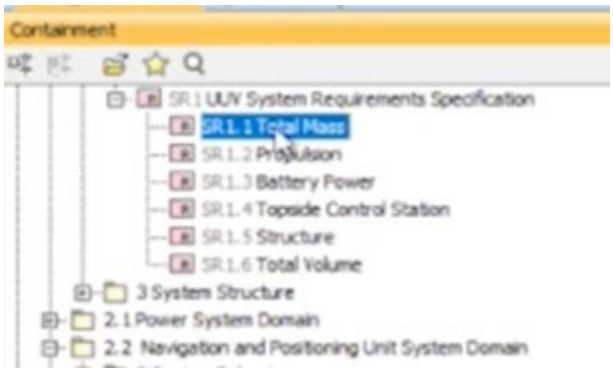
With the UUV Configuration package highlighted drag onto the Total Mass Verification diagram pane.

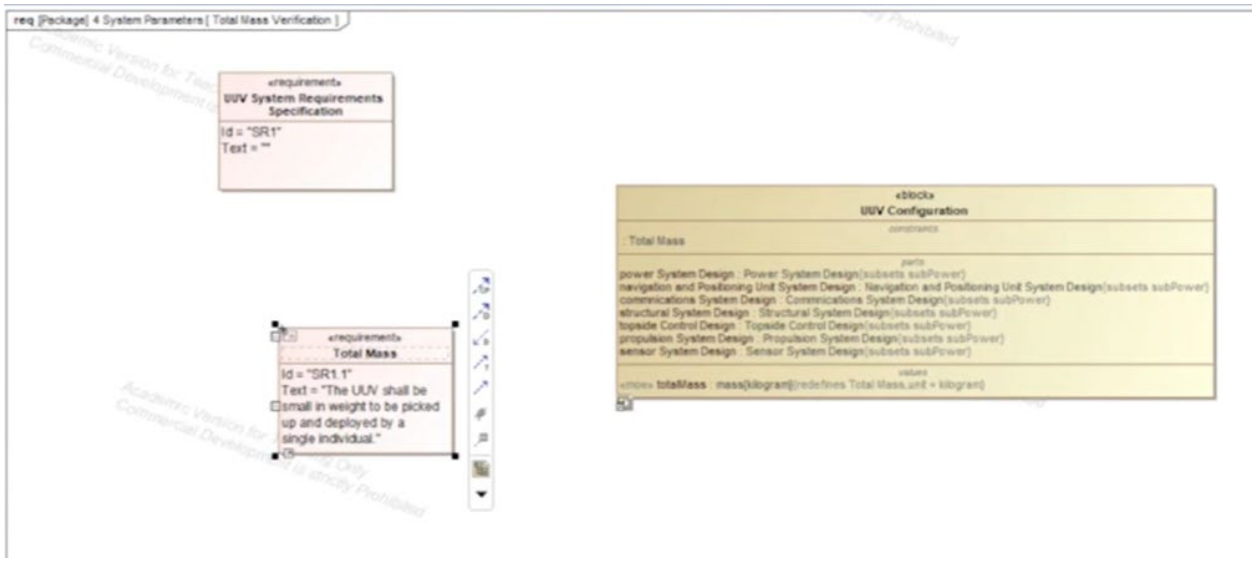
Locate UUV System Requirements Specification block by using the keypad press and hold Ctrl-Alt-F. This will automatically display a search window called Quick Find. Verify the Any Element is selected then enter UUV System Requirements Specification in the search window. Select UUV Configuration System Requirements Specification block, this will automatically highlight the desired package in the Containment tree.



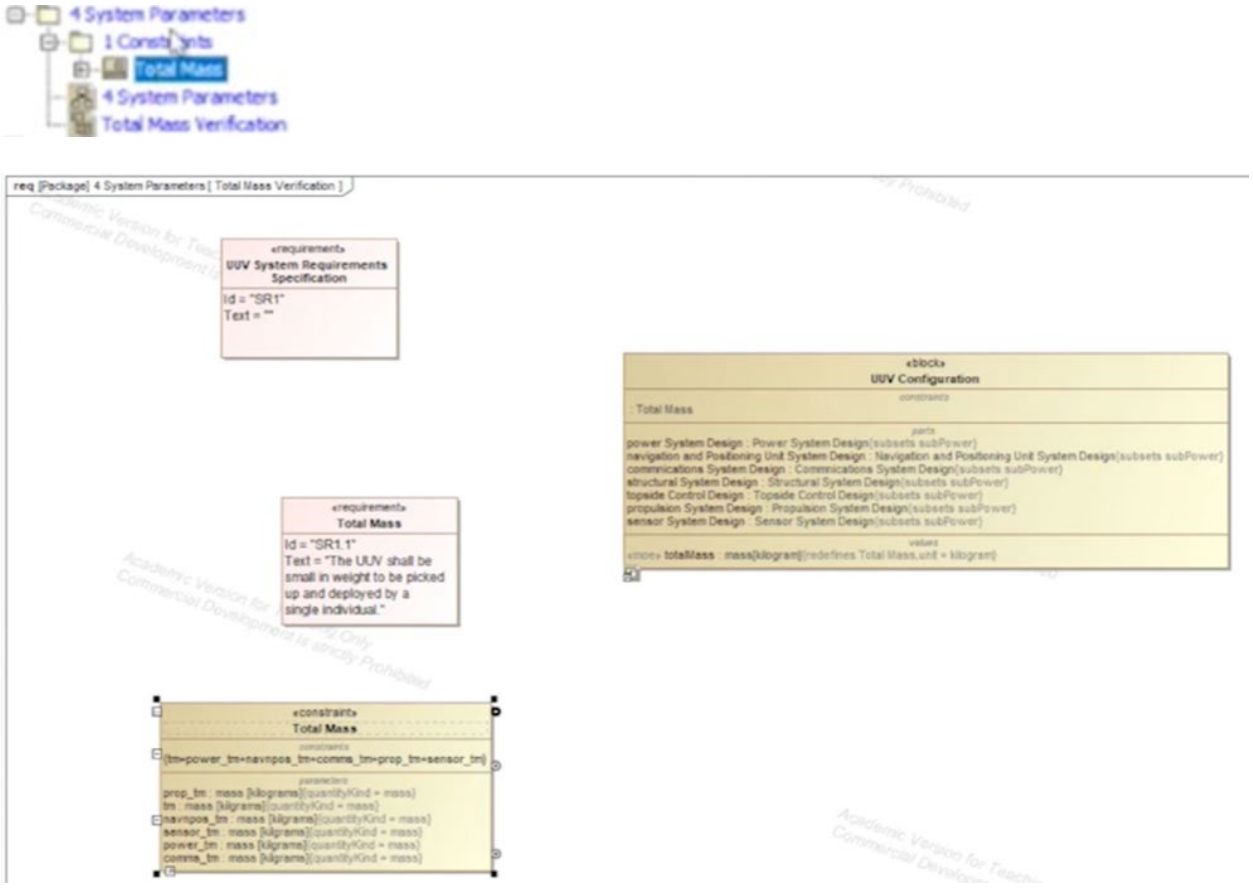
With the UUV System Requirements Specification package highlighted drag onto the Total Mass Verification working window.

Expand the UUV System Requirements Specification package in the containment tree, locate, highlight and drag Total Mass into the working window.

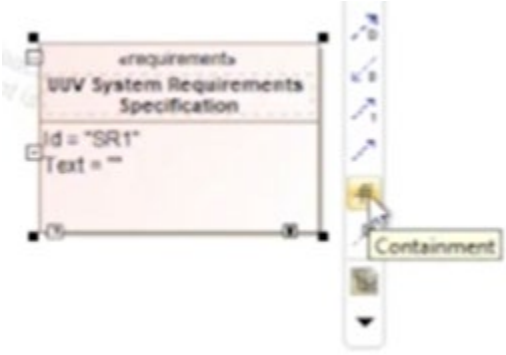




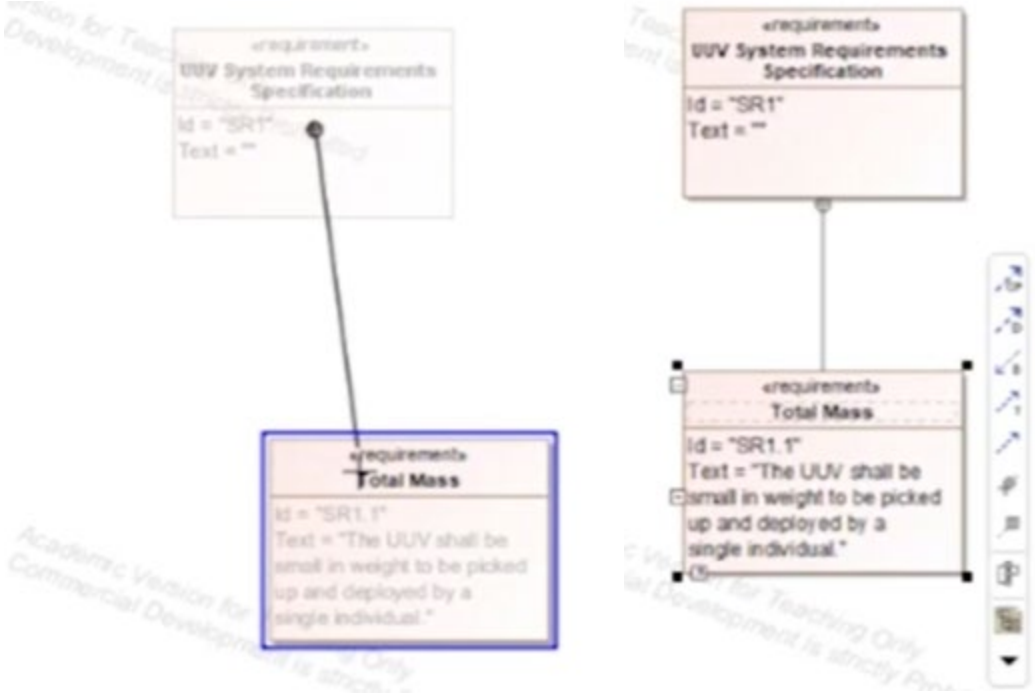
In the containment scroll down to find the 4 Systems Parameters tab, expand and locate, highlight and drag Total Mass constraint package into the working window.



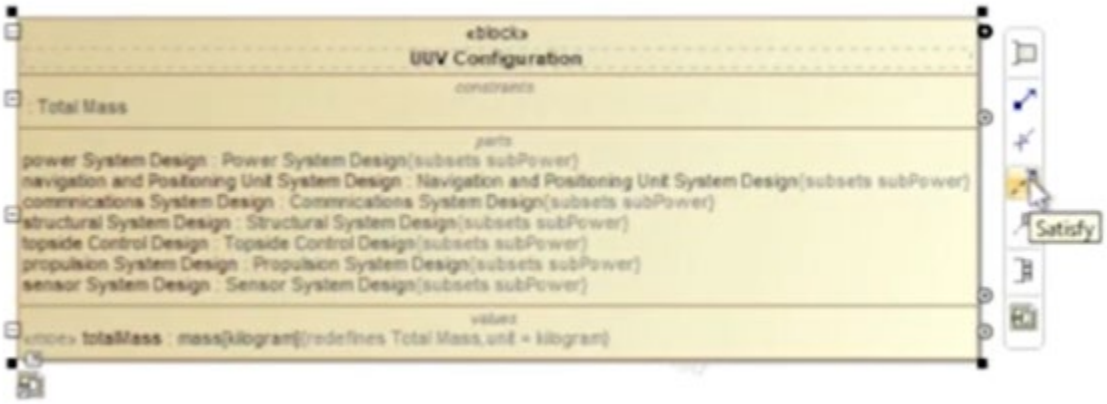
Click the UUV System Requirements Specification block to open the smart manipulator tool bar, select Containment.



With the Containment mode active drag the line over to the Total Mass requirements block and select.



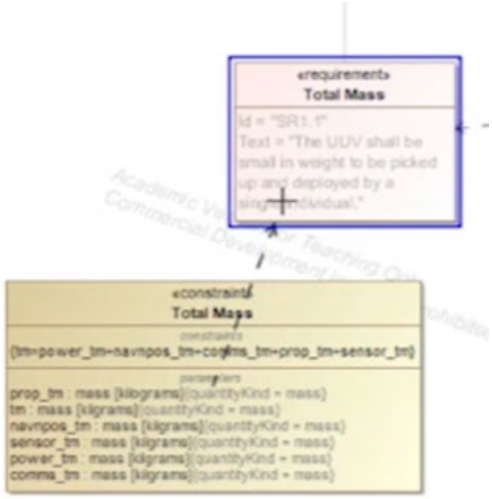
Click the UUV Configuration block to open the smart manipulator tool bar, select Satisfy.



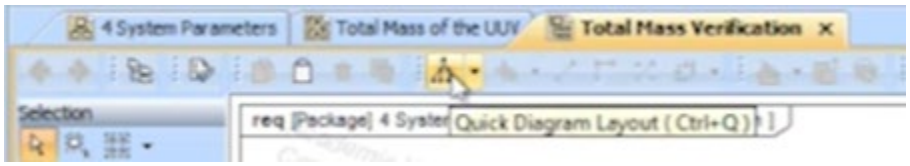
With the Satisfy mode active drag the arrow to the Total Mass Requirements block, then select.

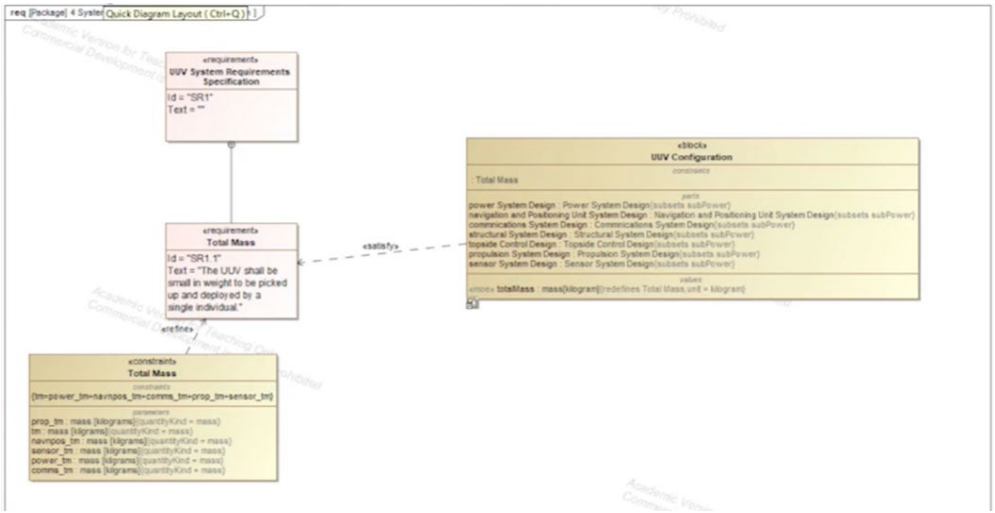
Click on the Total Mass constraints block to open the smart manipulator tool bar, select Refine.

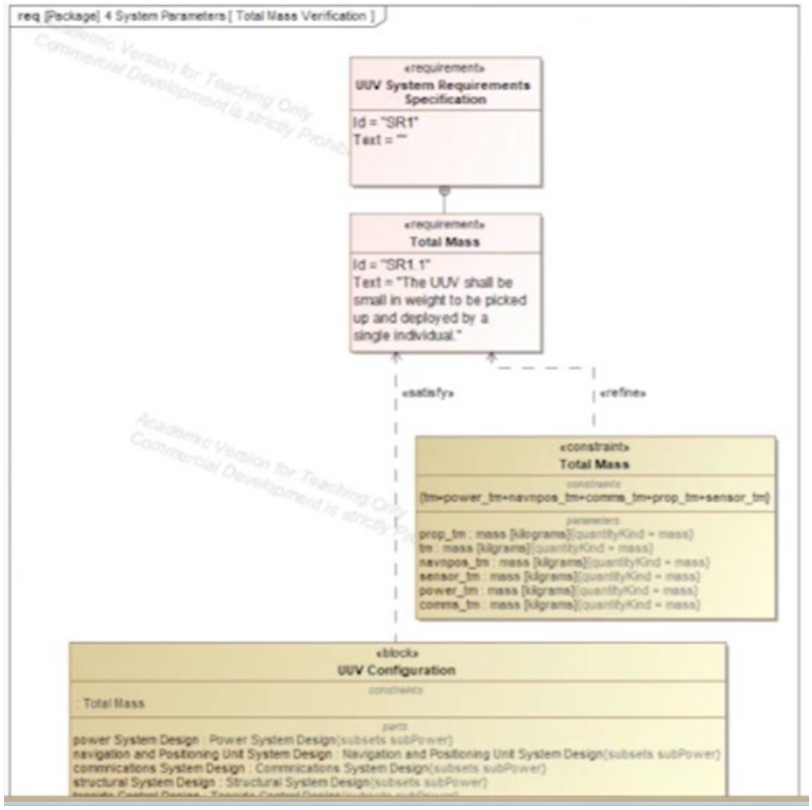
With the Refine mode active drag the arrow to the Total Mass block and select.



Use the Quick Diagram Layout button to organize the diagram.







This concludes the System Parameters model.

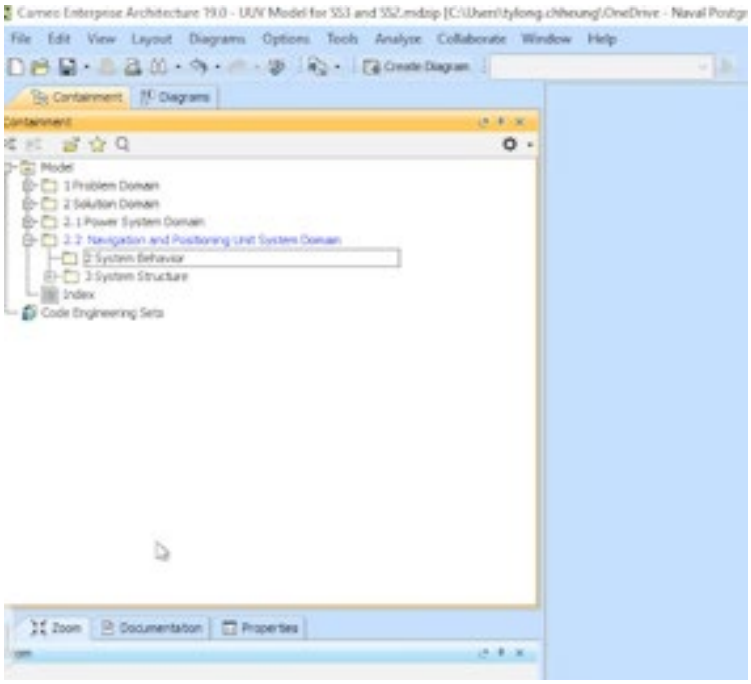
SS2: Subsystem Behavior Tutorial

Background:

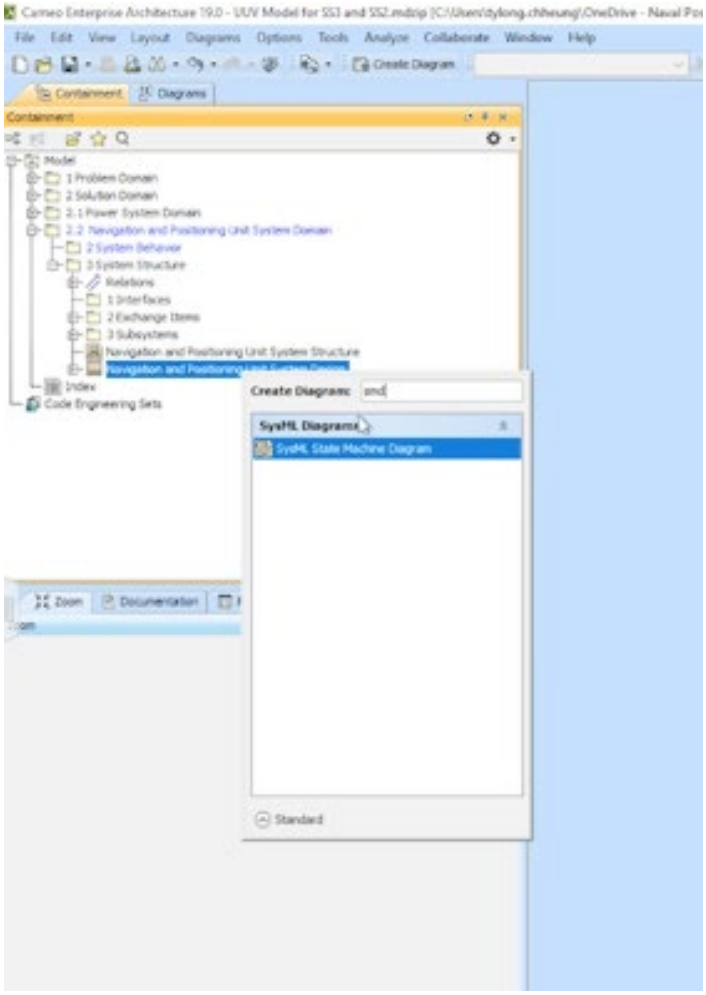
In this section, the system and subsystem behaviors are defined and should satisfy functional system requirements. The solution architecture for the subsystems is developed further and integrated into one whole system, while meeting functional requirements. The purpose of this is to determine how the subsystems will communicate with each other through compatible ports. Using a SysML state machine diagram and activity or sequence diagrams in combination, the system behavior can be understood. Sequence of events are specified through the different states of the machine: entry, do, and exit behaviors. The combination of these identifies the internal behaviors of the subsystems and which signals are compatible.

Step-By-Step Guide:

In this video we will cover the behavior model of the *Navigation and Position Unit System*. We will start with by organizing the overall model for SS2. Therefore, we will go

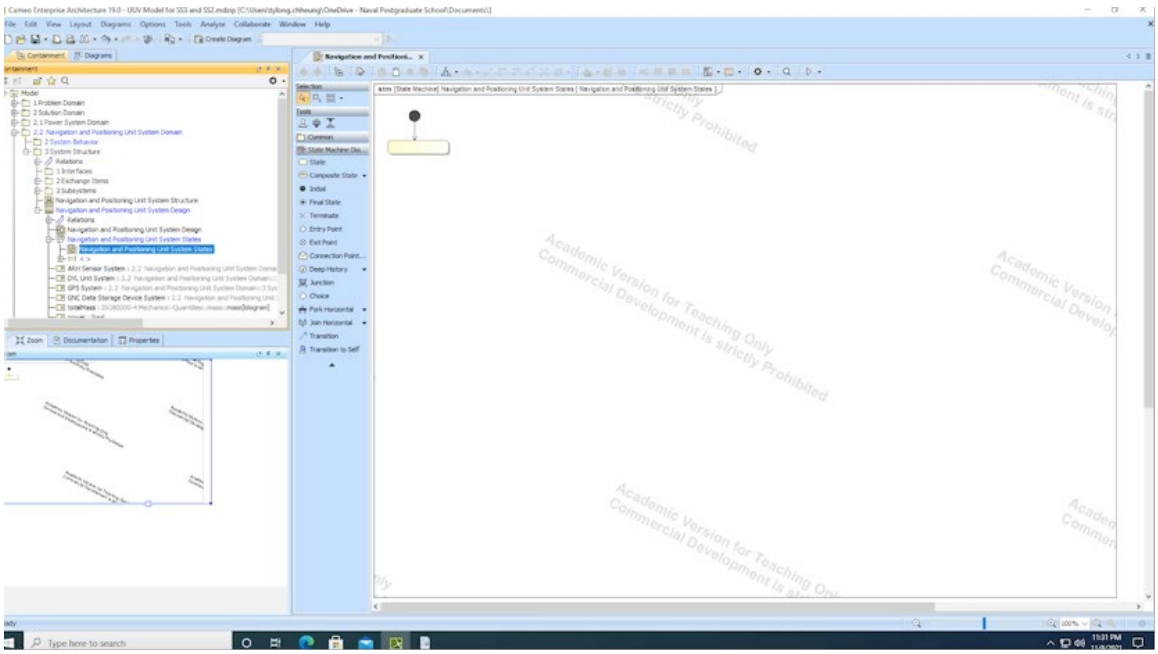


to the containment tree of the left-hand side and right click on the *2.2 Navigation and Positioning Unit System Domain*. For the drop-down menu we will select *Create Element* button. Then from the *Create Element* popup menu we will be selecting the *Package* option. We will label the new package as *2 System Behavior*.

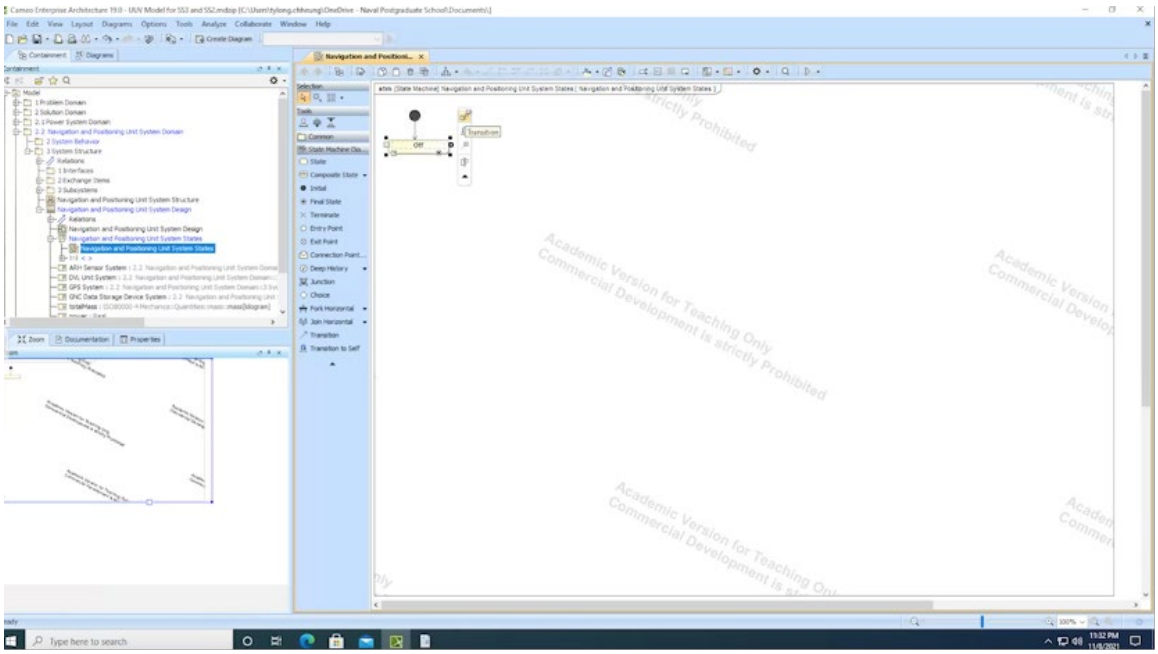


Once we have that completed we will expand the *3 System Structure* package and scroll down until we find the *Navigation and Positioning Unit System Design* block. We will then right click the *Navigation and Positioning Unit System Design* block and the drop-down menu we will select *Create Diagram* button. Then from the *Create Diagram* popup menu in the search bar we will type *smd* and from the result list select *SysML State Machine Diagram* option.

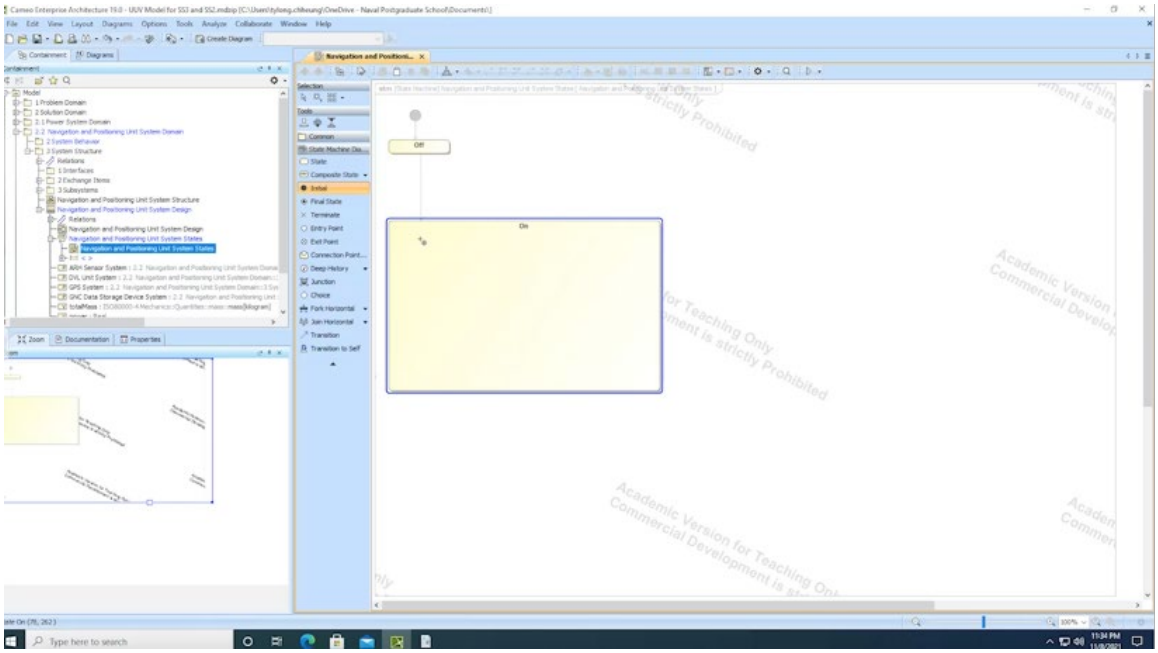
This will create a state machine diagram and we are going to label it as *Navigation and Positioning Unit System States*.



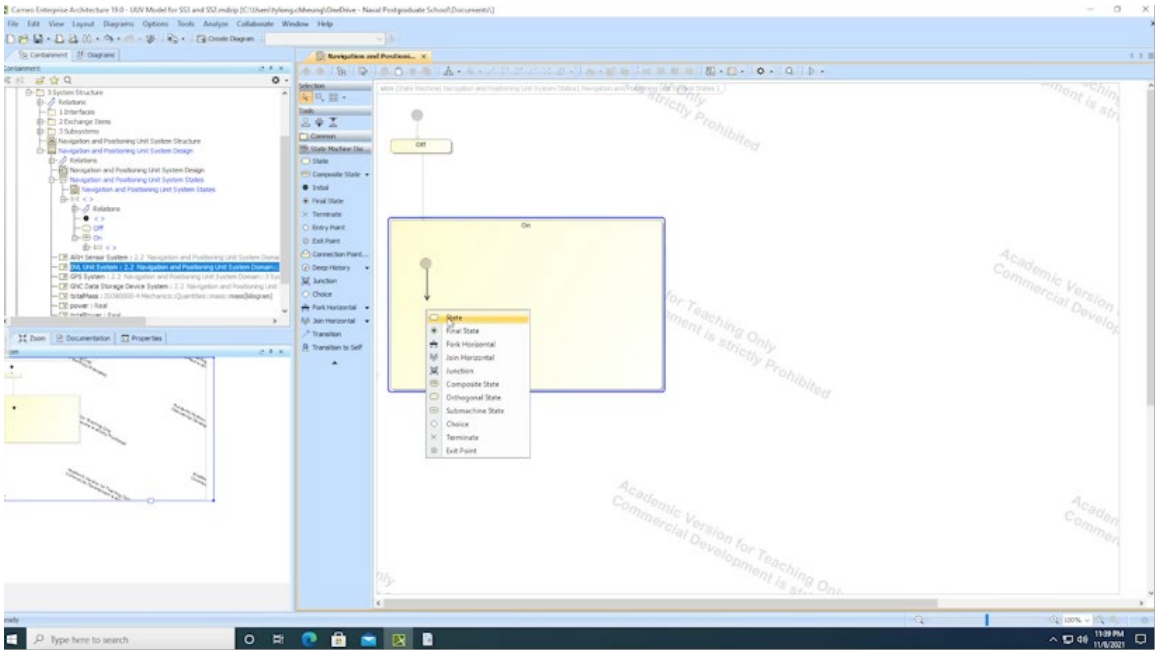
Once you have that started, we will head over to the diagram pane and select the first state and label it as *Off*. Then in the smart manipulator tool bar we will select the *Transition* button and drag it to an empty space in the diagram plane.



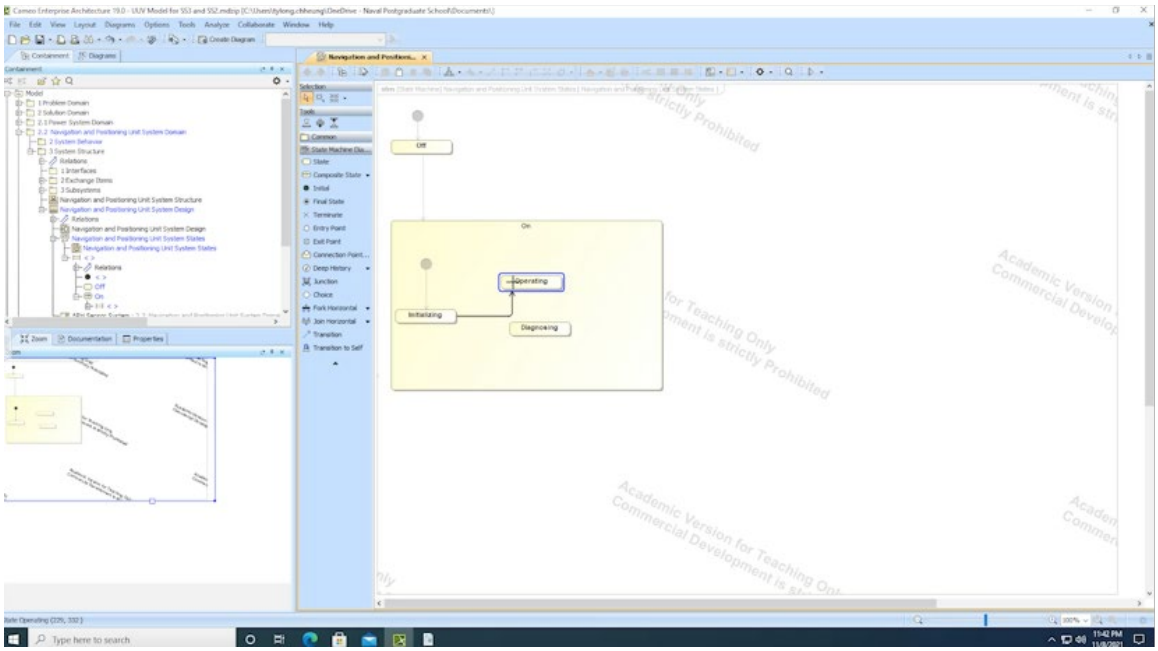
We will then label the state as *On*. Then we will expand the *On* state a bit bigger in order to list more functions while it's on. Now to capture more functions in the ON position we will go to the diagram pallet and select the *Initial* button and drag it to the *On* state.



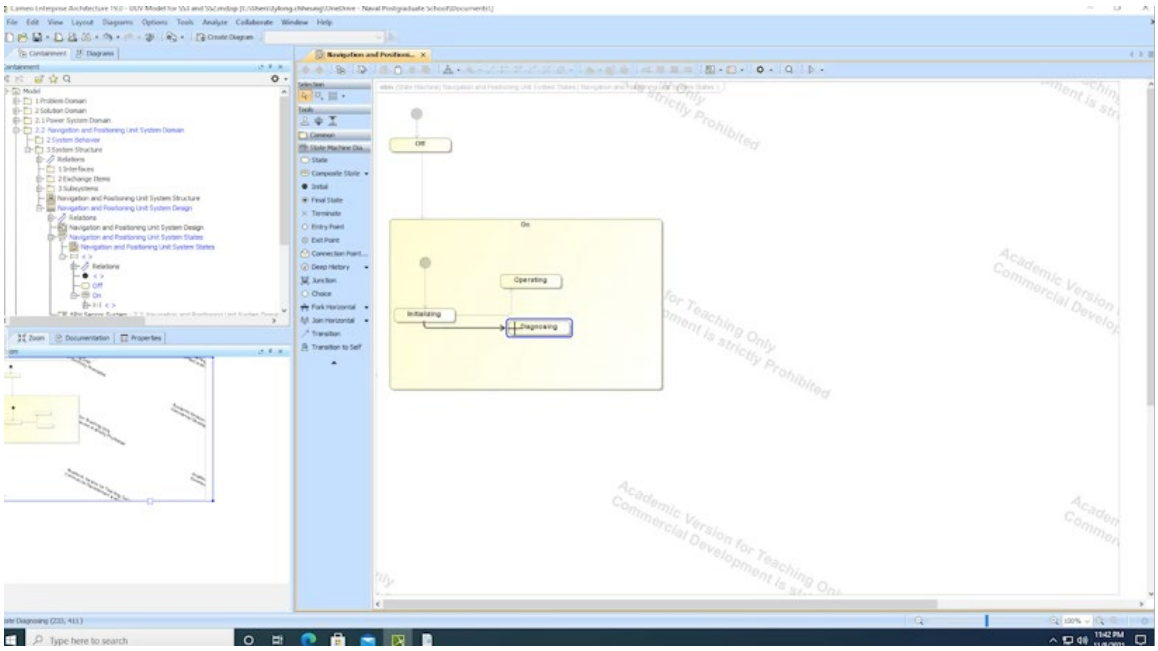
From the manipulator tool bar of the initial node we will select the *Transition* button and drag it to an empty space in the *On* state and right click and select the *State* option in order to create a new state.



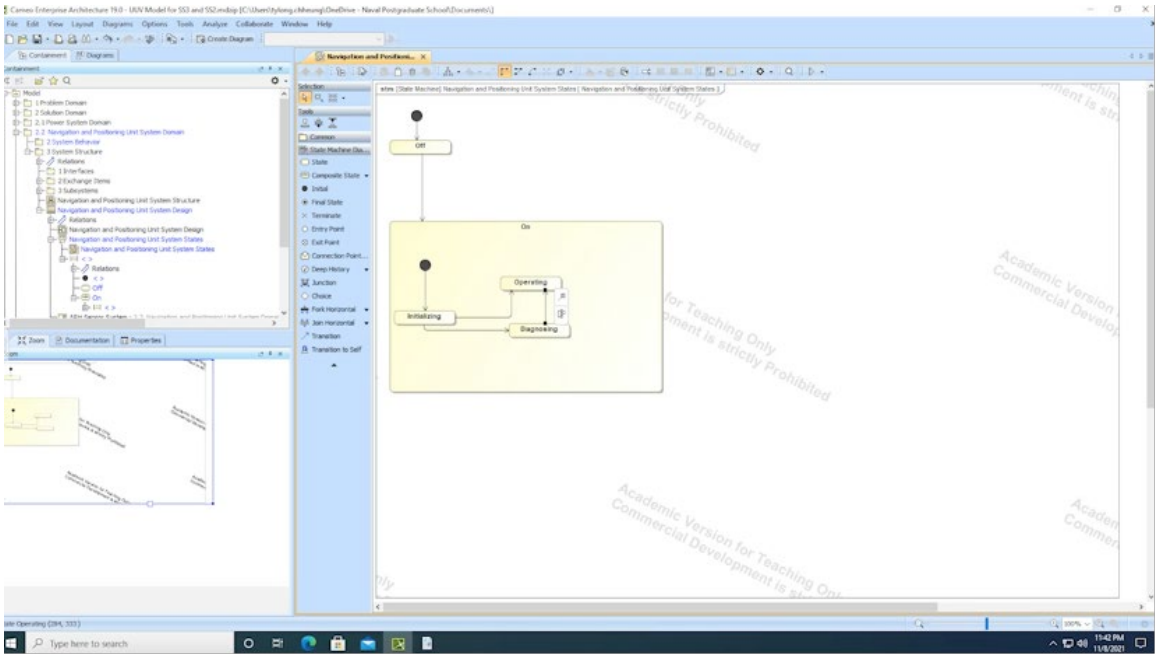
We will label the new state as *Initializing* for the initializing state of the system. We will head over to the diagram pallet and selecting the *State* button and dragging it to an empty space in the *On* state. We will then label the state as *Operating*. And we will repeat the last step by going over to the diagram pallet and selecting the *State* button and dragging it to an empty space in the *On* state. We will then label the state as *Diagnosing*. We will then click on the *Initializing* state and from the manipulator tool bar we will select the *Transition* button and drag it to the *Operating* state.



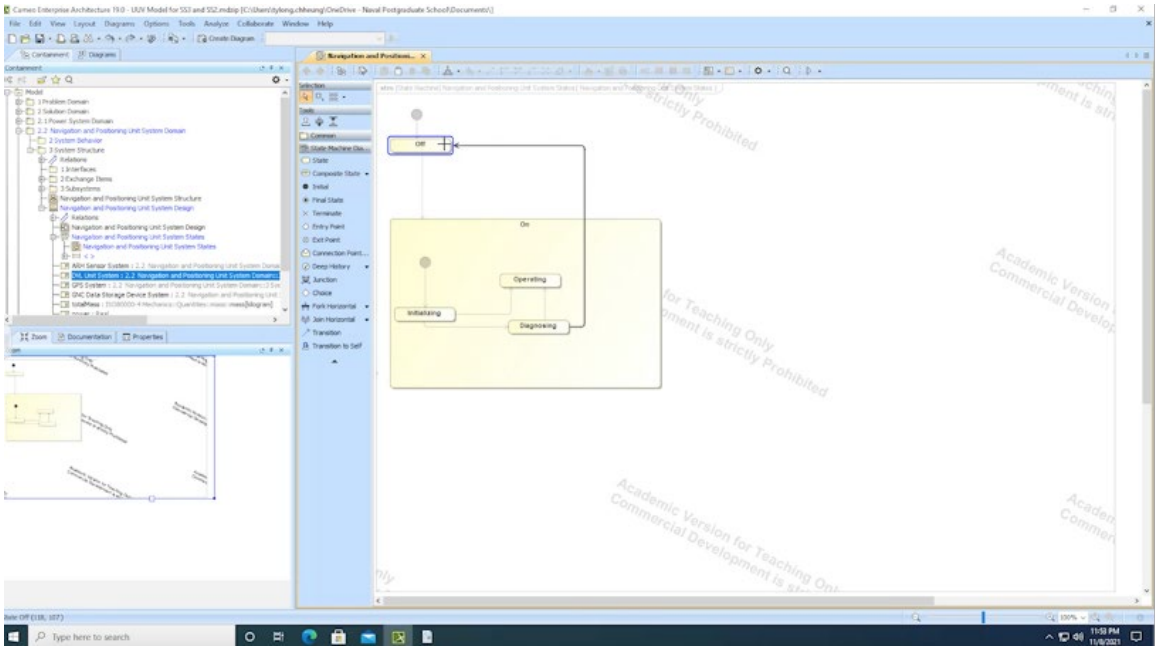
We will go back to click on the *Initializing* state and from the manipulator tool bar we will select the *Transition* button and drag it to the *Diagnosing* state.



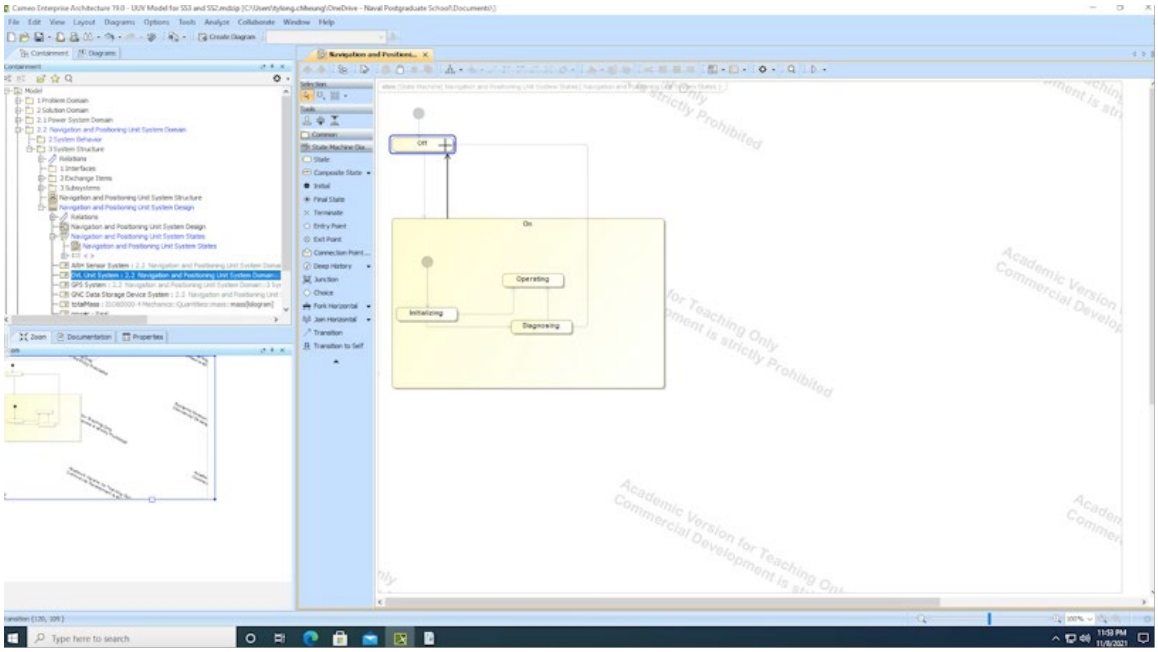
And we click on the *Diagnosing* state and from the manipulator tool bar we will select the *Transition* button and drag it to the *Operating* state.



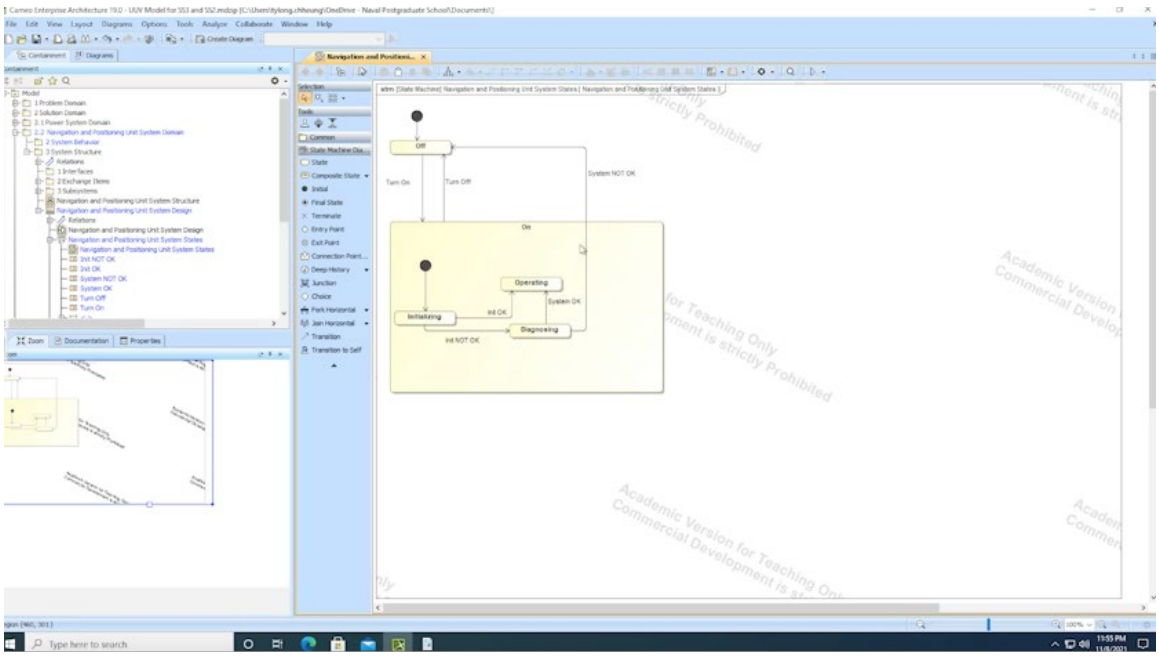
The last two transitions we have are the *Diagnosing* state that transitions to the *Off* state. Therefore, we click on the *Diagnosing* state and from the manipulator tool bar we will select the *Transition* button and drag it to the *Off* state.



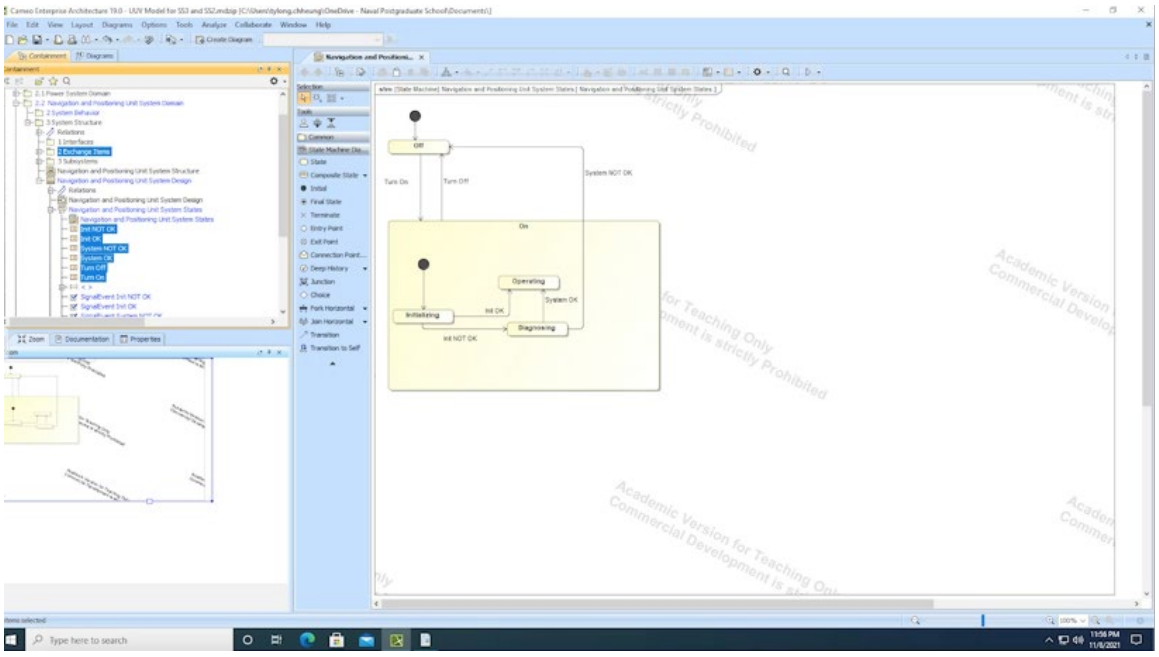
The other transition we have left is the overall *On* state will transition to the *Off* state. Therefore, we will click on an empty space in the *On* state and from the manipulator tool bar we will select the *Transition* button and drag it to the *Off* state.



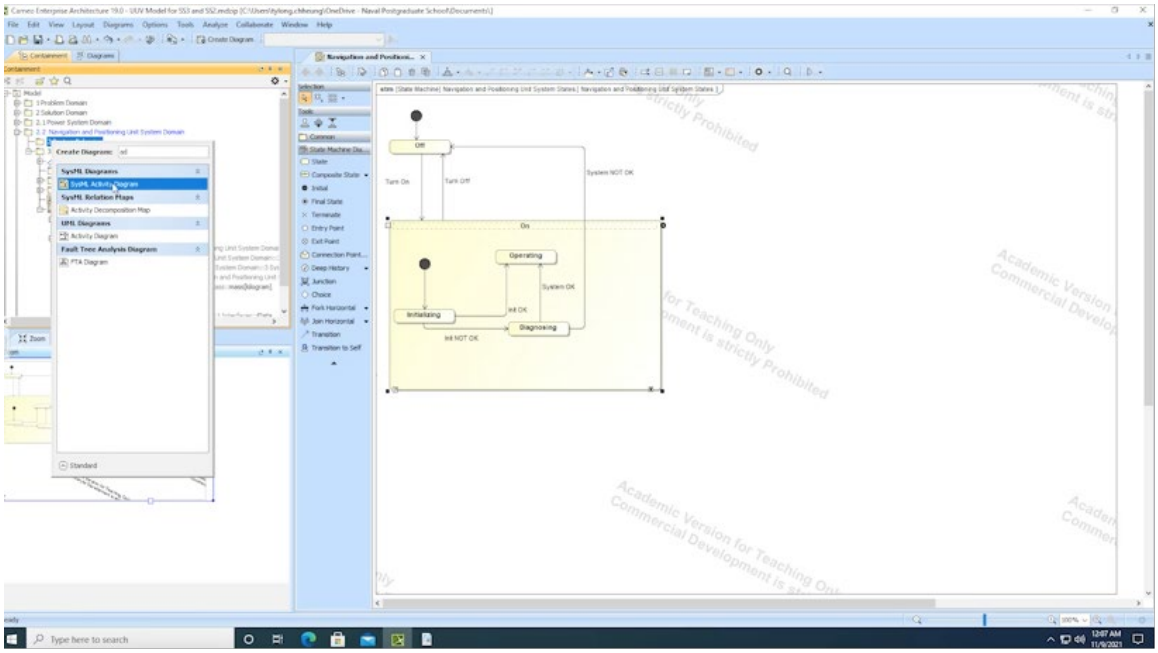
Next, we are moving into specifying the event occurrences between each transition. So let's select the off to on transition type in *Turn On*. After if you head over to the containment tree, you will be able to see that it created a signal. We will continue to do the same by selecting the on to off transition type in *Turn Off*. Repeat the step to specify the event occurrences between the other transitions. For example, for the transition between the *Initializing* state and the *Diagnosing* state, we will type in *Int NOT OK*. And for the transition between the *Initializing* state and the *Operating* state, we will type in *Int OK*. Now the transition between the *Diagnosing* state and the *Operating* state, we will type in *System OK*. And finally, the transition between the *Diagnosing* state and the *Off* state, we will type in *System NOT OK*.



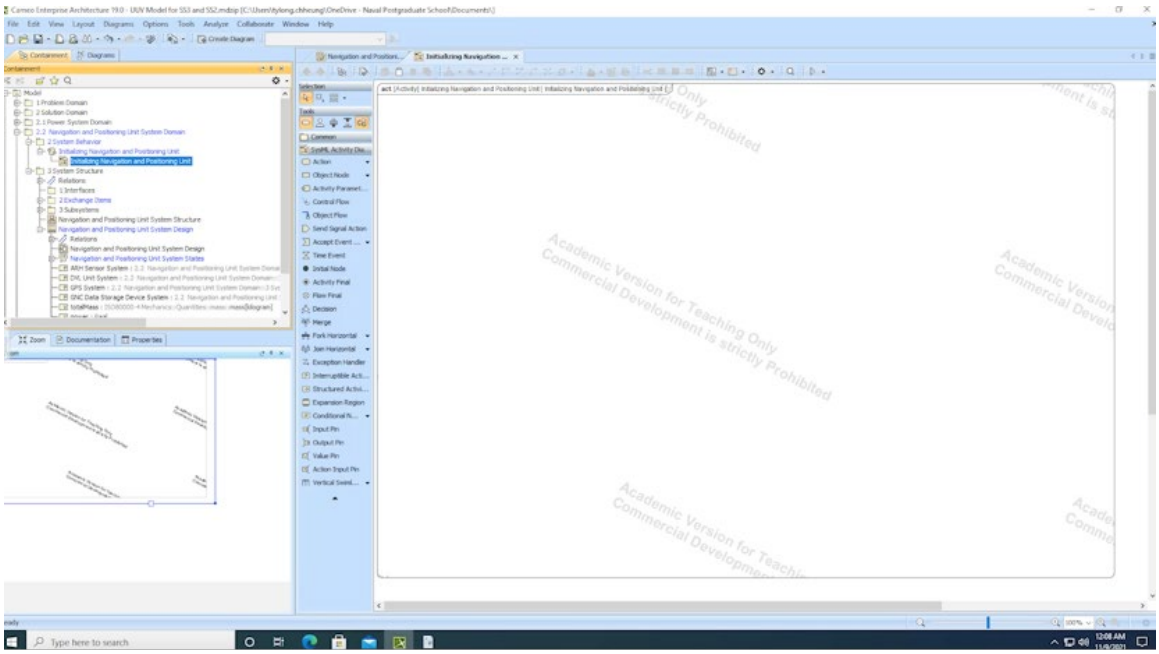
You can see in the containment tree that every time you label a transition it created a signal. We will now organize our containment tree a bit better by selecting all the newly created signals and drag it into *2 Exchange Items* package.



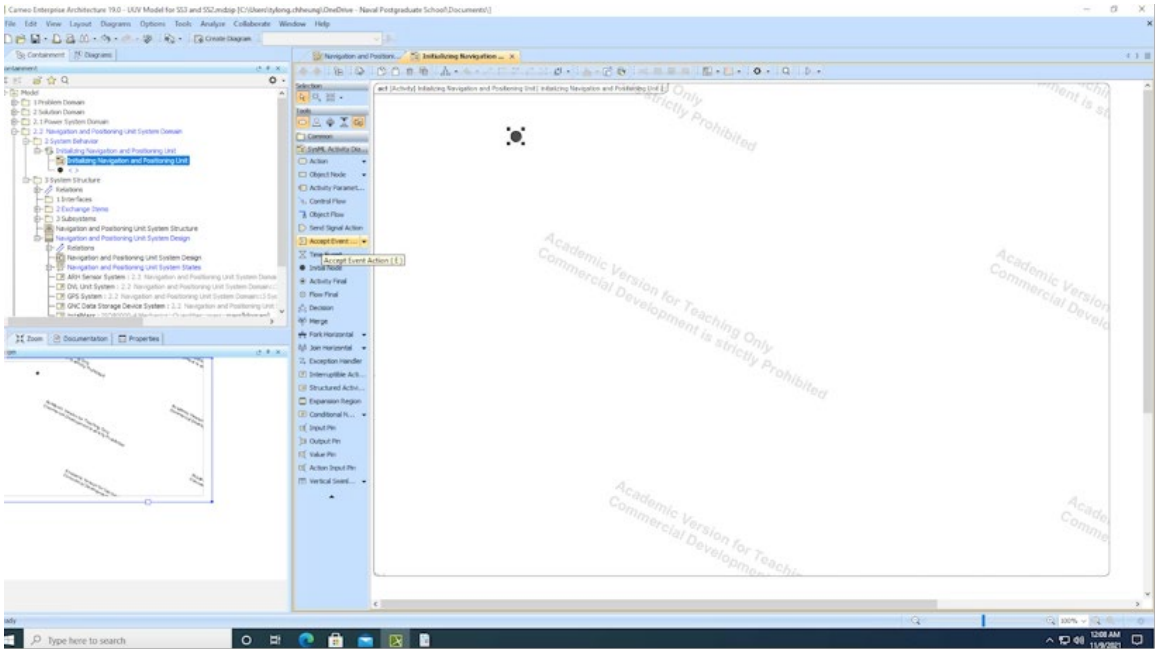
Once we have all the signals moved to the *2 Exchange Items* package, we can start specifying the behavior of each state. We will first with the *Initializing* state. We will head over to the containment tree and selecting the *2 System Behavior* package. From the drop-down menu we will be selecting the *Create Diagram* button. Then from the *Create Diagram* popup menu in the search bar we will type *ad* and in the result list we will select the *SysML Activity Diagram* option.



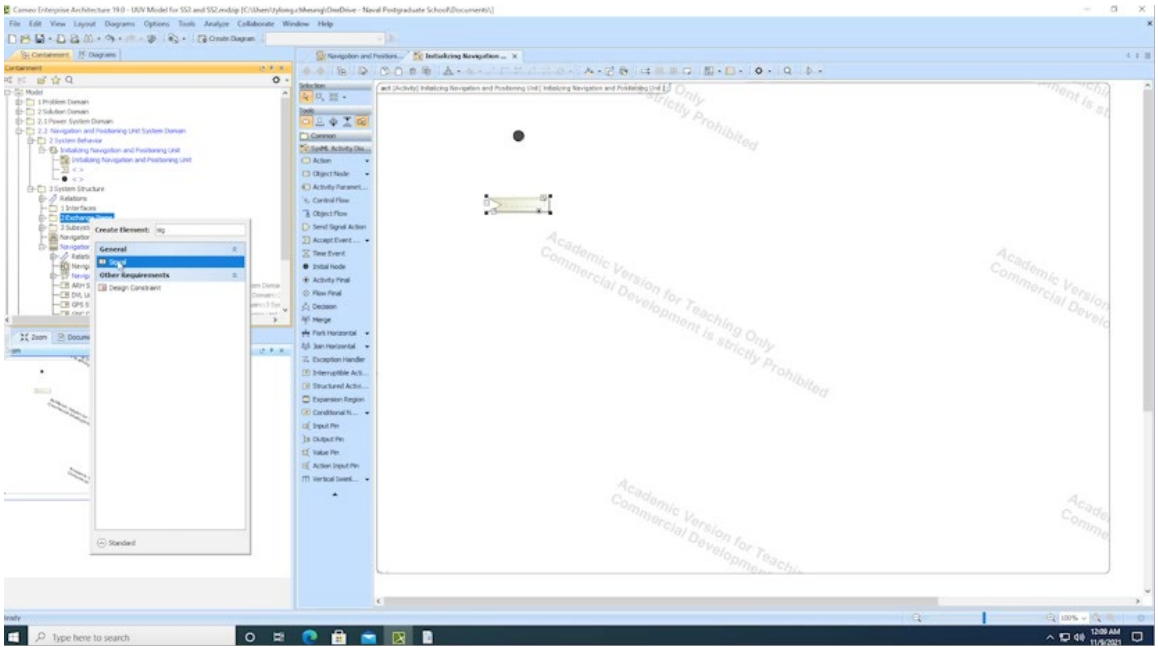
We will then label the newly created activity diagram as *Initializing Navigation and Positioning Unit*.



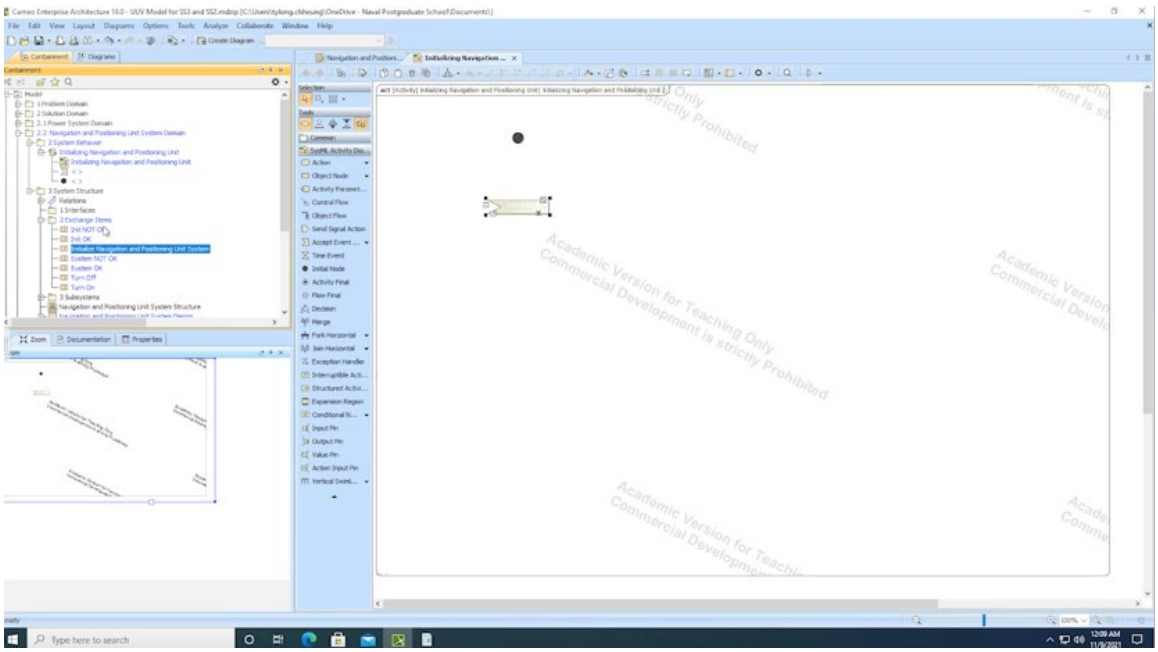
Inside the *Initializing Navigation and Positioning Unit* activity diagram we start by going to the diagram pallet and selecting the *Initial Node* option and drag it to an empty space in the activity diagram. Then going back to the diagram pallet, we will select the *Accept Event Action* option drag it to an empty space in the activity diagram.



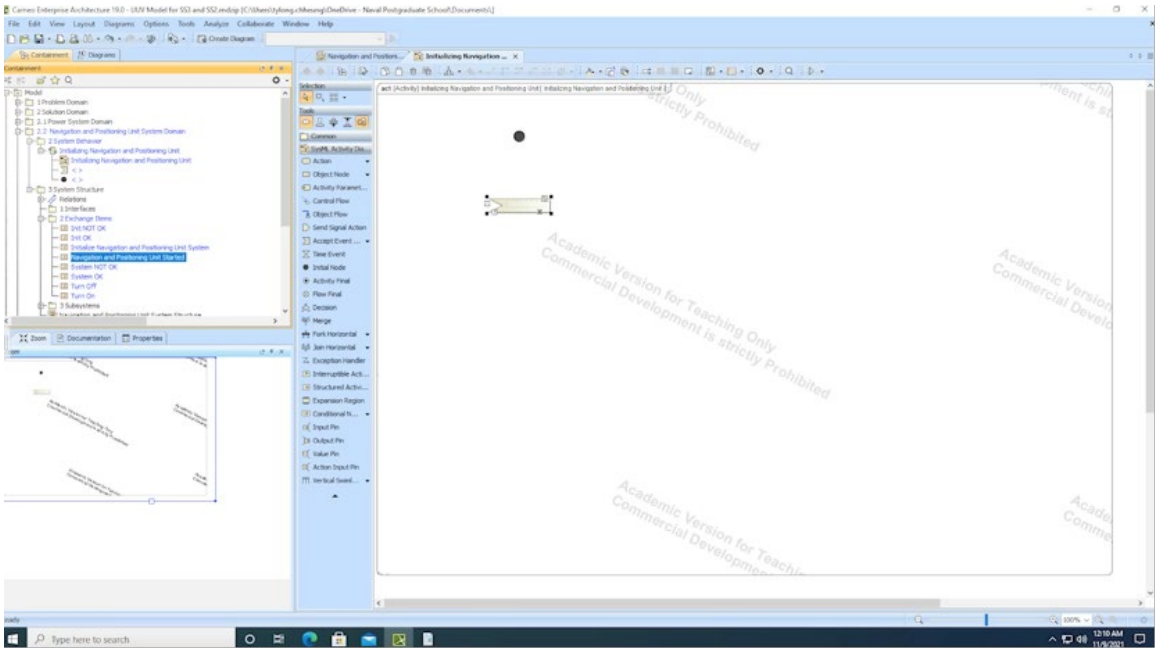
We will then head over to the containment tree on the left-hand side and select *2 Exchange Items*. We will right click and from the drop-down menu we will select the *Create Element* option. Then from the *Create Element* popup menu we will go to the search bar and type *sig* and from the result list we will select the *Signal* option.



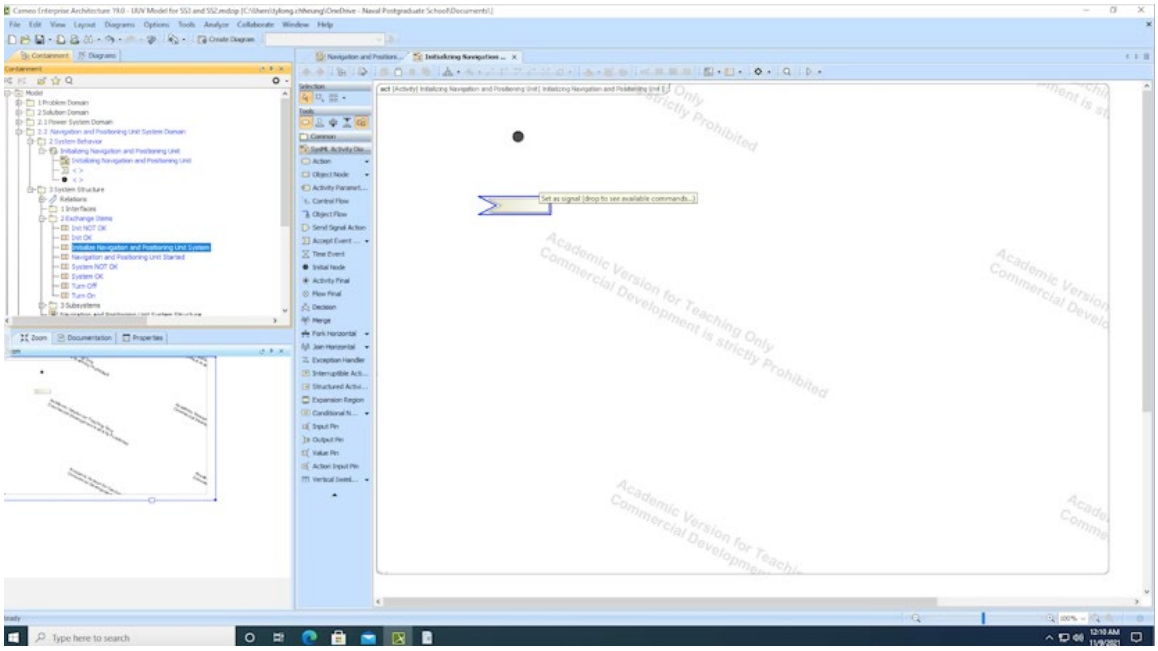
We will label the signal as Initialize Navigation and Positioning Unit System.



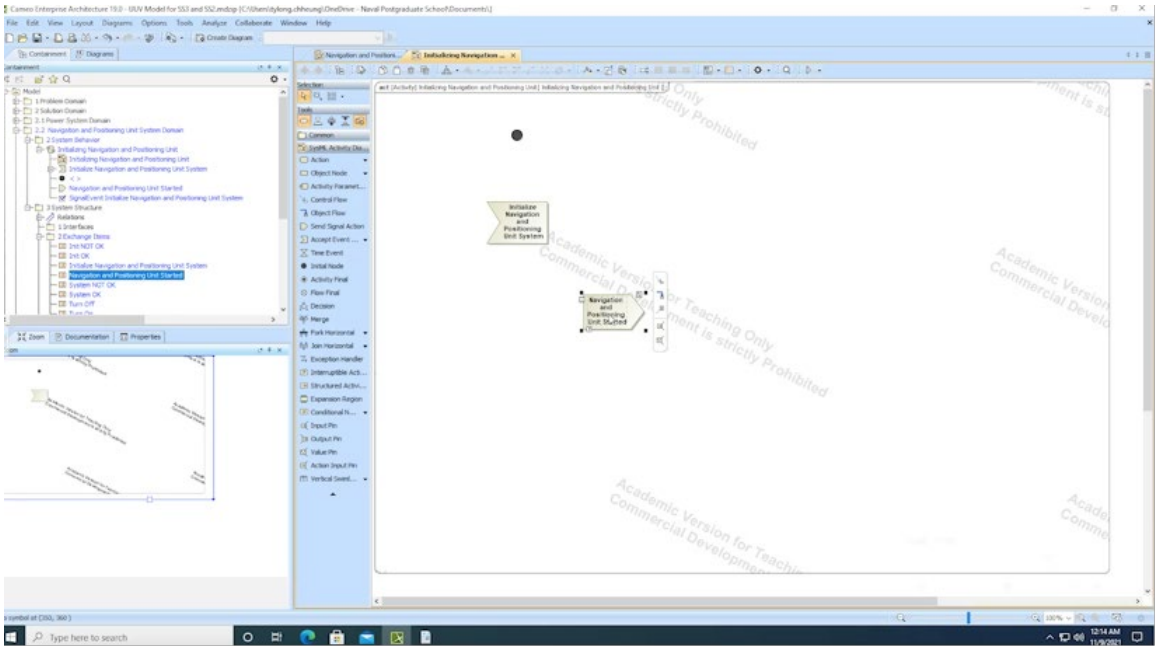
Now repeat the previous steps in order to create another signal which we will label as *Navigation and Positioning Unit Started*.



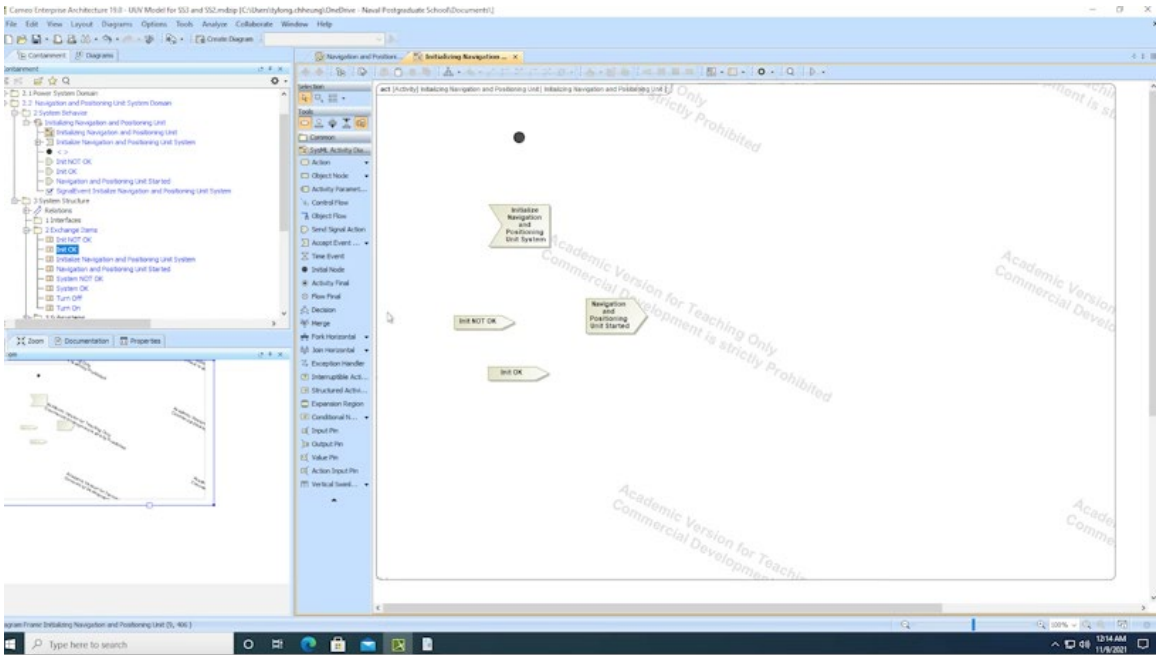
Now we will select the Initialize Navigation and Positioning Unit System signal and drag it to the Accept Event Action.



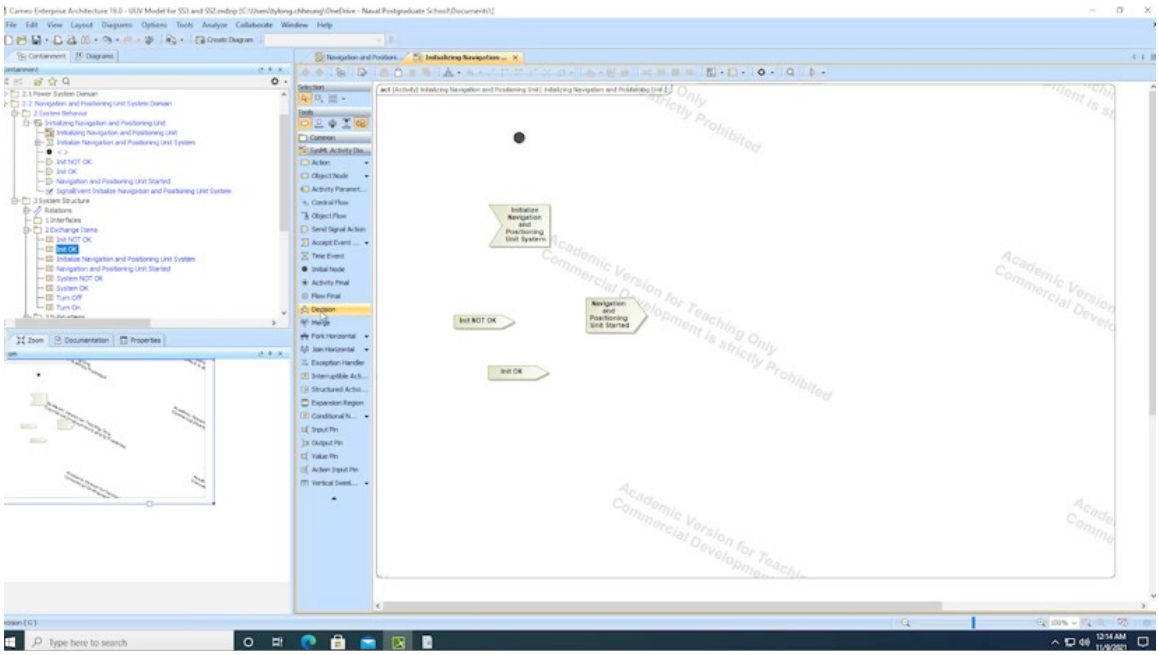
Once you have the *Accept Event Action* created, we can go back to the containment tree on the left-hand side and select the *Navigation and Positioning Unit Started* signal and drag it to an empty space in the diagram plane. This will create a send signal action.



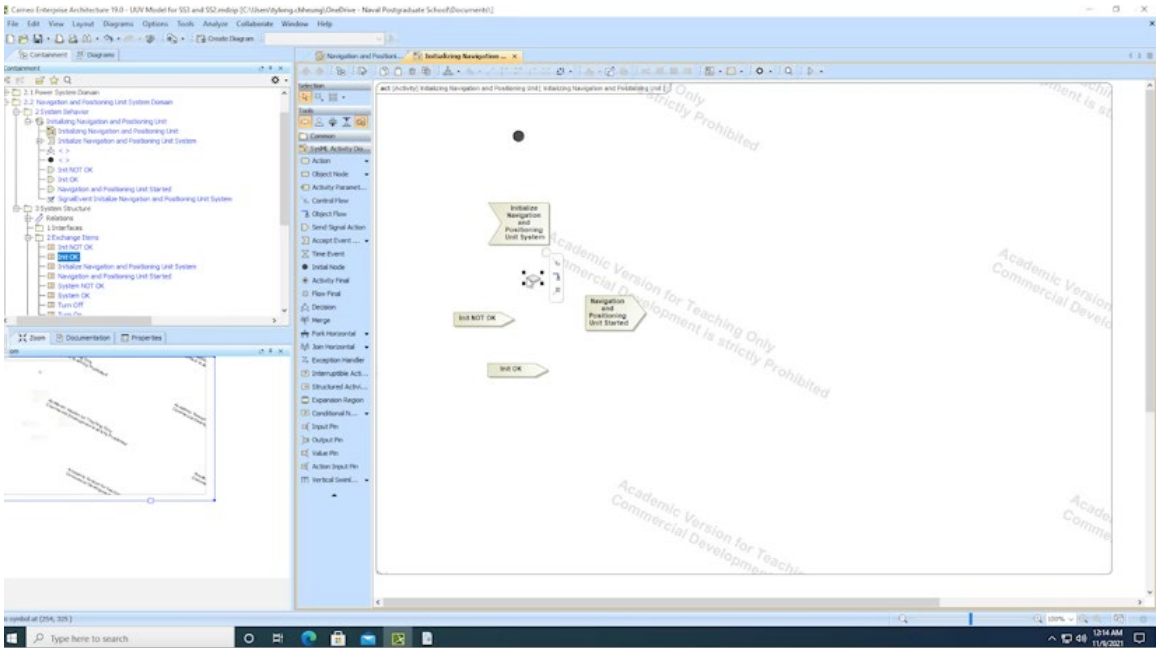
We will go back to the containment tree on the left-hand side and select the *Int NOT OK* signal and drag it to an empty space in the diagram plane. And we will do the same for the *Int OK* signal.



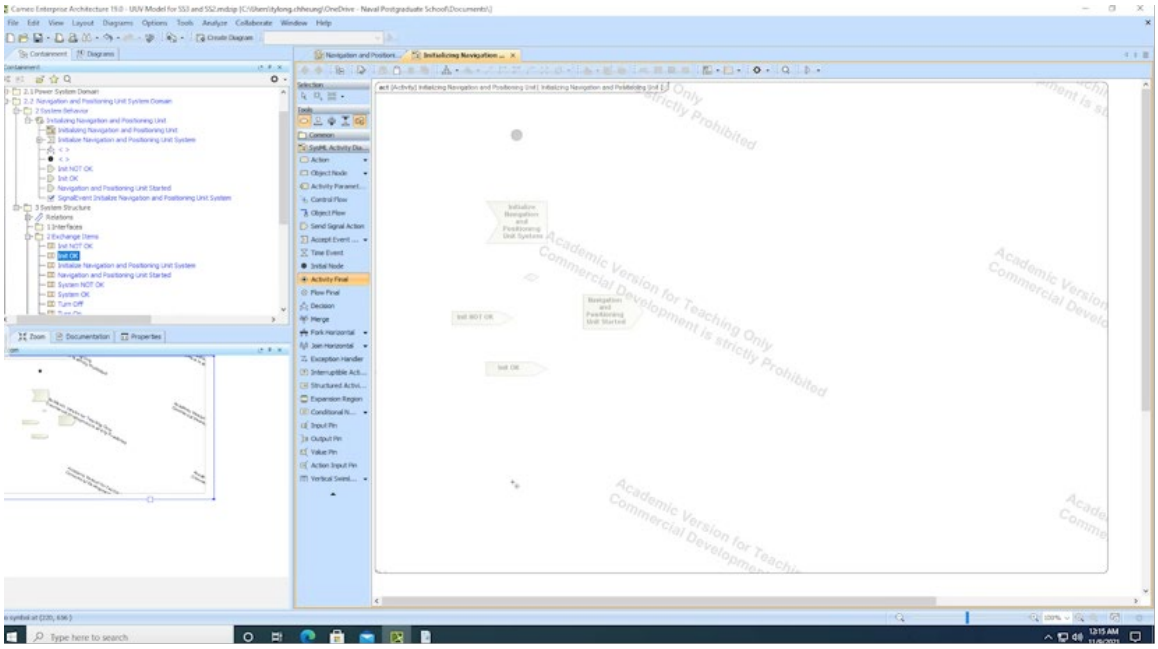
Now we will head over to the diagram pallet and select the *Decision* button.



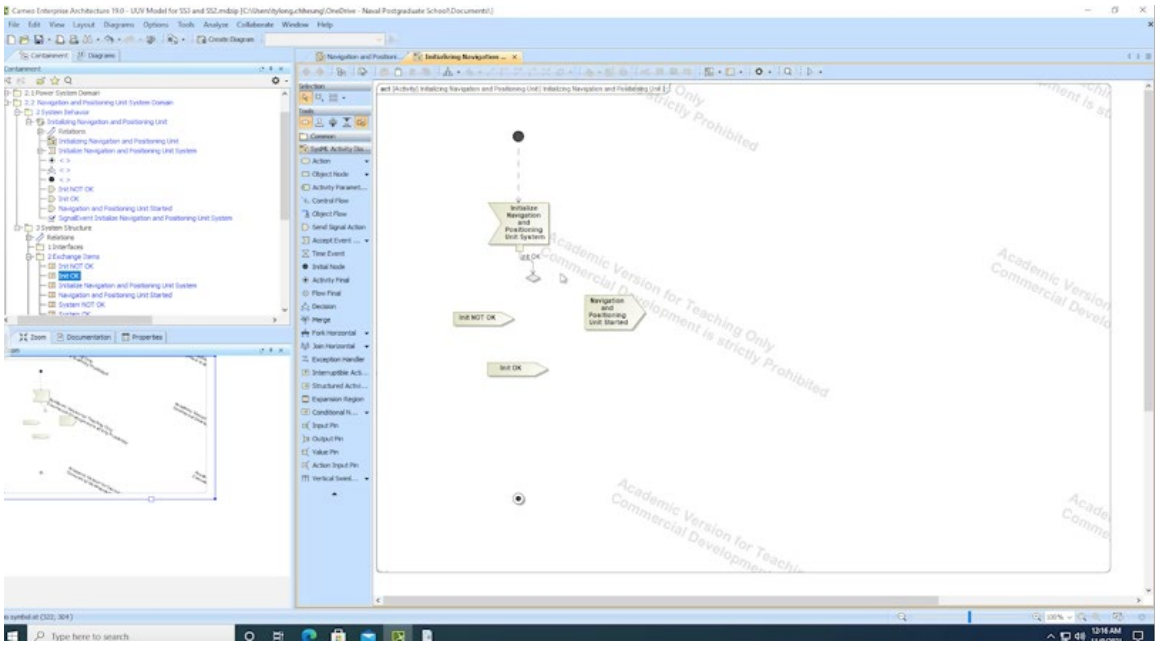
The we will drag it in the empty space below the *Accept Event Action* in the diagram plane.



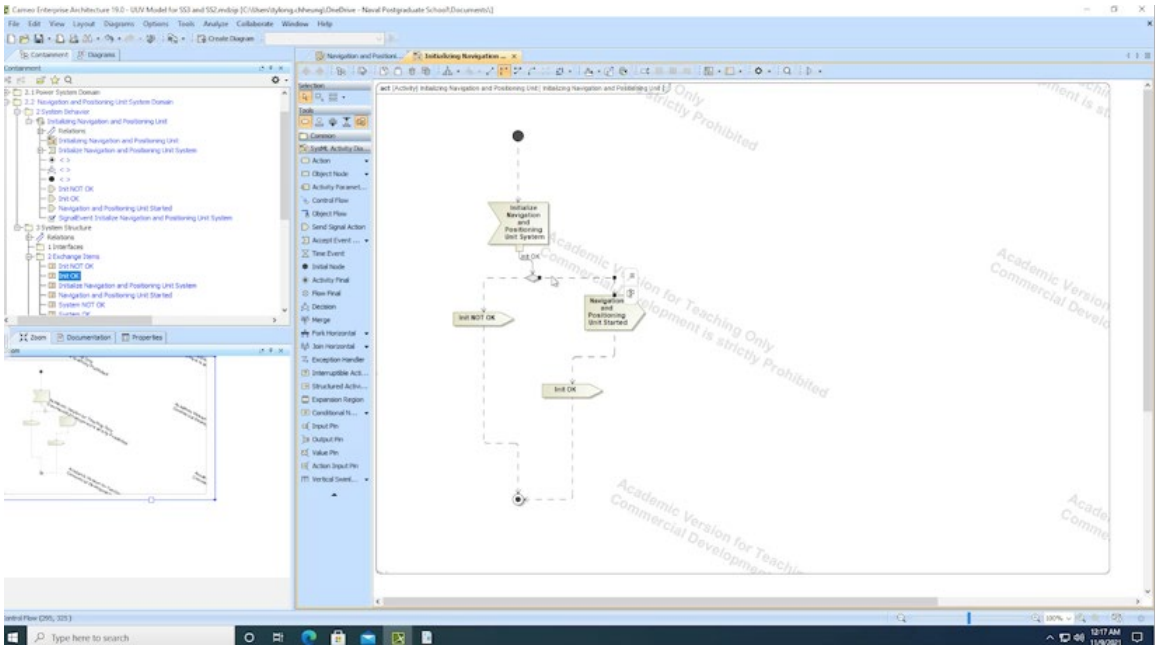
And the last thing will be that we head back to the diagram pallet and select the *Activity Final* button and drag it to an empty space in the diagram plane.



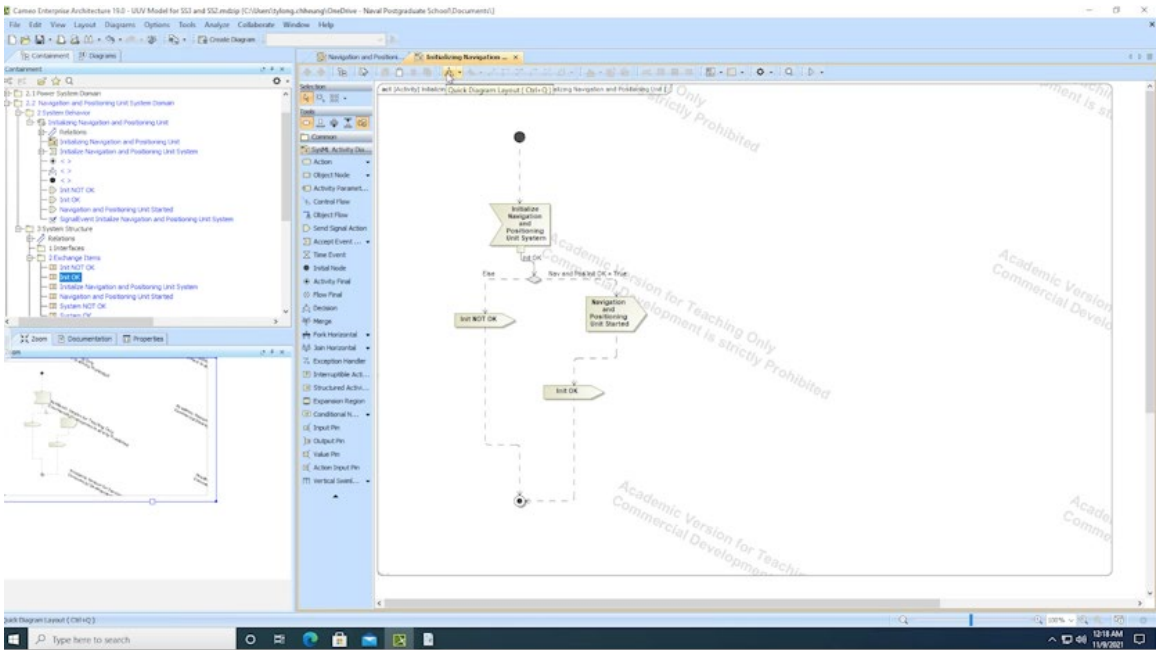
With everything in the diagram plane, it is just a matter of connecting all the different actions. We will start with go and select the *Initial Node* and from the manipulator tool bar we will select the *Control Flow* option and drag it to the *Initialize Navigation and Positioning Unit System Accept Event Action*. Now we will select the *Accept Event Action* and from the manipulator tool bar we will select the *Object Flow* option and drag it into the *Decision Node*. This will create a result pin and we will label as *Int Ok*.

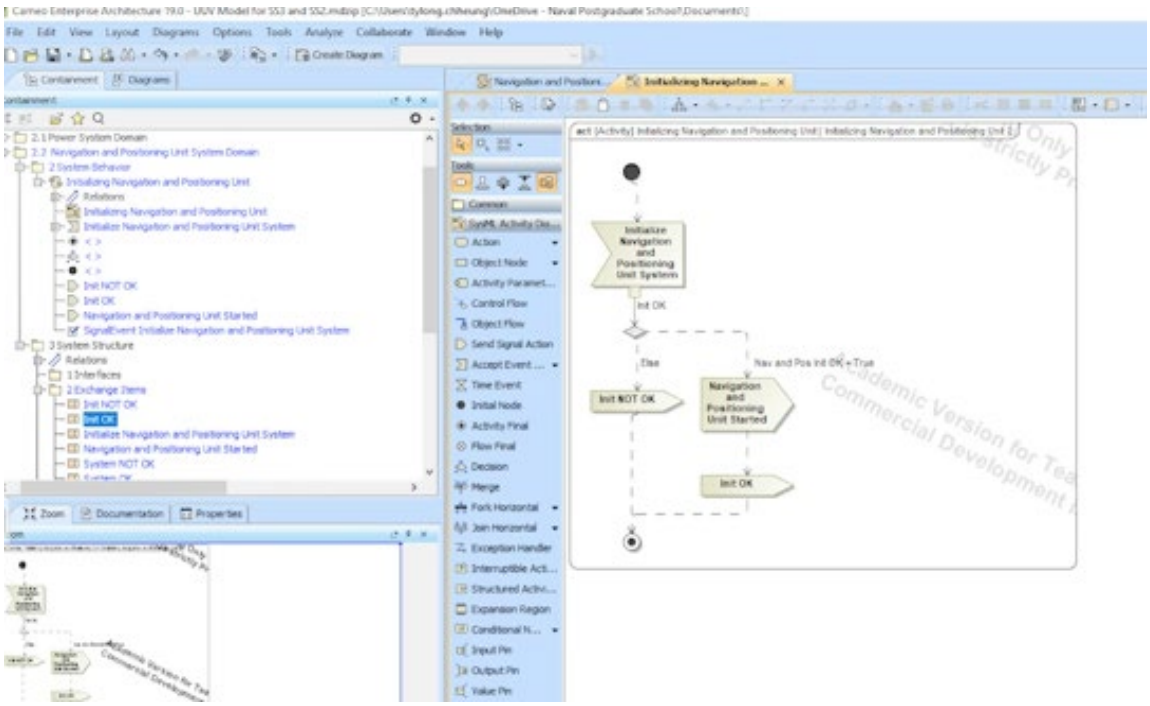


We will select the *Decision Node* and from the manipulator tool bar we will select the *Control Flow* option and drag it to the *Navigation and Positioning Unit Started* send signal. Then we will select the *Navigation and Positioning Unit Started* send signal and from the manipulator tool bar we will select the *Control Flow* option and drag it to the *Int Ok* signal. And then we select the *Int Ok* signal and from the manipulator tool bar we will select the *Control Flow* option and drag it to the *Activity Final Node*. Now we go back to the *Decision Node* to work on the other path, so we start by selecting the *Decision Node*. And from the manipulator tool bar we will select the *Control Flow* option and drag it to the *Int NOT Ok* signal. Then we select the *Int NOT Ok* signal and from the manipulator tool bar we will select the *Control Flow* option and drag it to the *Activity Final Node*.



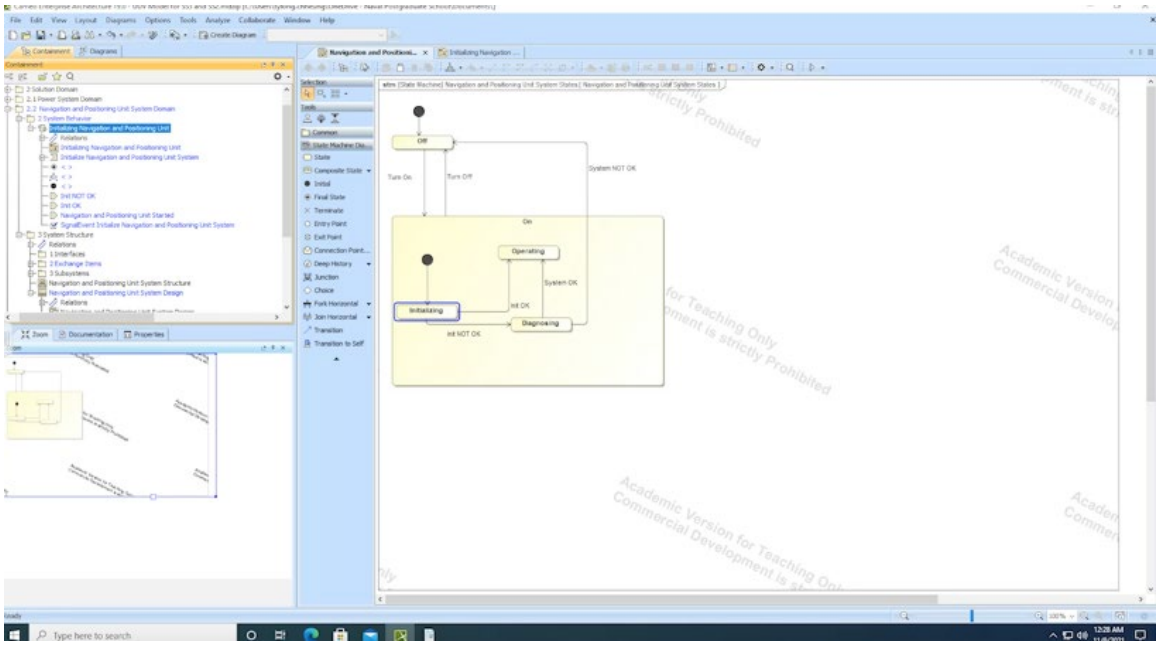
Now we have to label the outcomes of the *Decision Node*. On the right-hand side of the *Decision Node*, we will label it as *Nav and Pos Int Ok = True*. And for the left side we will label as *Else*. Once we have that done, we will select the *Quick Diagram Layout* button.



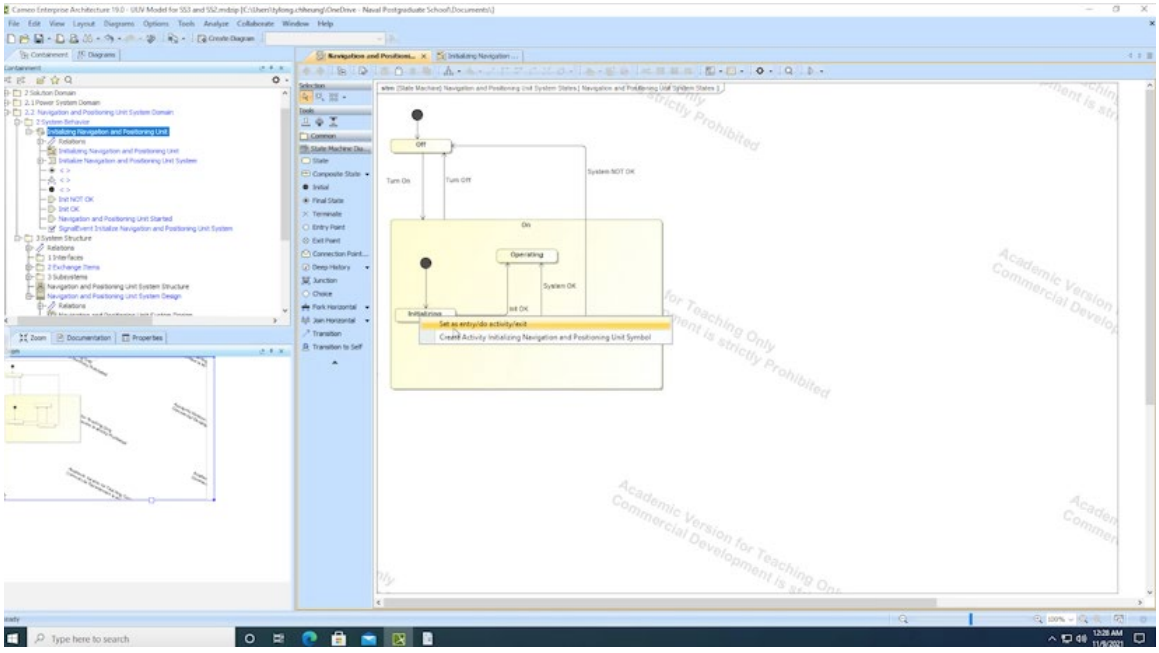


This to organize the activity diagram. And the final activity diagram should look similar to the photo below.

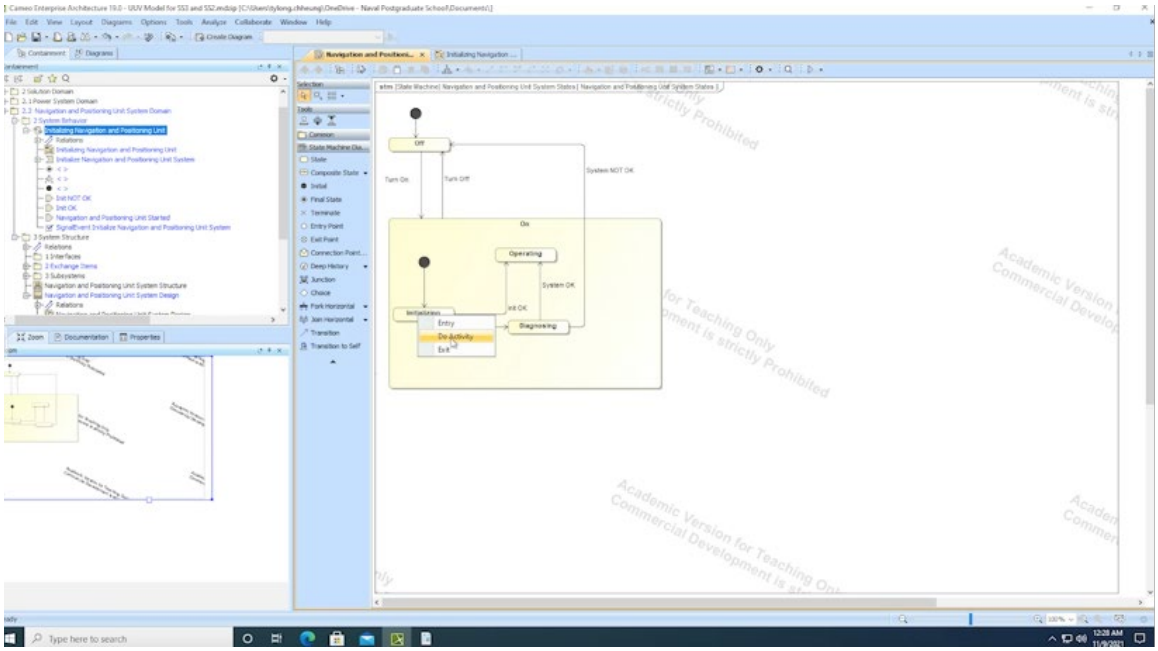
Now this is how you connect all the activities inside the *Initializing Navigation and Positioning Unit* activity. Now we will be returning to our state machine diagram. We can drag the newly created *Initializing Navigation and Positioning Unit* activity to the *Initializing* state.



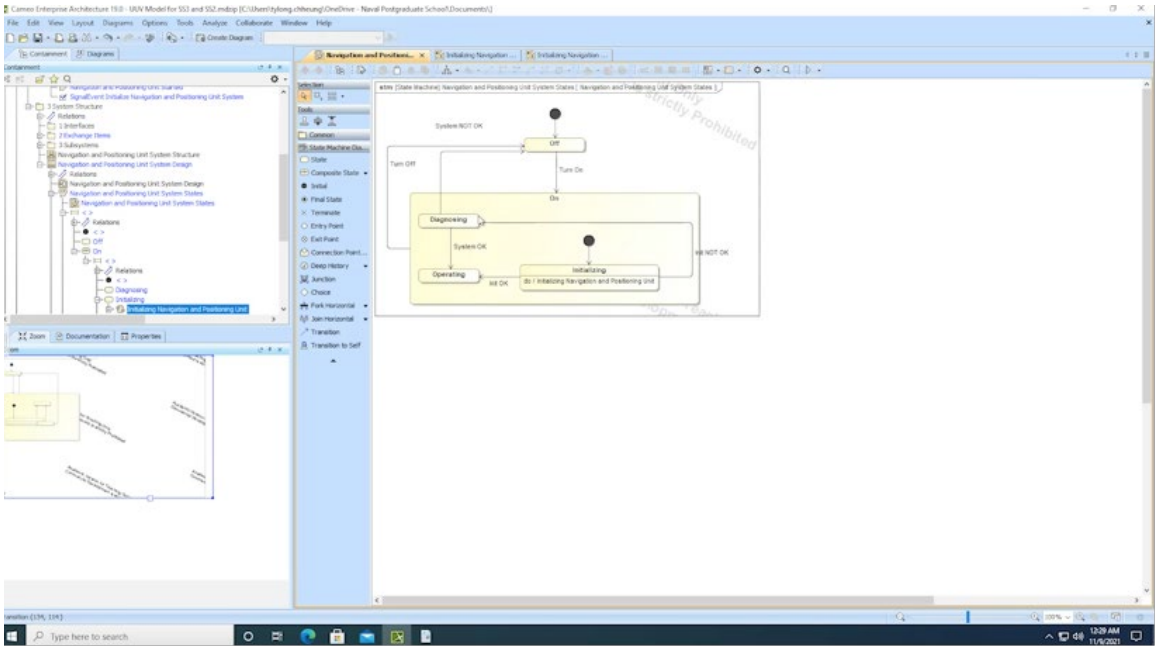
A popup drop-down menu will appear, and we will select the *Set as entry/do activity/exit* option.



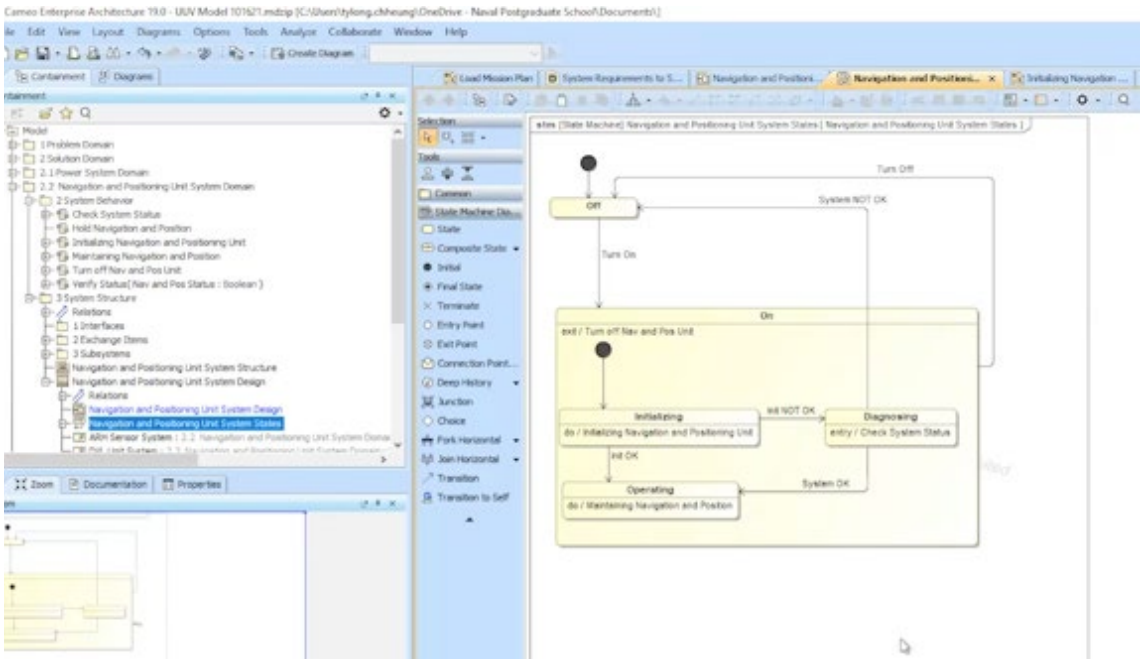
Another popup drop-down menu will appear, and we will select the *do activity* option.



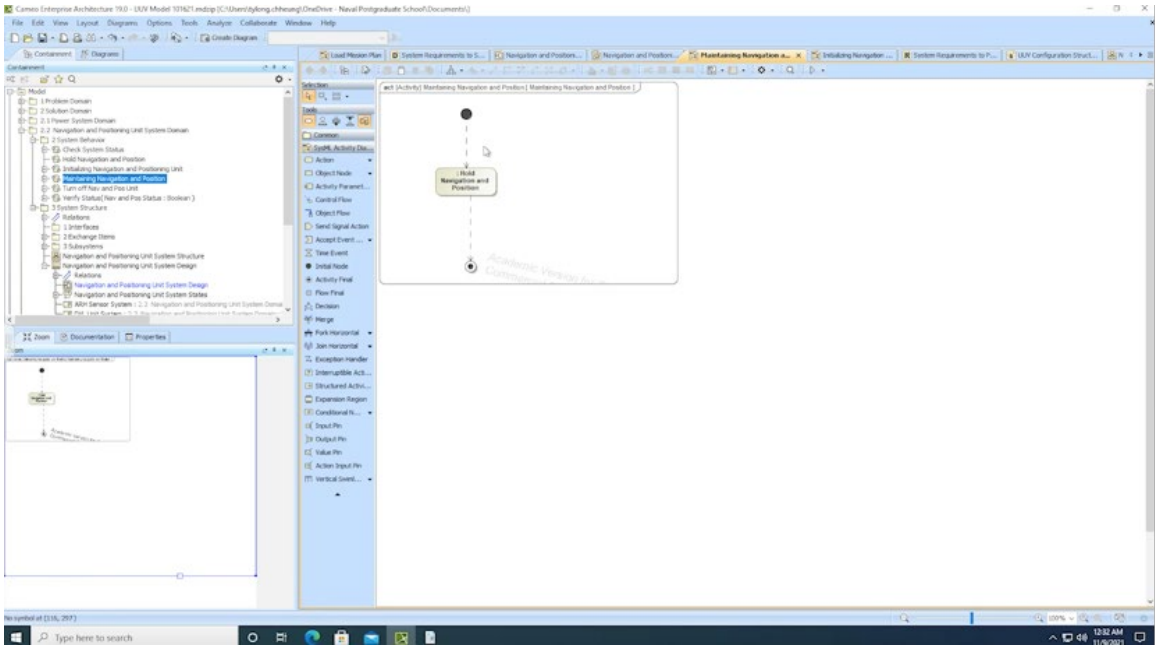
This will allocate the initializing action inside the *Initializing* state under do activity. Now we will select the *Quick Diagram Layout* button to organize the state machine diagram.



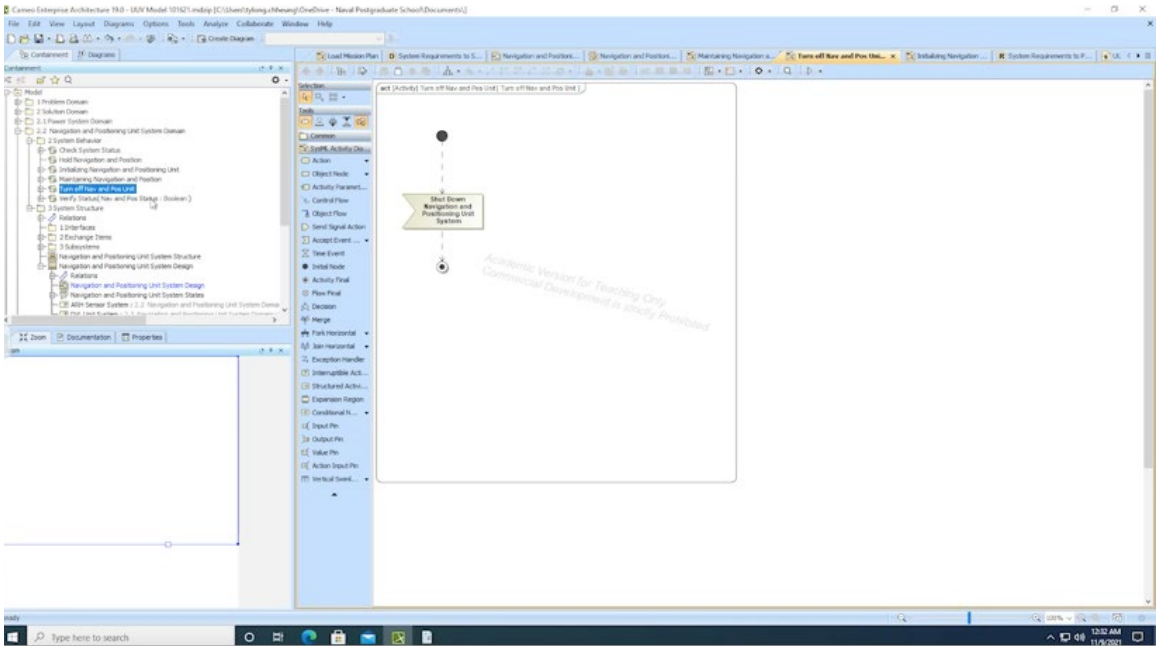
Now repeating the steps and process for the other states the final state machine diagram should look similar to the photo below.

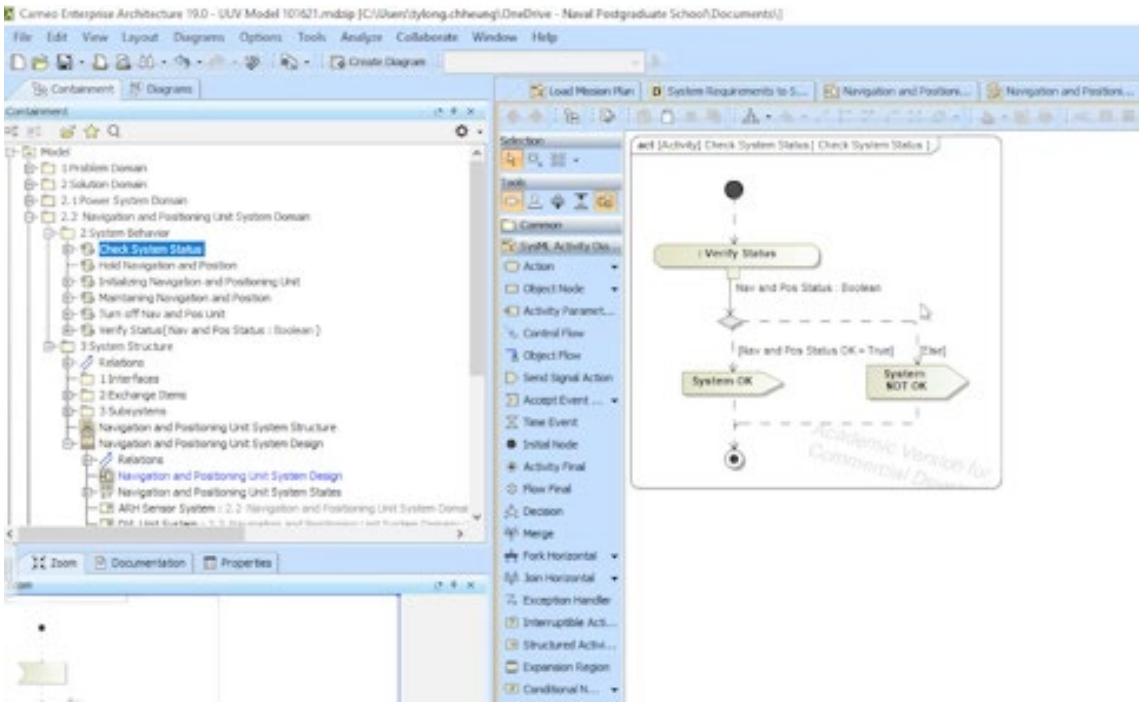


Here is the activity diagram for the *Maintaining Navigation and Positioning* activity.



Here is the activity diagram for the *Turn off Nav and Pos Unit* activity.





Here is the activity diagram for the *Check System Status* activity.

This concludes the tutorial for subsystem behavior. Thank you.

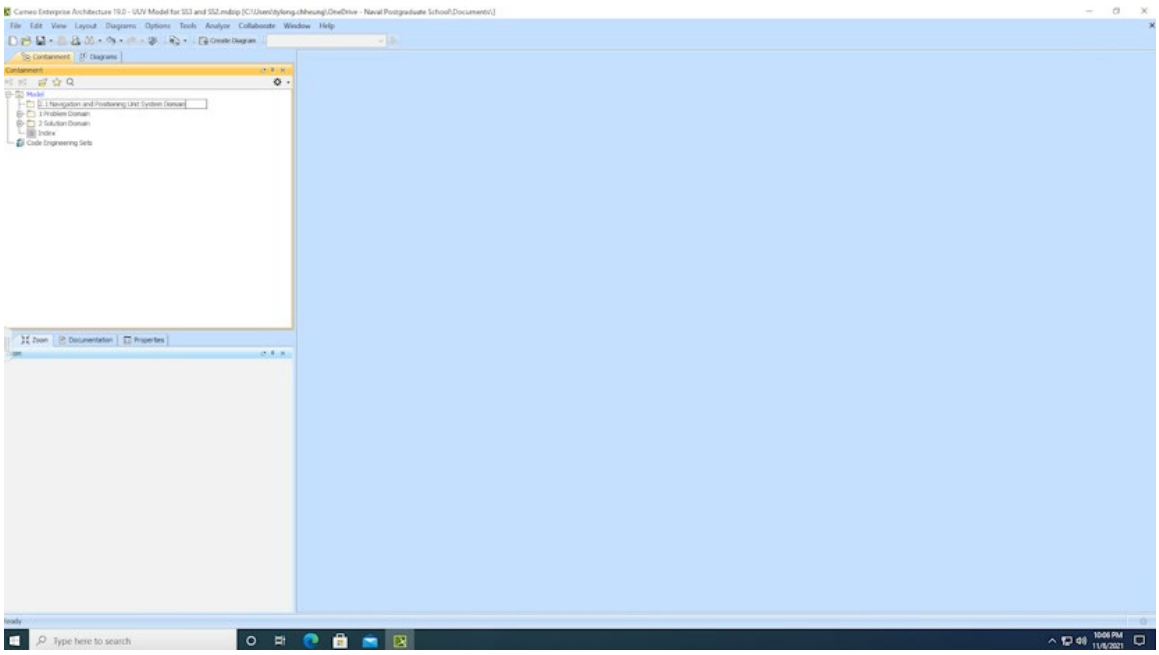
SS3: Subsystem Structure Tutorial

Background:

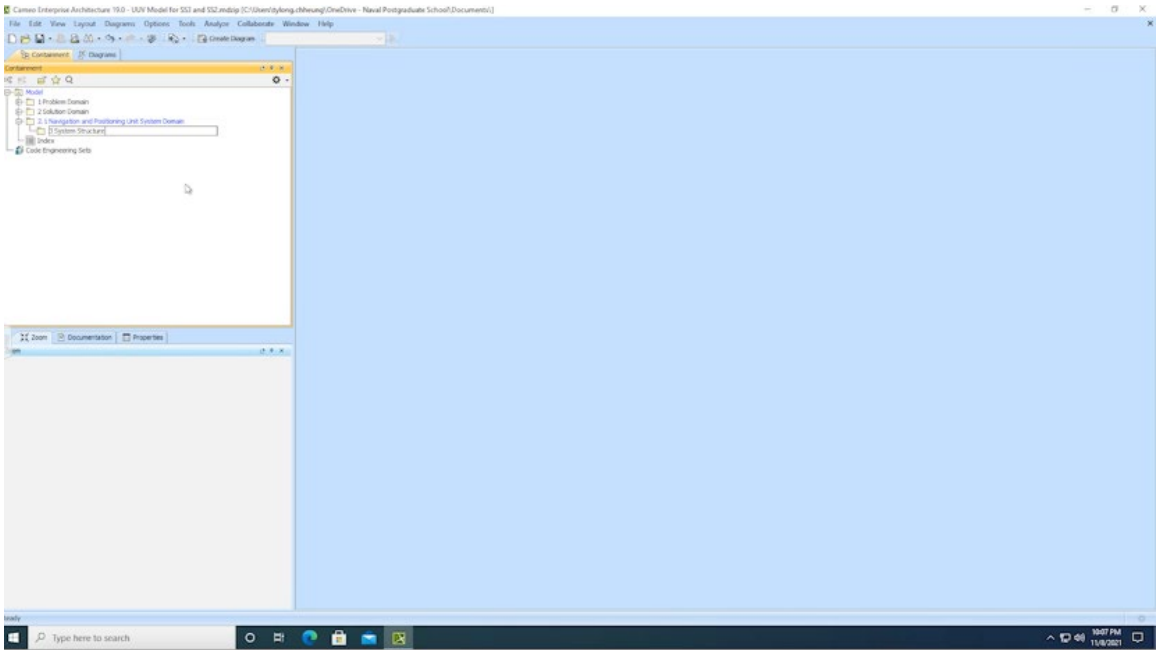
The solution architecture for the subsystems can be designed once the HLSA is complete. The subsystem structure will define how the different parts interoperate and interact with the rest of the system of interest. The subsystem structures use a SysML BDD to capture the subsystem and components using blocks and interface blocks. The specifications of how the subsystems interact and information exchange are generated in this diagram.

Step-By-Step Guide:

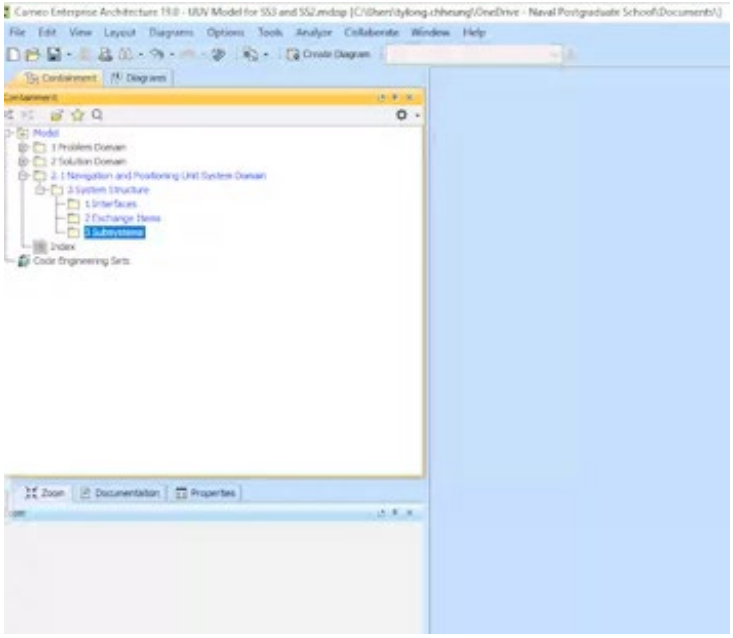
In this video we will cover how to create one of the subsystem structure architectures for the UUV. We will cover the *Navigation and Positioning Unit System*. So, let's start off with organizing the model for subsystem structure. In the containment tree on the left-hand side, we will right click on the *Model* package and from the drop-down menu we will select *Create Element*. The *Create Element* popup menu will appear, and we will select the *Package* option. We will label the new package as *2.1 Navigation and Positioning Unit System Domain*.

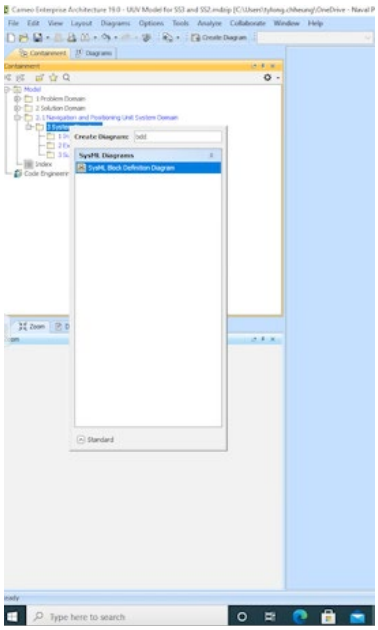


We will then right click the newly create *2.1 Navigation and Positioning Unit System Domain* package and from the drop-down menu we will select *Create Element*. The *Create Element* popup menu will appear, and we will select the *Package* option. We will label the new package as *3 System Structure*.

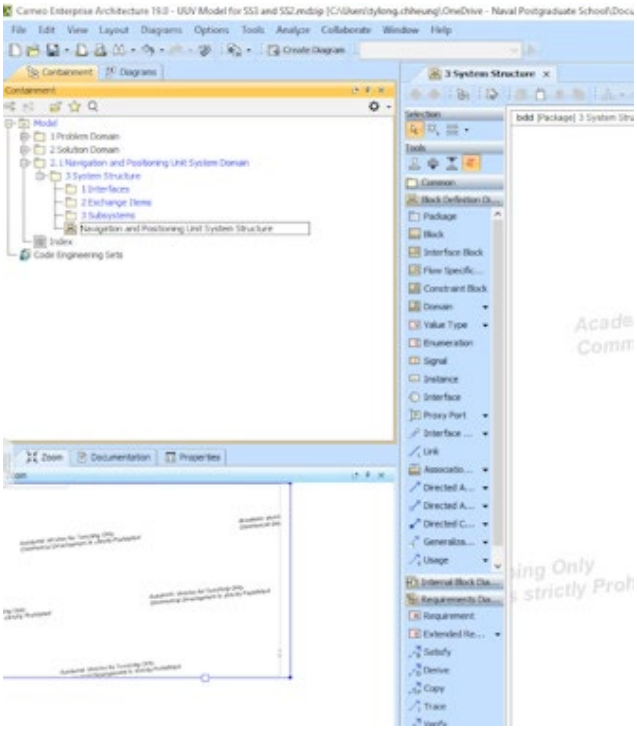


Next, we have to create another three sub packages for the *System Structure* package, which are the interfaces, the exchange items, and subsystem package. We will select the newly create *3 System Structure* package and from the drop-down menu we will select *Create Element*. The *Create Element* popup menu will appear, and we will select the *Package* option. We will label the new package as *1 Interfaces*. We will repeat the process for the other two packages, which will be *2 Exchange Items* and *3 Subsystems*. The subsystem package should look similar to the photo below.

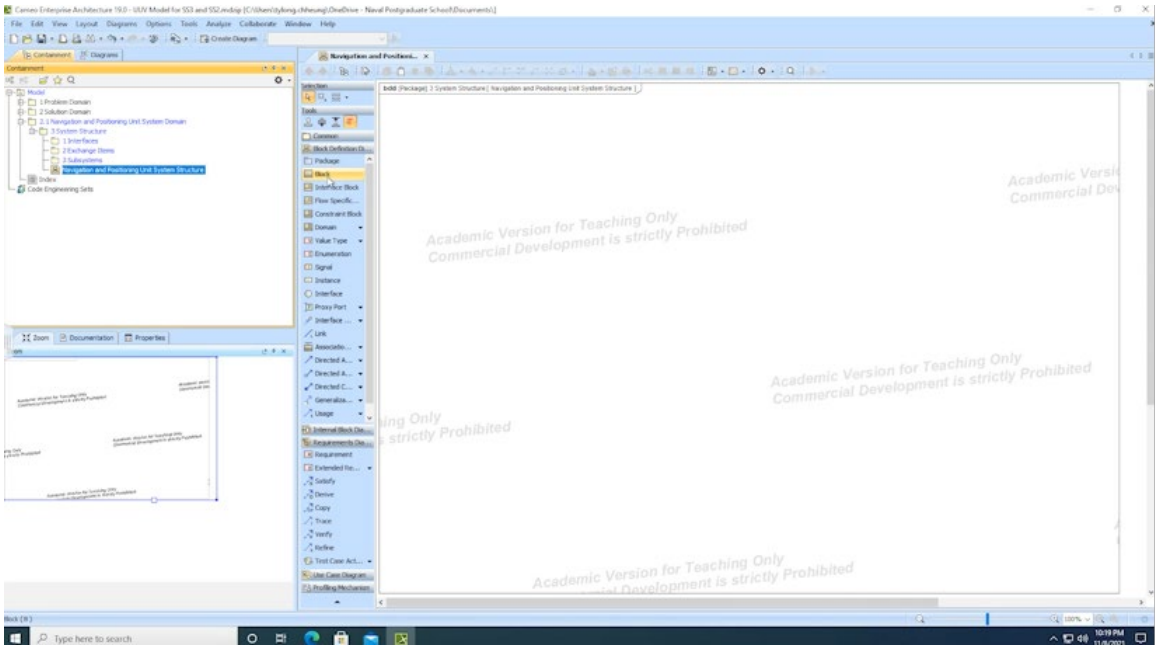




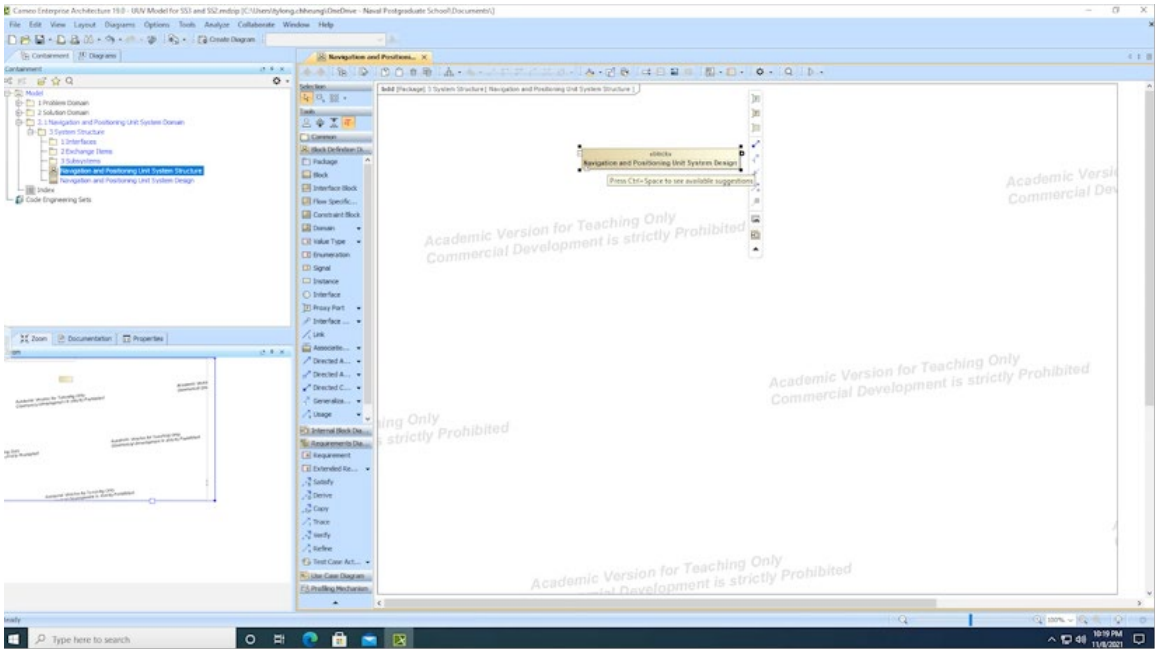
We can start to create a *BDD* to capture the structure of the *Navigation and Positioning Unit*. We are starting by right clicking *3 System Structure* package and from the drop-down menu we will select *Creat*e *Diagram*. The *Creat*e *Diagram* popup menu will appear, and we will type *bdd* in the search bar and select the *SysML Block Definition Diagram*. Then we will label the Block Definition Diagram as Navigation and Positioning Unit System Structure.



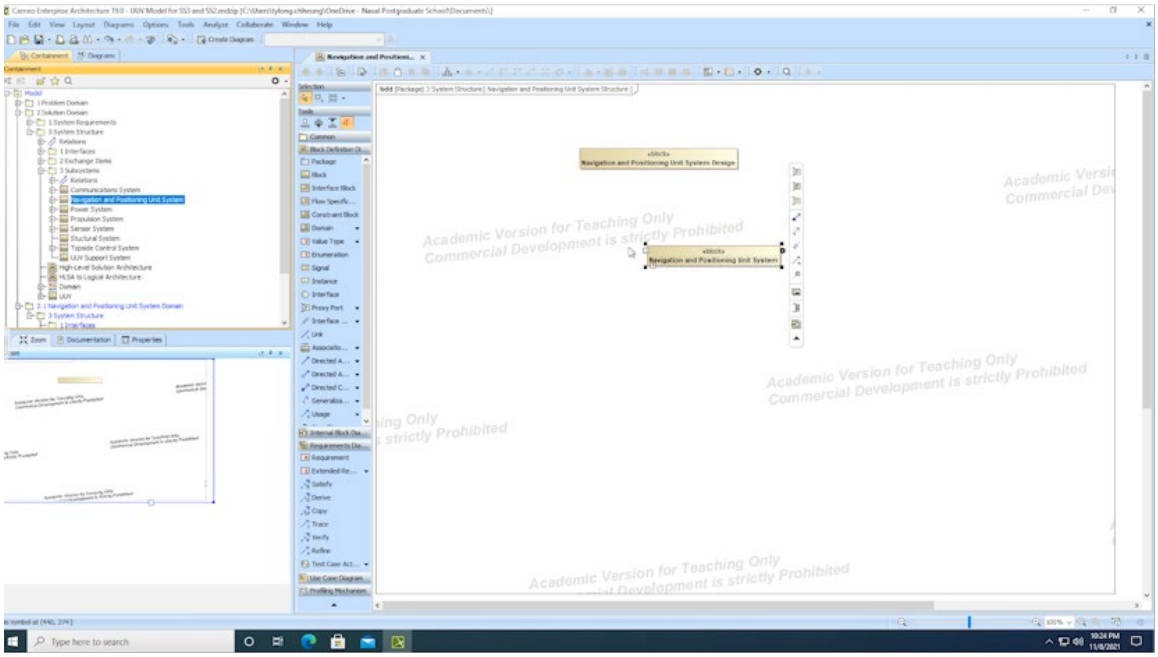
Then the first thing we will have to do is to select the *Block* button in the diagram pallet and drag it to an empty space in the diagram plane.



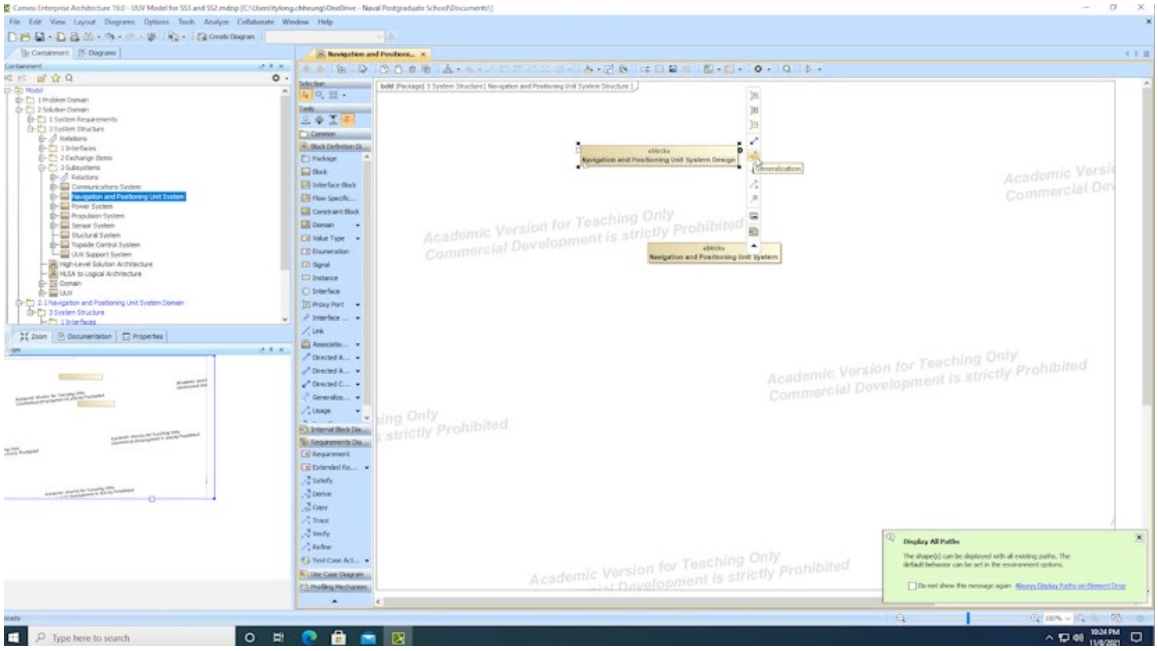
We will then label the block as Navigation and Positioning Unit System Design.



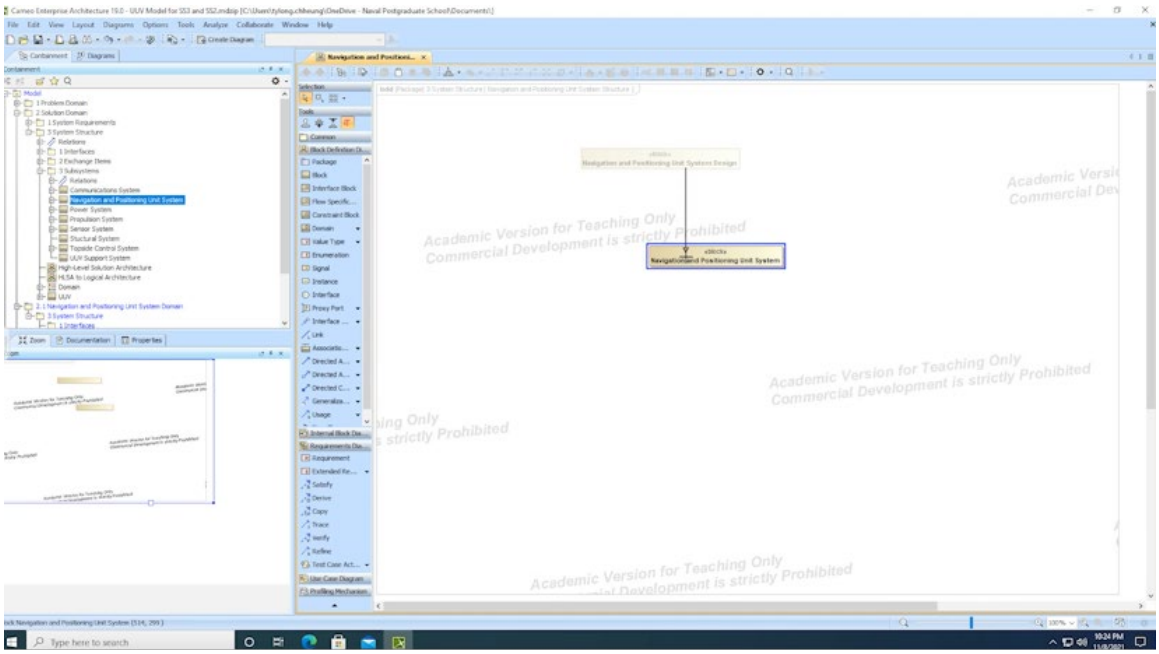
Once the *Navigation and Positioning Unit System Design* block is created then we can head over to the containment tree in the left-hand side and expand the *2 Solution Domain*. Then we will expand the subfolder *3 System Structure* where we will select the *Navigation and Positioning Unit System* package and drag it to an empty space in the diagram plane.



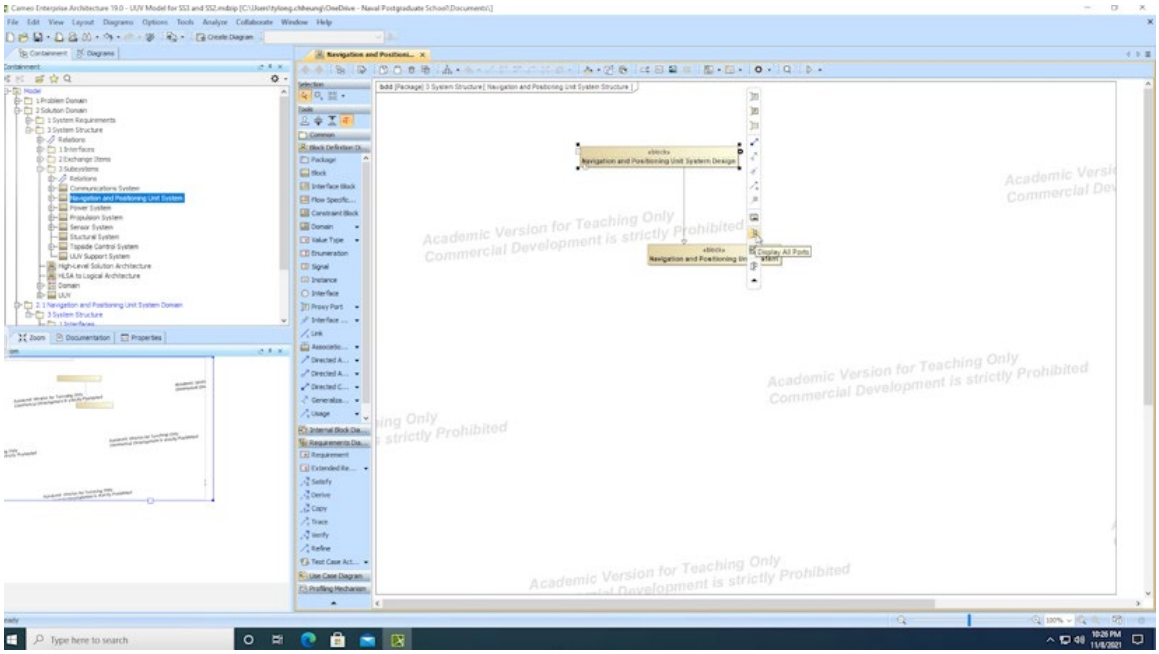
We will then click on the *Navigation and Positioning Unit System Design* block and select the *Generalization* button on the smart manipulator tool bar.



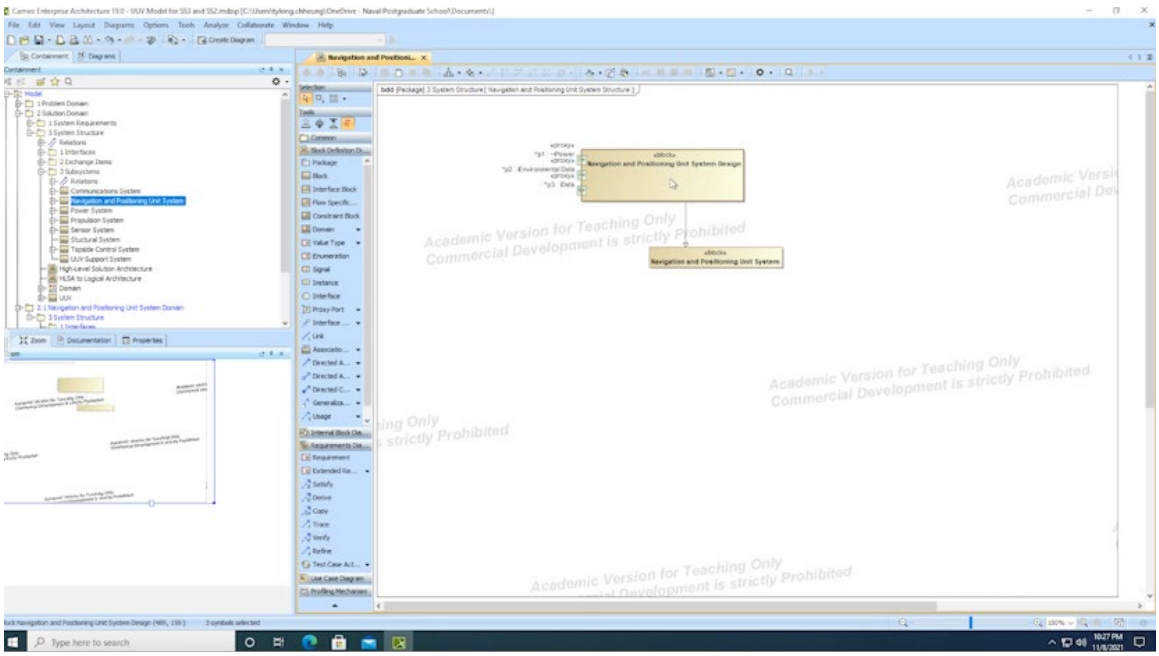
Then we will drag the arrow to the *Navigation and Positioning Unit System* block, which will highlight the block in blue.



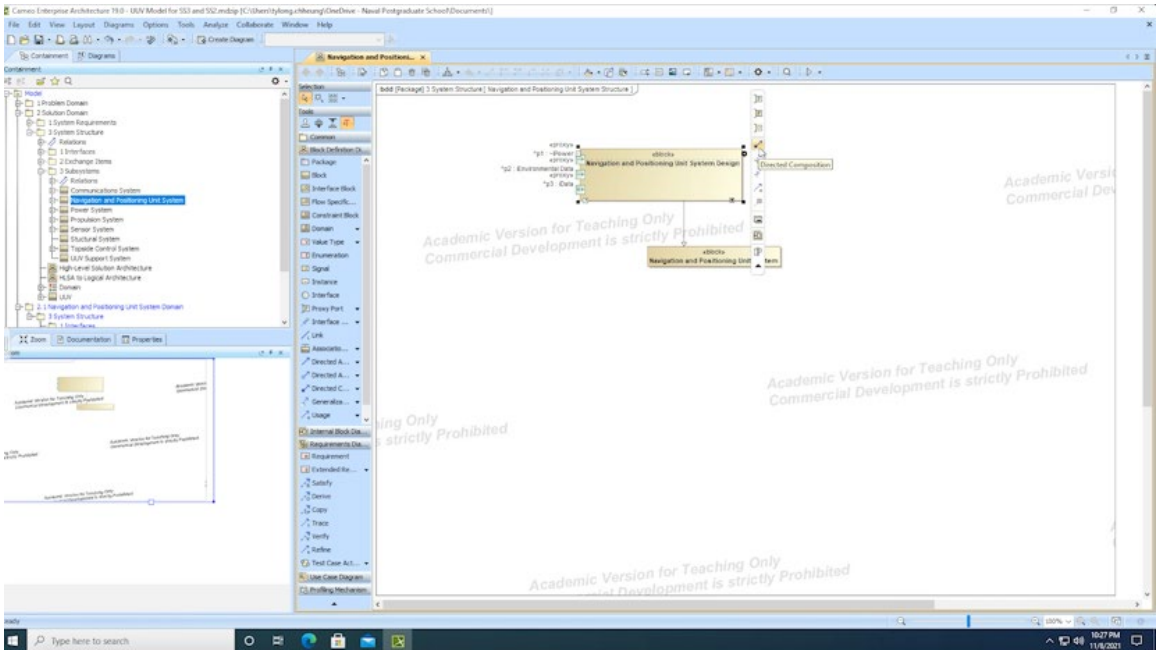
From the smart manipulator tool bar from the *Navigation and Positioning Unit System Design* block, we will select the *Display All Ports* button.



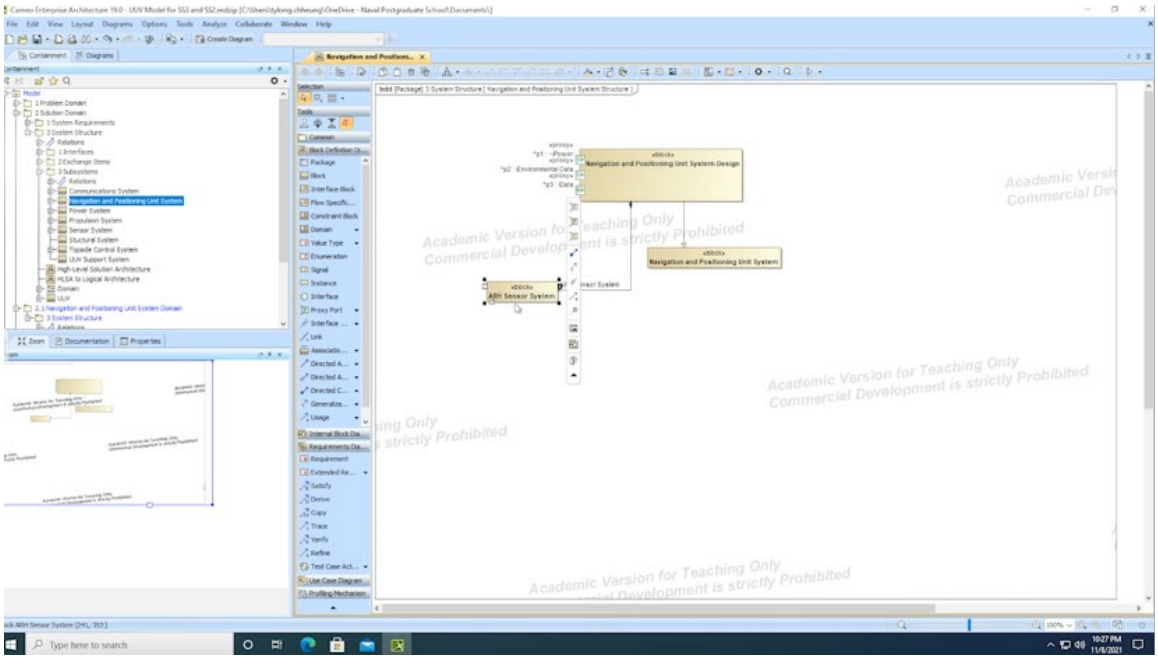
This will show all the ports that incoming from the Navigation and Positioning Unit System block to the Navigation and Positioning Unit System Design block.



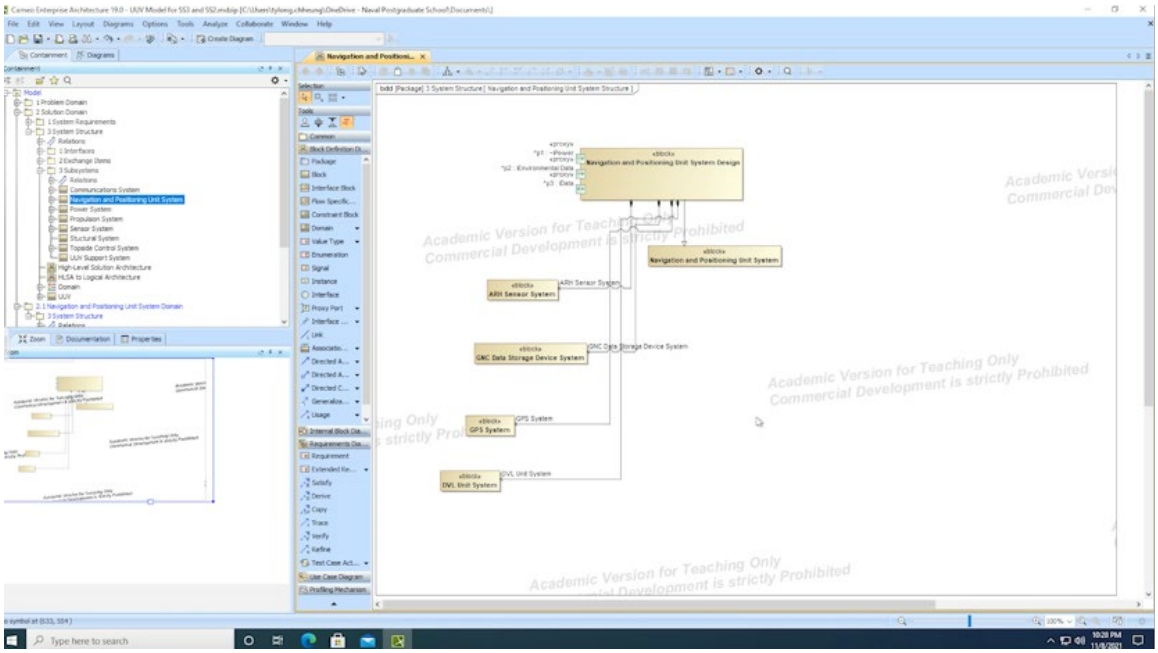
Once we have all of these, we can select *Navigation and Positioning Unit System Design* block again. From the smart manipulator we will select the *Directed Composition* button to start build its subsystem components.



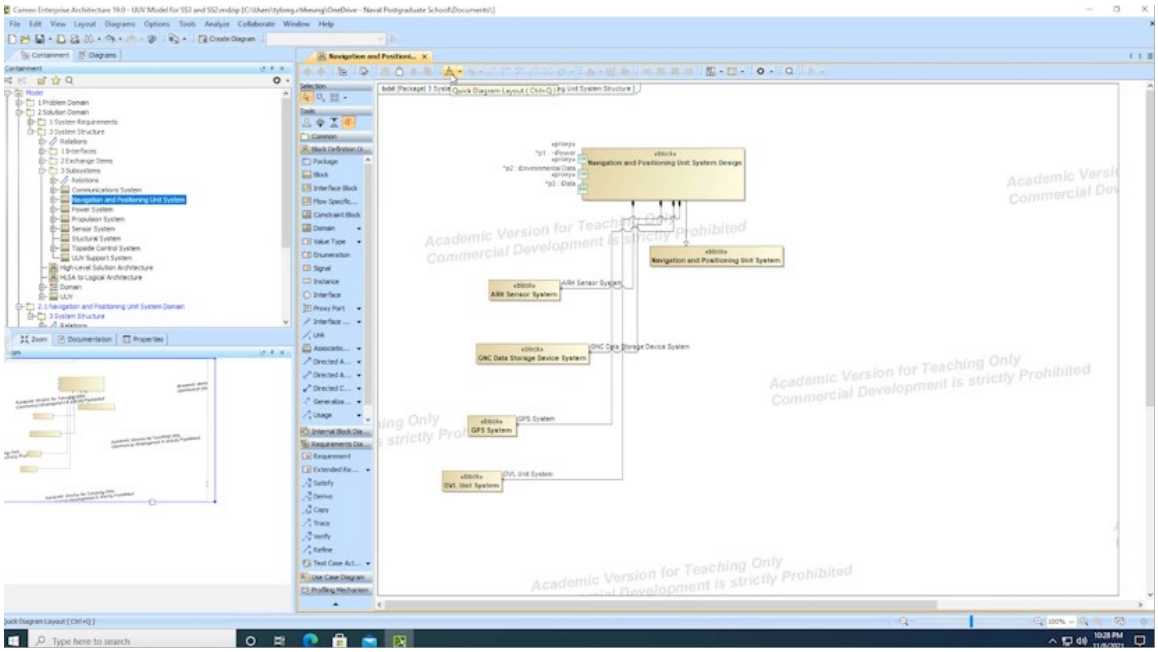
After selecting the *Directed Composition* button, we will click on any empty space in the diagram plane create a subsystem block. We will label the subsystem block as *ARH Sensor System*.



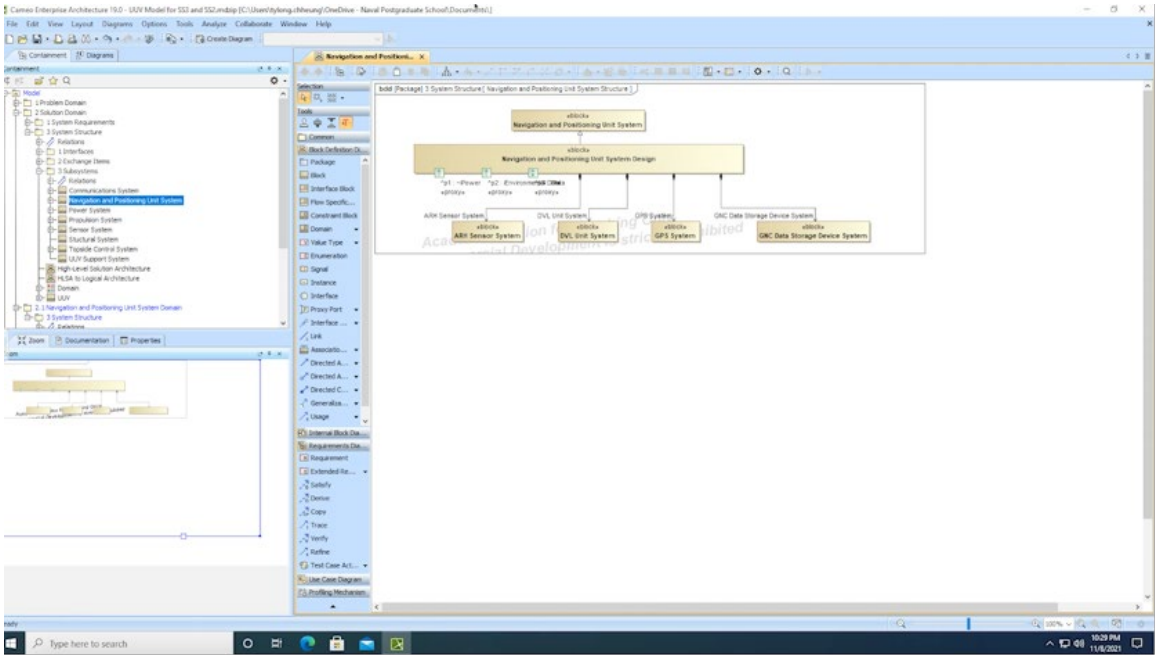
We will repeat the previous steps to create the remaining subsystem blocks, which are the *GNC Data Storage Device System*, *GPS System*, and *DVL Unit System*. The subsystem blocks should be similar to the photo below.



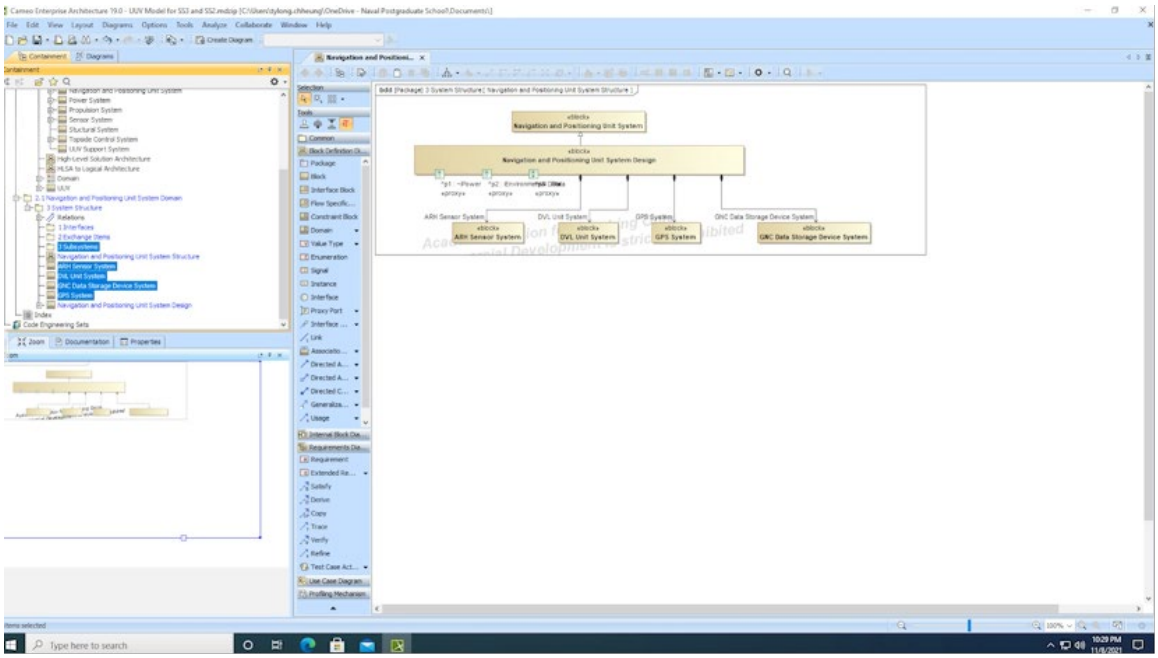
On the diagram tool bar, we will select the *Quick Diagram Layout* button to organize the *Block Definition Diagram*.



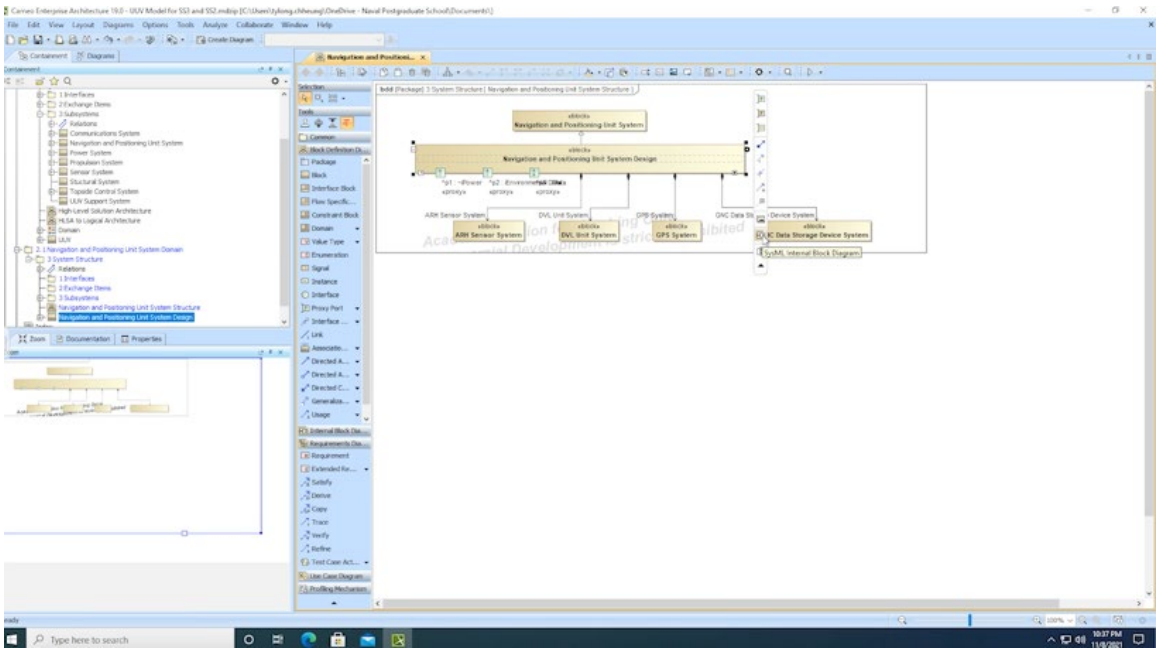
The new layout of the diagram should look similar to the photo below.



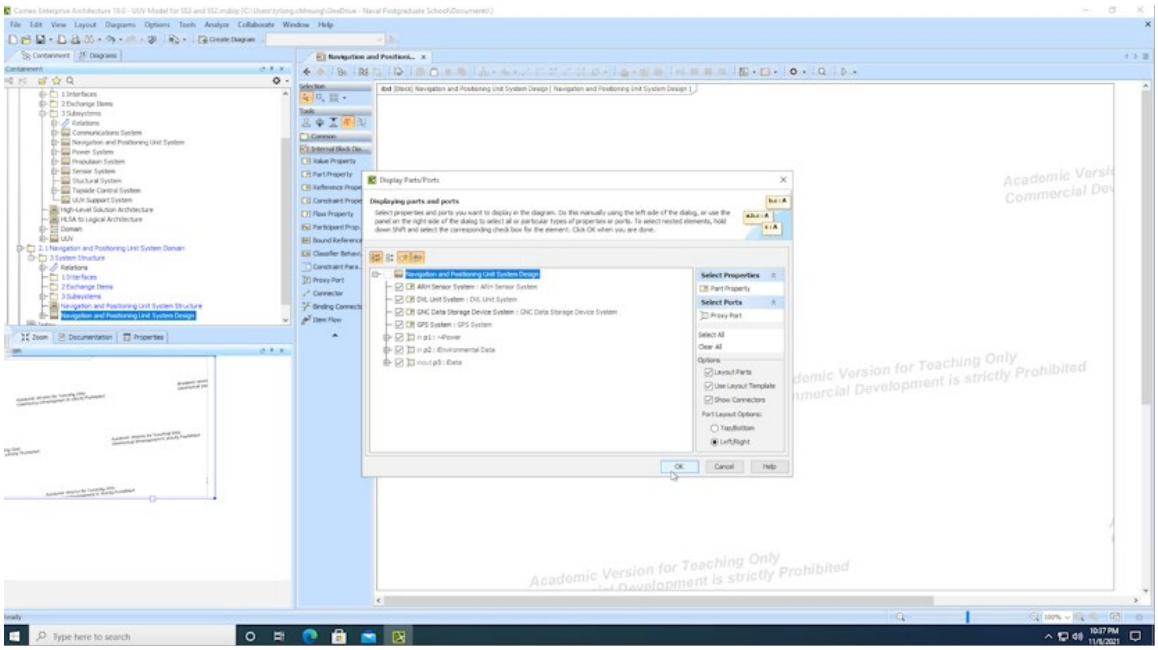
The last thing we have to do in order to finish the *Block Definition Diagram* is select the subsystems that were just created and drag them to *3 Subsystems*.



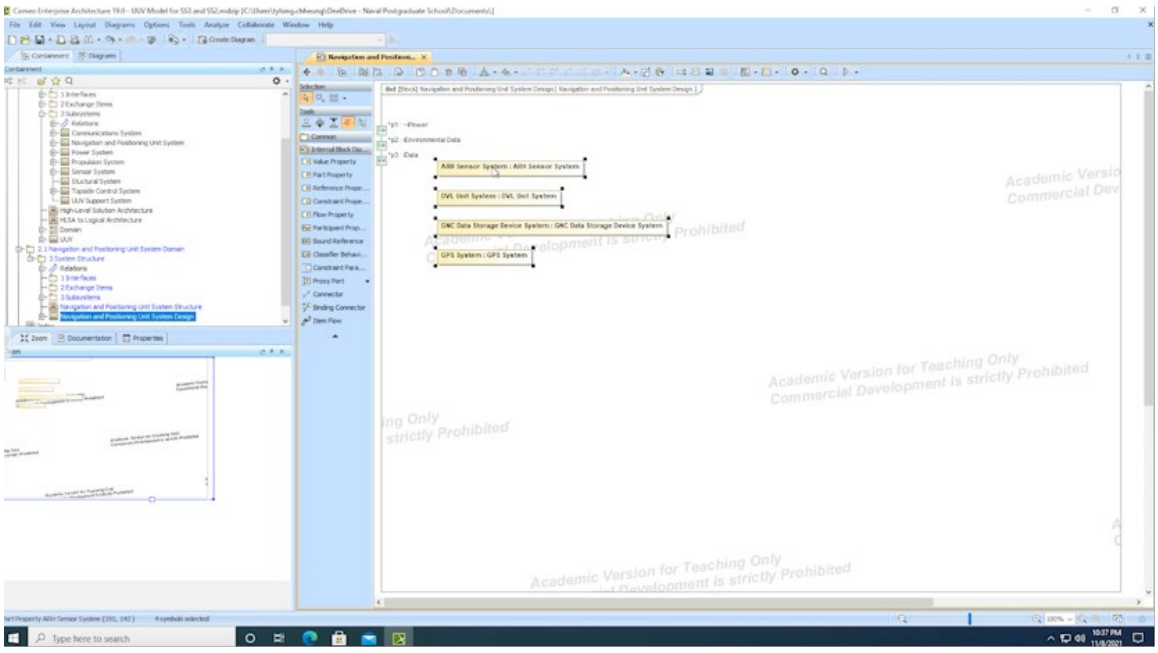
The last thing we have to do in order to complete the *3 System Structure* for the *2.1 Navigation and Positioning Unit System Domain* is create an *IBD* to specify the interactions between the subsystems that were newly created. So we will start by selecting *Navigation and Positioning Unit System Design* and from the smart manipular tool bar we will select the *SysML Internal Block Diagram* button.



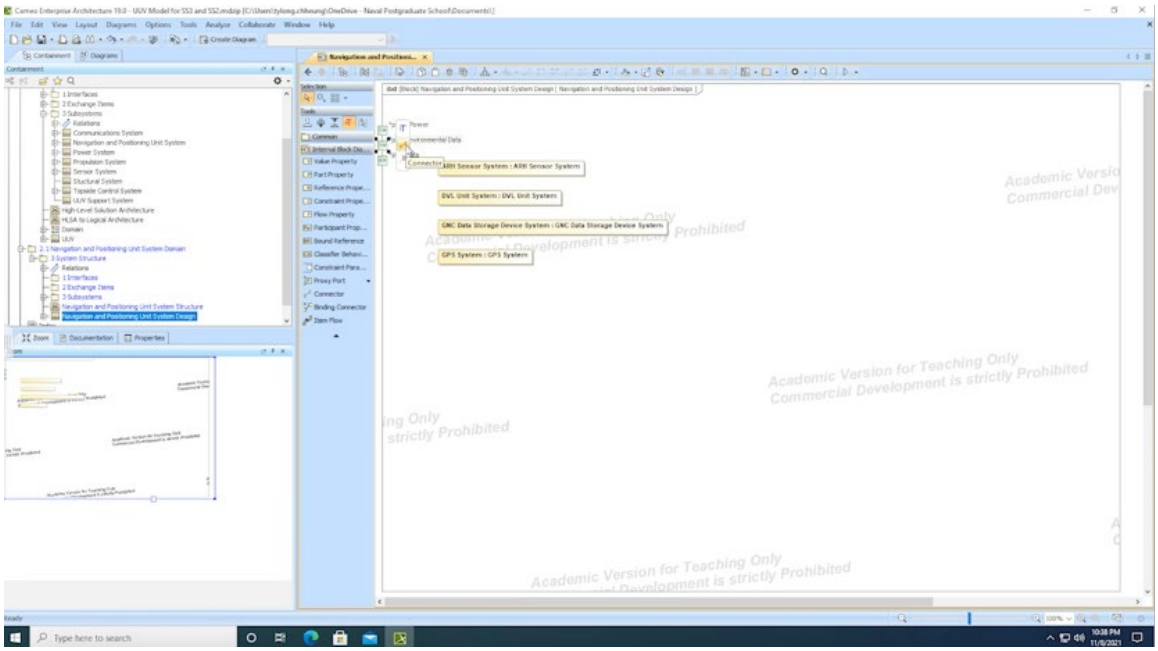
Then the *Display Parts/Ports* popup window will appear which will help generate the *Internal Block Diagram*. We have to make sure that all the parts properties are selected and then we click on the *Ok* button.



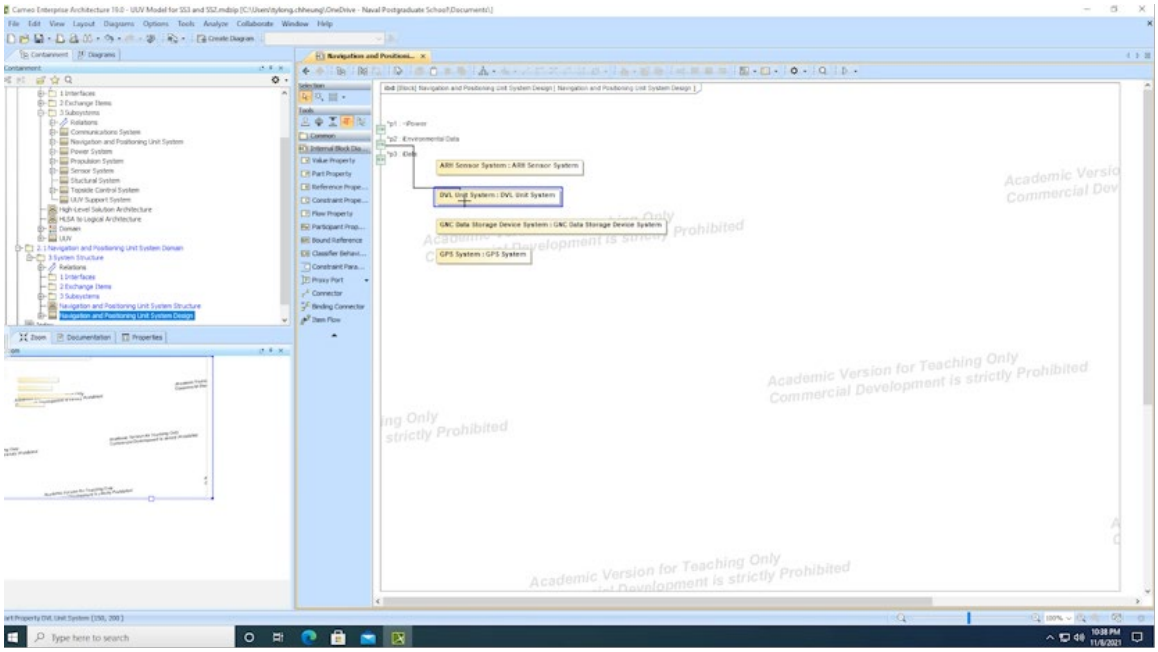
Then the *IBD* will be generated and now we will have to specify the interactions to fully complete the *IBD*.



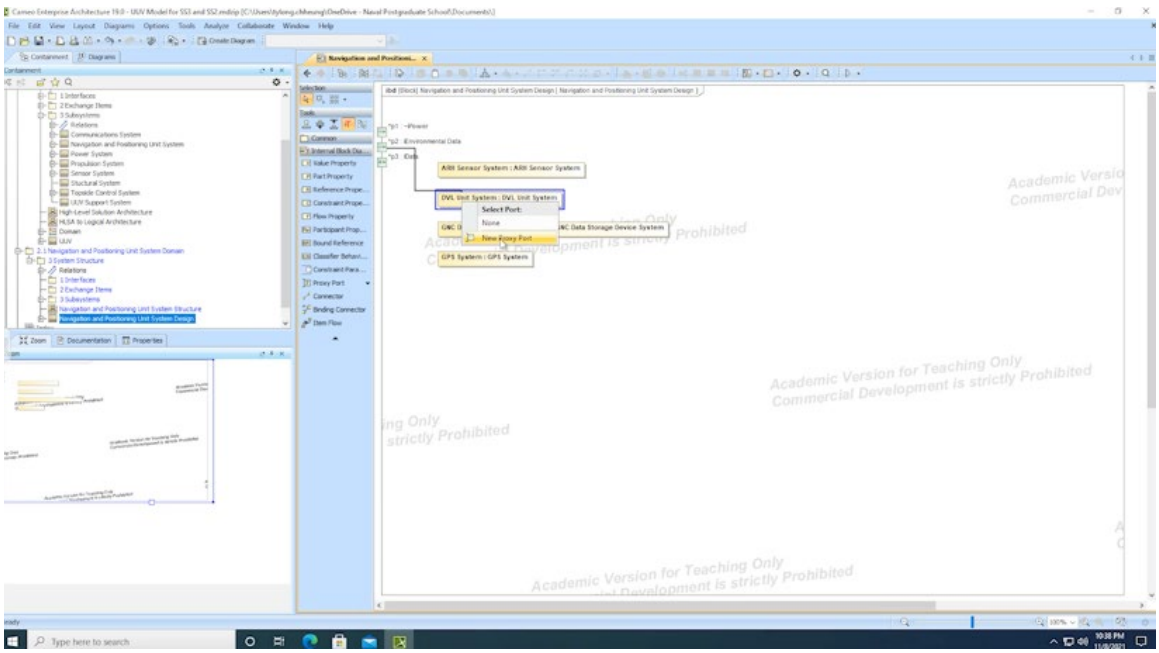
We will start by selecting the Proxy Port *P2 iEnvironmental Data* and from the smart manipulator tool bar we will select the *Connector* button.



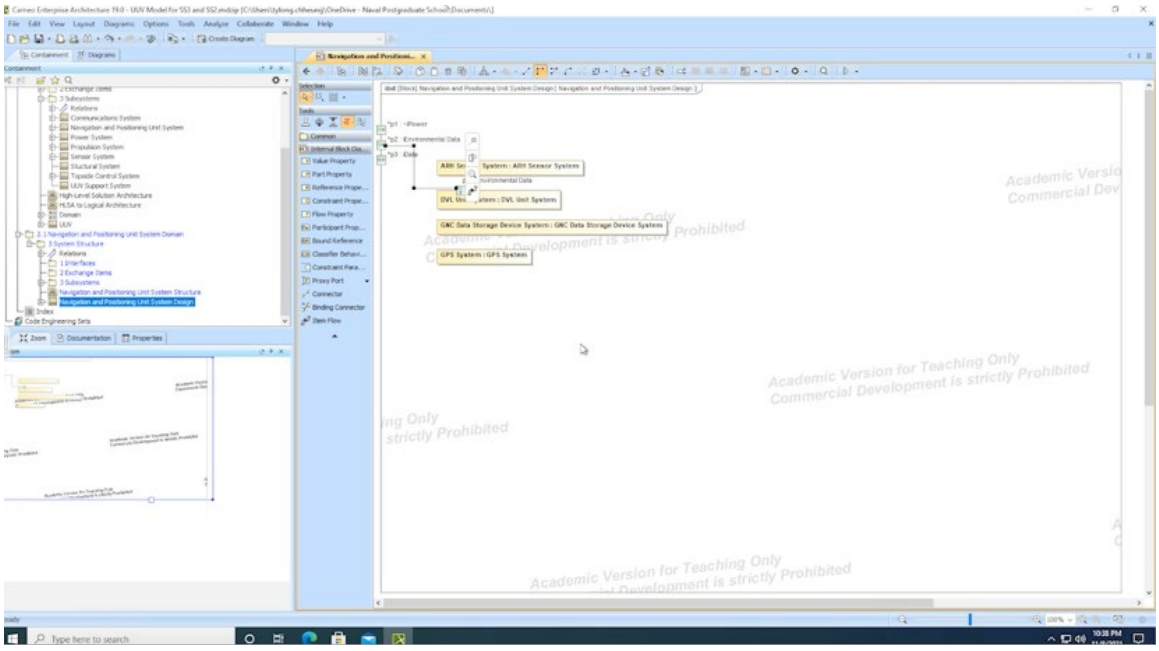
Then we will connect the *P2 iEnvironmental Data* Proxy Port with the *DVL Unit System* part property.



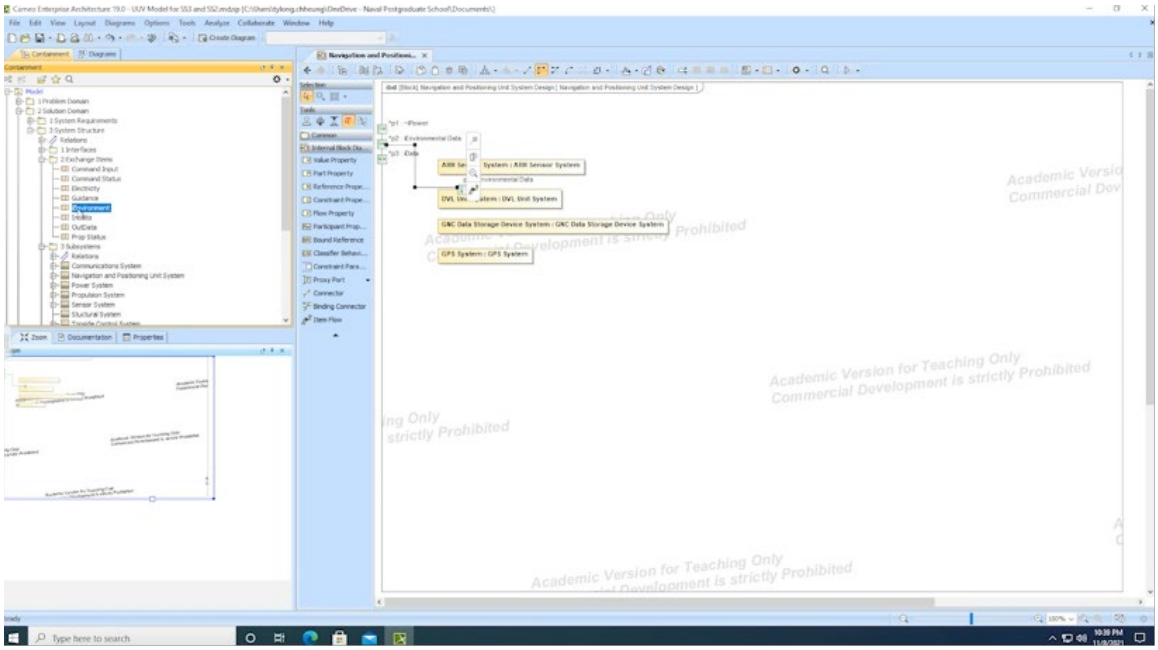
And we will select *New Proxy Port* option from the popup menu.



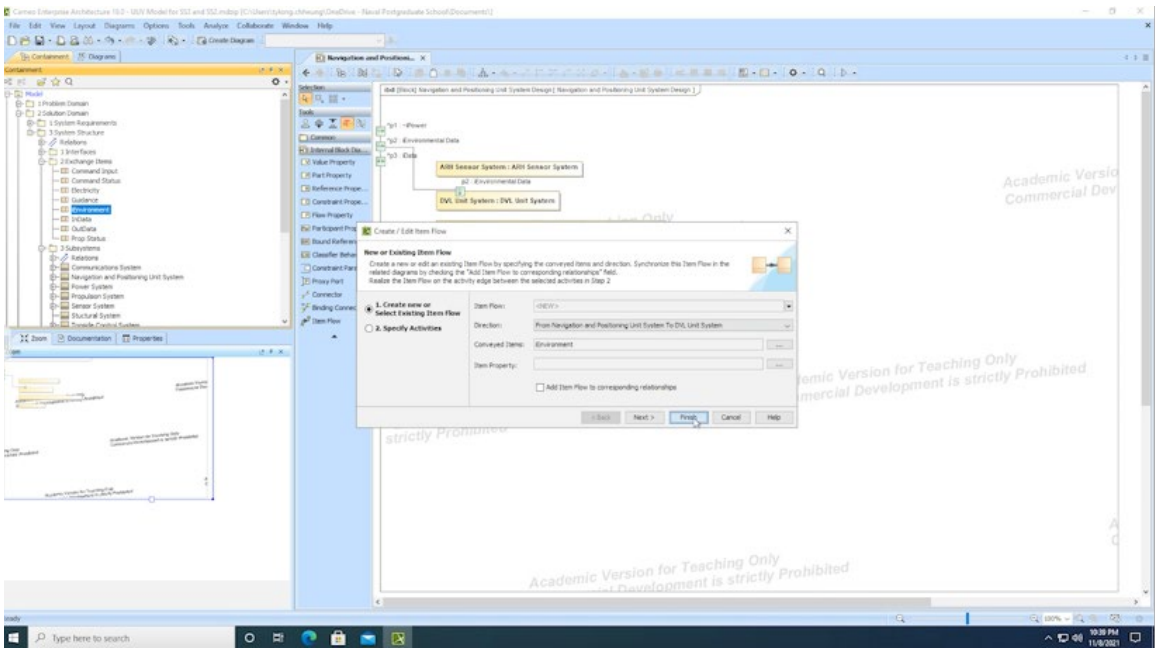
After selecting that option, a compatible proxy port is created to connect the *DVL Unit System* part property and the connector established to connect *DVL Unit System* parts property to the external part property.



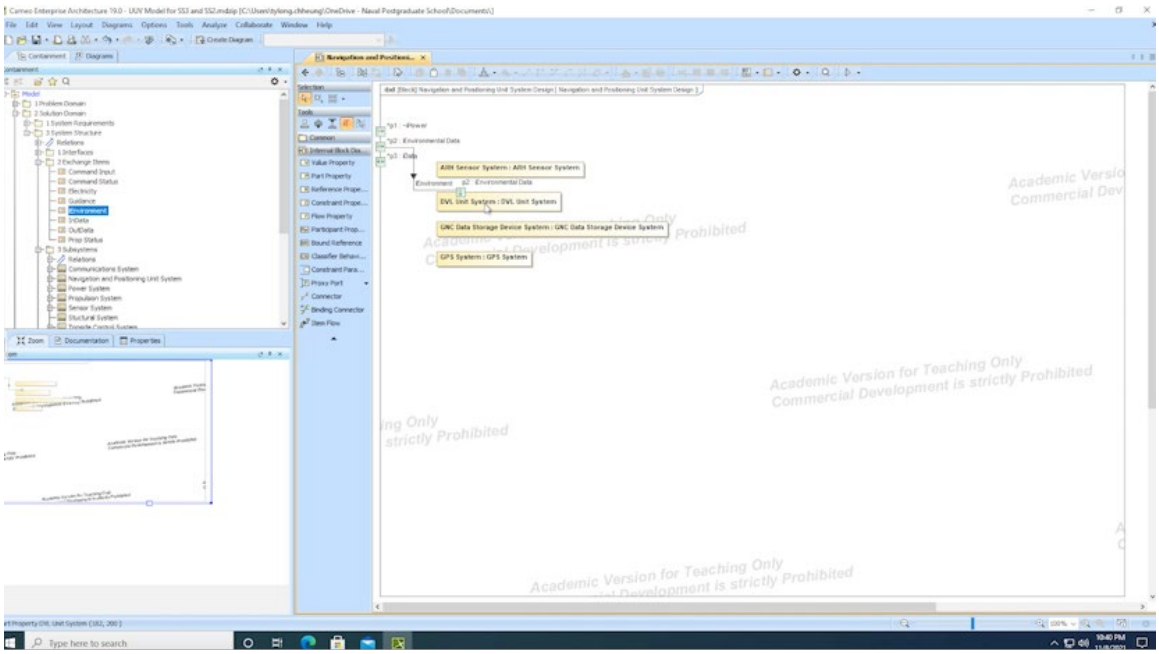
In order to distinguish what kind of information is being transferred from this connector, we will head over to the containment tree on the left-hand side. We will go to the *2 Solution Domain* folder and go to the subfolder *3 System Structure* and then expand the *2 Exchange Item* folder. Then we will be selecting *iEnvironment* Data Signal.



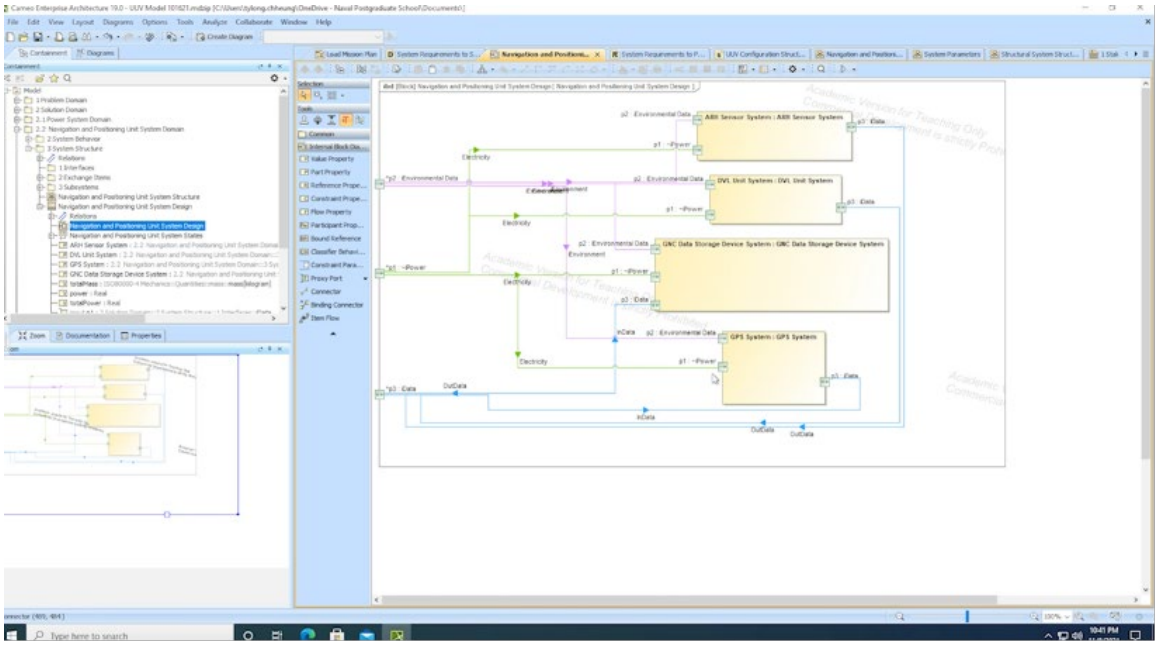
Then we will drag the *iEnvironment* Data Signal on top of the connection. Then the *Create/Edit Item Flow* popup window will appear and we will click on the *Finish* button.



Here it will signify that the information is flowing from the outside into the *DVL Unit System* parts property.



Go ahead and we will repeat the process for the rest of the part properties. Once you are done the completed model should look similar to the photo below.



And this concludes the tutorial on subsystem structure. Thank you.

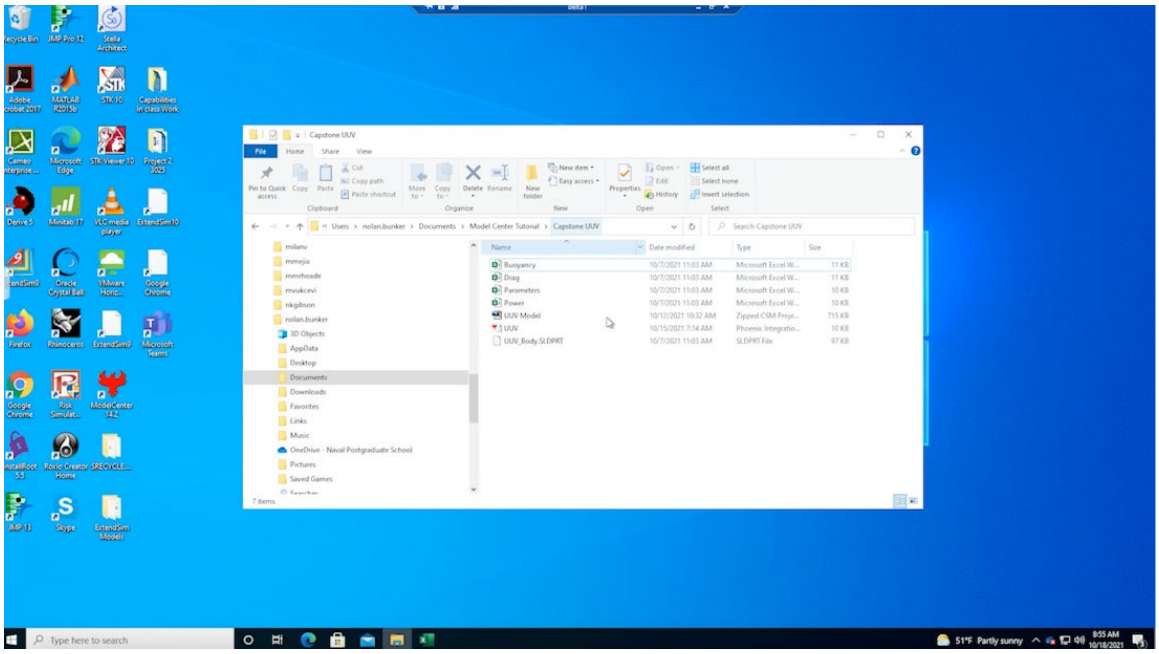
Model Center Integrate Tutorial I

Background:

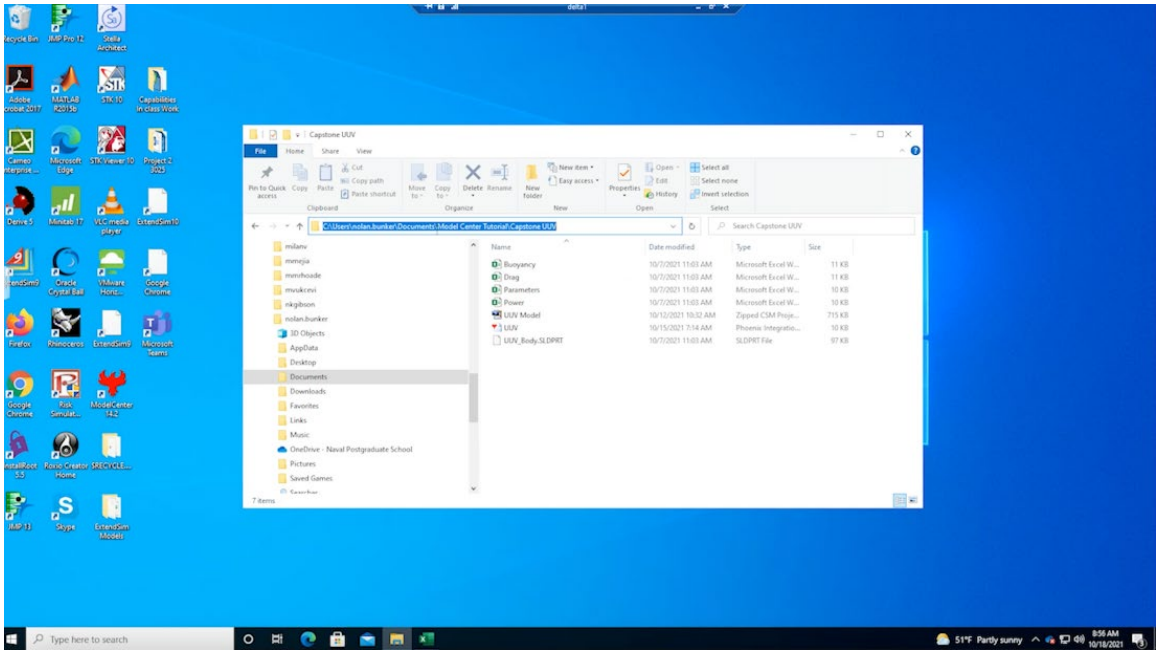
In this tutorial, we will discuss how to perform the steps necessary to set up a ModelCenter Analysis using Microsoft Excel files. These steps are important to accomplish in order to be able to execute the model simulation that will be created in the part 2 of this tutorial. Special attention is to be paid at the portion about excel file range names.

Step-By-Step Guide:

Welcome to this tutorial in how to create an executable model within Model Center using Excel files. To begin the method used in this tutorial is one of many different ways you can use to establish or create a working simulation. In this tutorial we will be using Microsoft Excel to create input variable and execute calculations within Excel, which will then be exported to Model Center for automation. Let's begin. First, you will want to initiate the *Analysis Server*. This is important in trying to make a connection between Model Center Analysis and a Cameo Model. This connection between Model Center Analysis and Cameo will be shown on a different tutorial video provide later. However, before we can move forward you will want to create a folder where you want your analysis path to link to. Go to whichever location you desire for your project and create a path. For my project used in this tutorial video I went into my Users folders located on the C: drive, which I will show you now. I went to my *Local Disk (C:)* drive and found the *Users* folder and then scroll down to find the user folder with my name. Under the Users folder with my name I went into the *Documents* folder and created a folder labeled *Capstone UUV*.

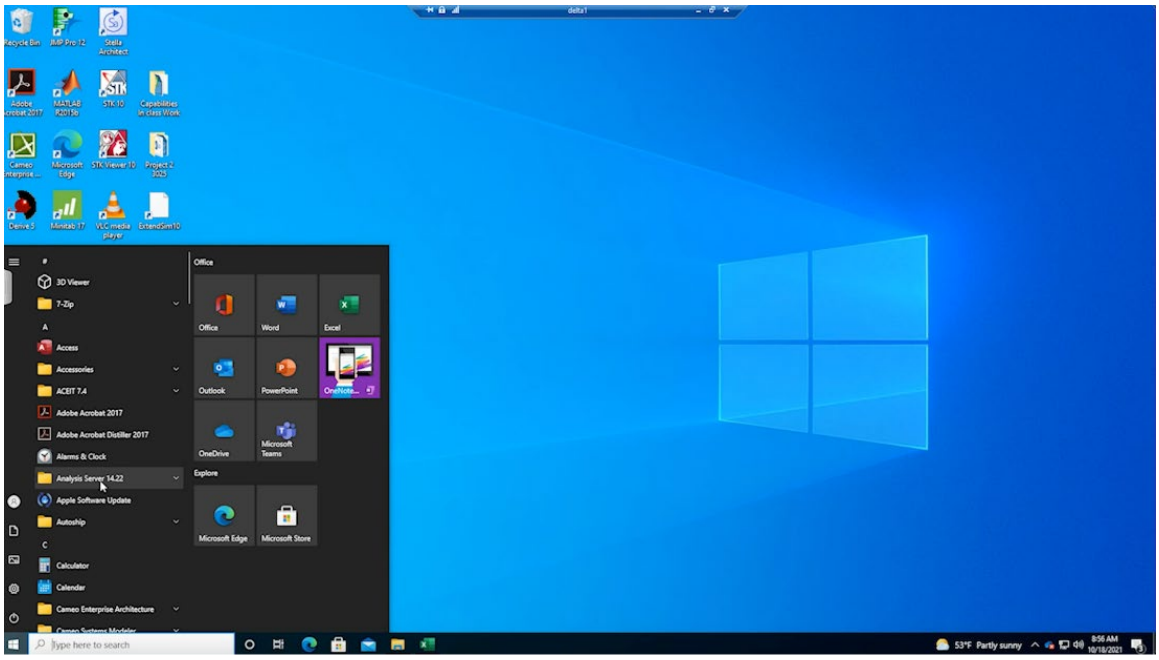


After you create your desire path you will click on the folders icon in the address bar. It will highlight the link path that you created to that folder.

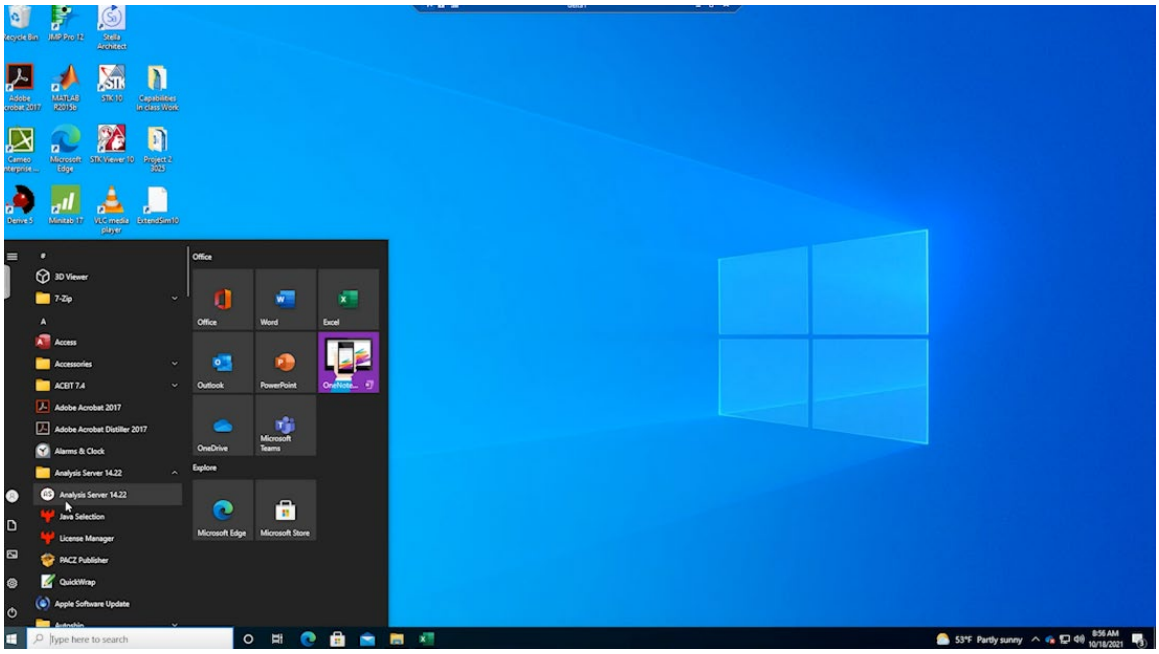


You can either use CTRL + Shift + C or you can right click on the highlighted path and from the drop-down menu select copy to copy the highlighted path. Once you created that path you can exit the folder or minimize the folder. The next step is to open the Analysis Server Tool and you can do this by going to the Start Menu and locating your

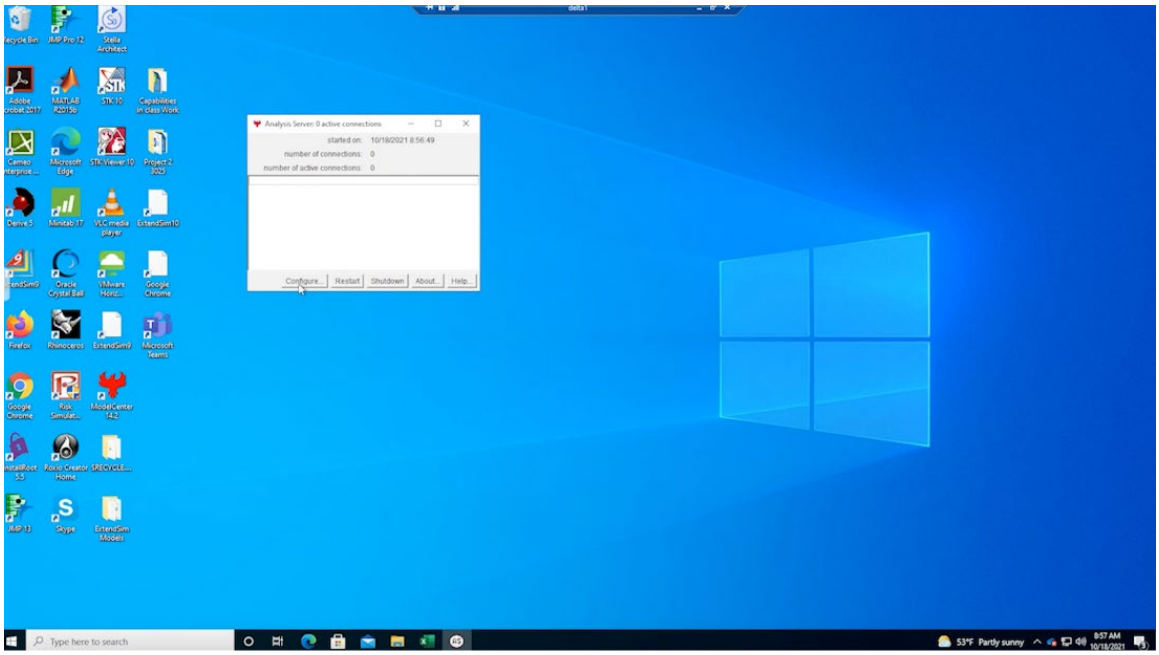
Analysis Server. For this tutorial, the Analysis Server Tools version 14.22, but that does not mean that your will not work if it's an older version.



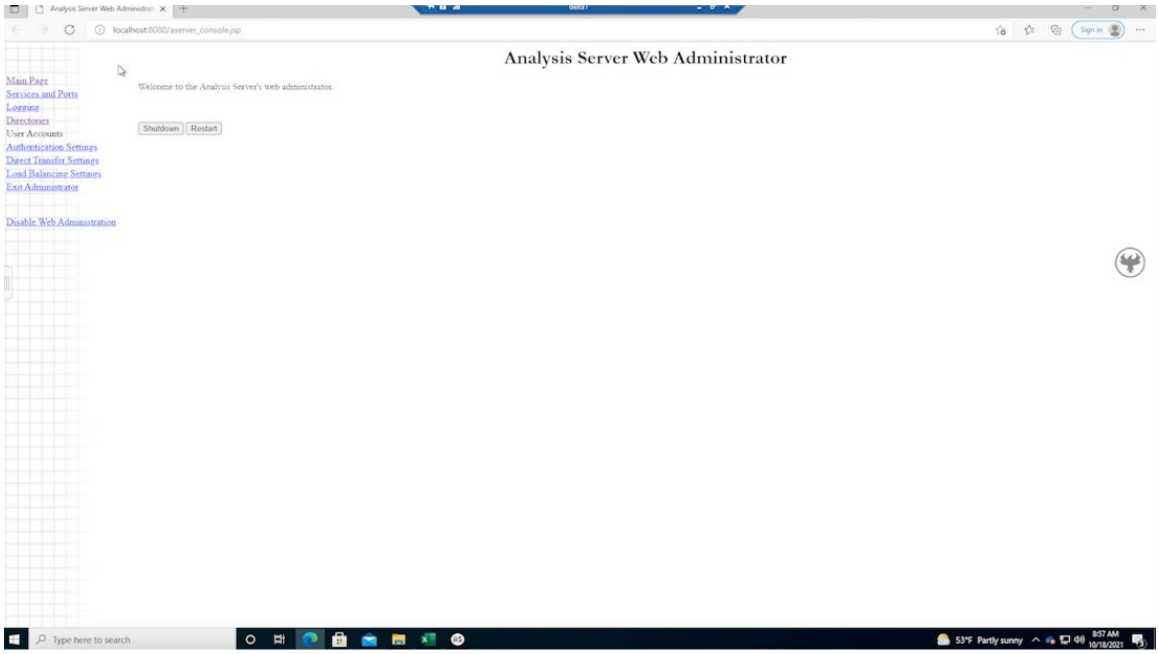
Click Analysis Server Tool folder and scroll down the drop-down menu and select Analysis Server 14.22 application.



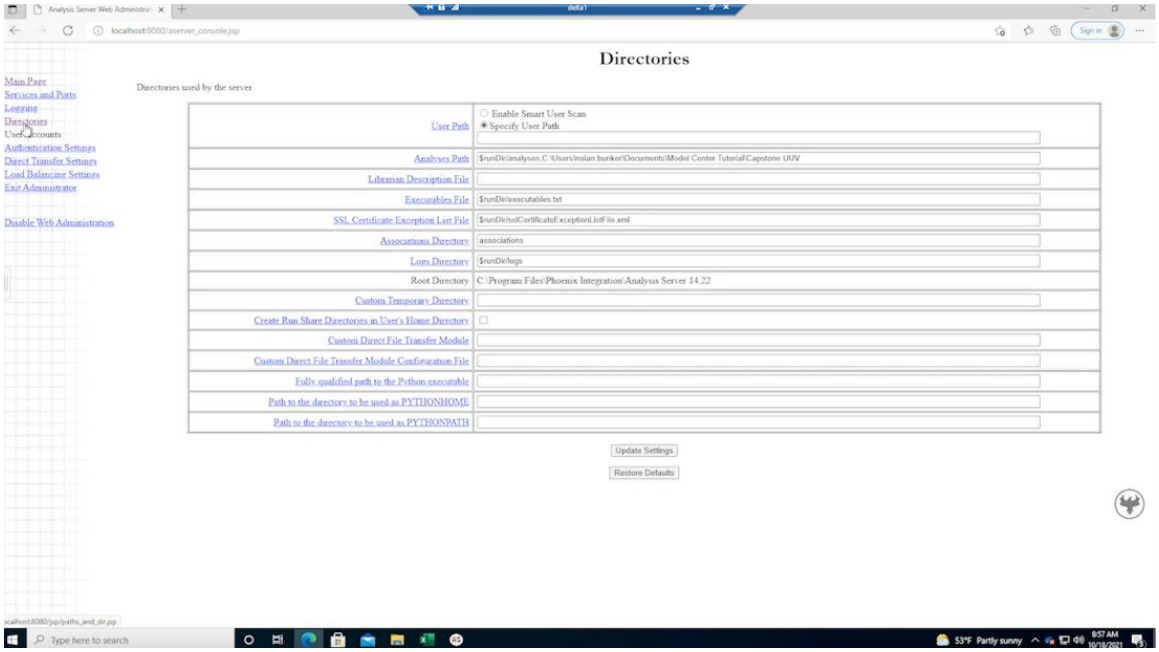
Then the Analysis Server Tool popup window will appear. In order for the Analysis Server Tool to work, you will need to link it to the path folder you just created. You do this by clicking on the Configure button.



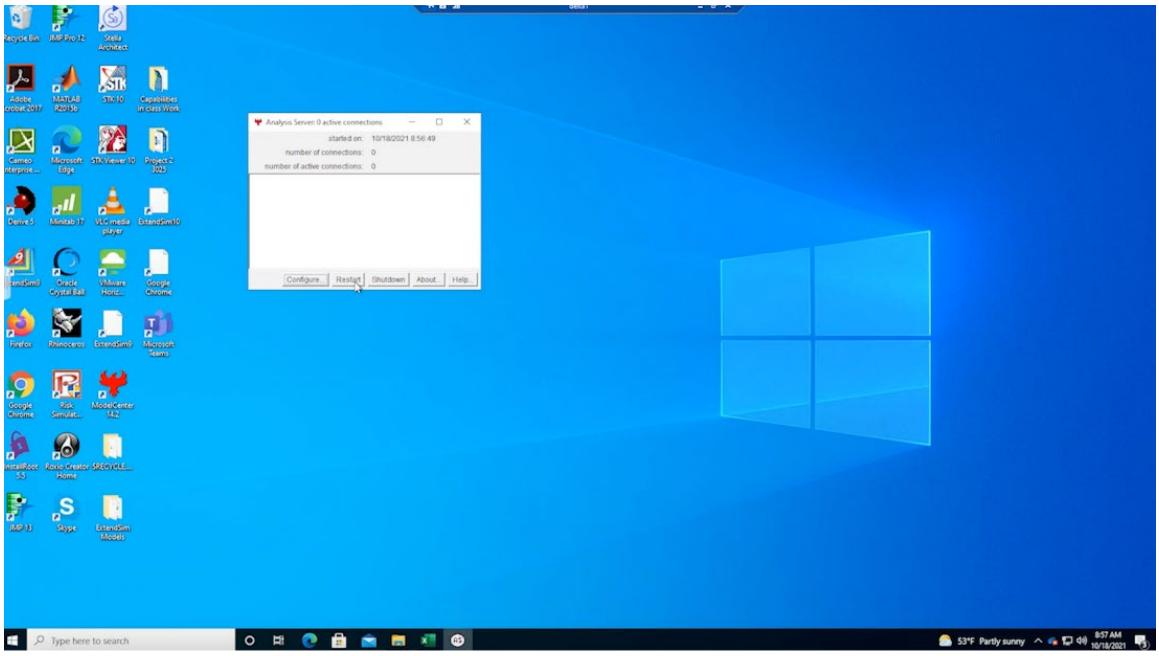
And it will bring up the Analysis Server Web Administrator window.



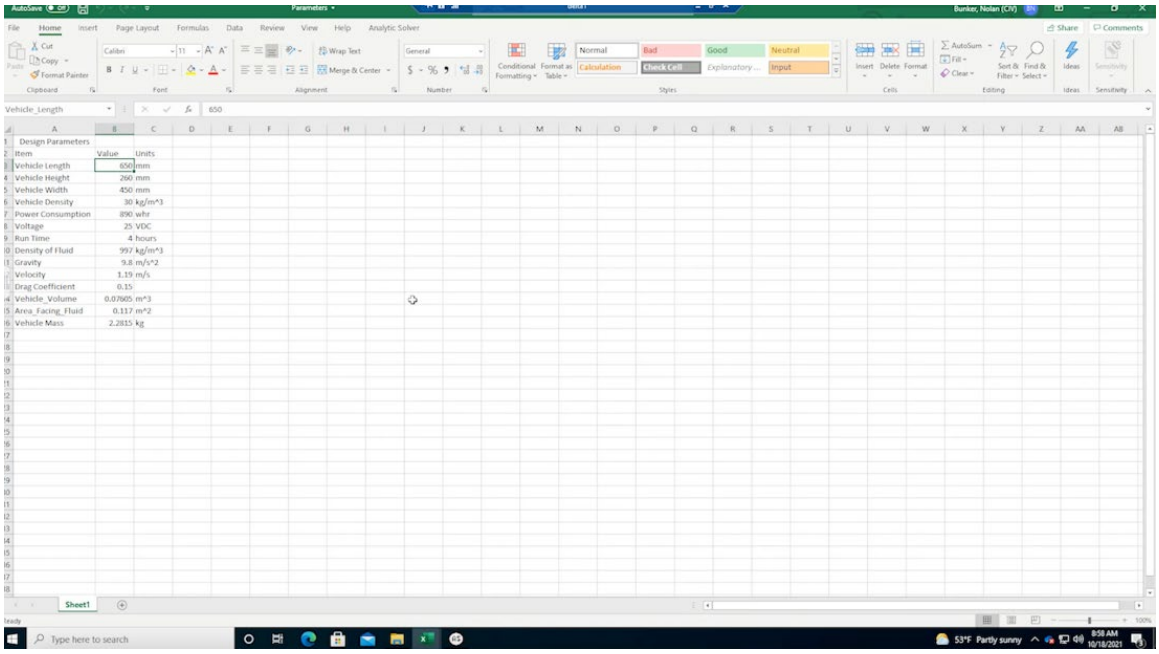
Go over to the left-hand side menu and click on the Directories hyperlink and this will bring out the Directories tab. Under the Analyses Path you can CTRL + Shift + V or right click and paste the path link you copied from your folder. Ensure that the $runDir/analyses and you can separate by a semicolon (;). Once you have done that then press the Update Settings button. This may require you to restart the Analysis Server if prompted, however, as nothing was changed for this tutorial then a restart is not needed, and we can move forward. After you are done you may close the window or minimize the window.



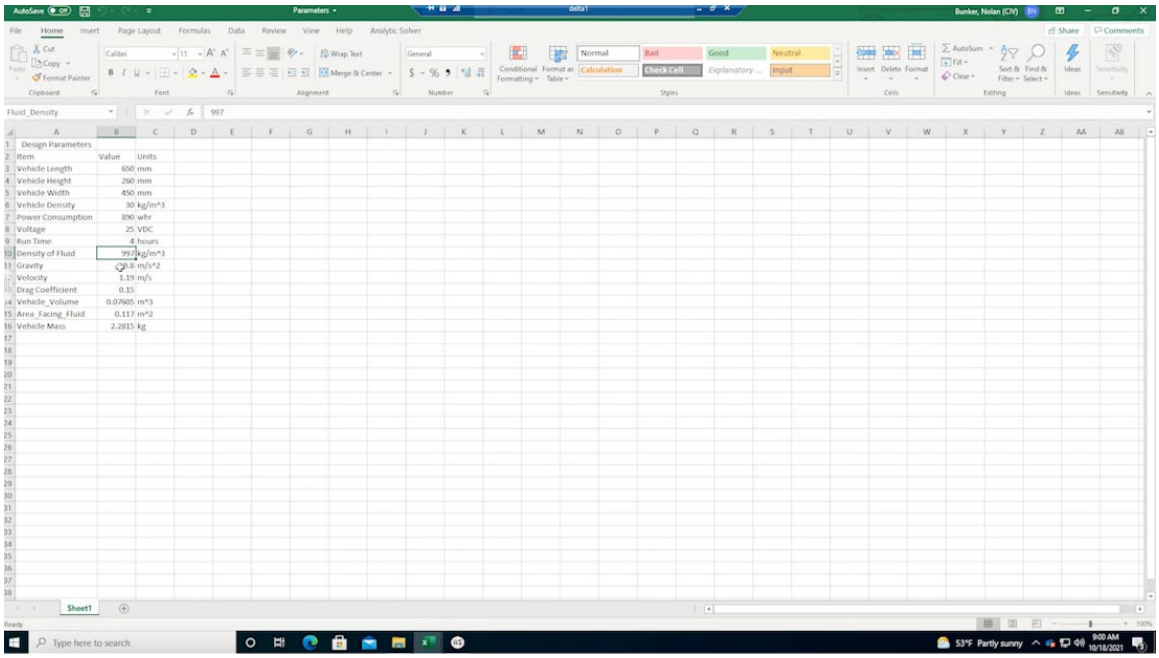
As I mentioned above, if it requests you to restart the *Analysis Server* then you can do so by clicking on the *Restart* button. After you have completed all that then you can minimize the *Analysis Server* window.



The next step in creating a Model Center Simulation is creating the Excel files you will use once you import them. To save some time on this tutorial, we have already created some excel files that we will be going over in order to use them for this tutorial. However, the knowledge you will need to do this is not extensive. The only thing you will need to understand is how to put information into the Excel cells, to do calculation for whatever values you want to have as outputs and be able to do range names in the cells, which will be shown in this tutorial. We will begin now shown the premade Excel file for this tutorial. First, Excel file that will be shown in this tutorial is the Parameters Excel file. As I mentioned you will just need to understand how to put cell values and be able to do range names. However you set up your Excel files is up to you including any best practice you like to follow. For this tutorial, we have set the Excel file as followed heading names and split them out based on what each one is. For the Parameters Excel file we have labeled it Design Parameters. Then we have an items tab, a values tab, and a units tab. In the items tab, we have listed out each variable that we will be using or at least tracking in our simulation.



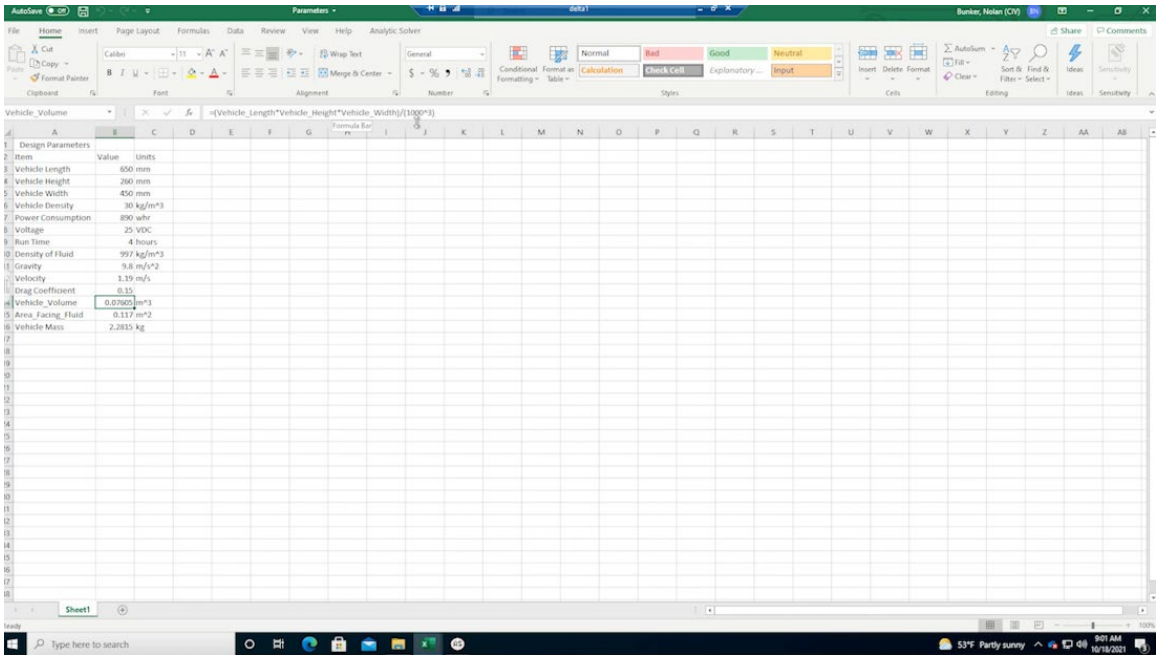
Now some of these variables are considered inputs and others outputs and as we move through this tutorial we will be identifying which one is which. The Vehicle Length, Vehicle Height, Vehicle Width, and Vehicle density are all vehicle inputs. The other inputs are Power Consumption, Voltage, Run Time, Density of Fluid, Gravity, Velocity and Drag Coefficient. In this excel file the cells from 3 to 13 are all inputs. To calculate for the outputs, you have to do standard calculations in Excel. However, before you start on the calculations you do range name for each cells. A range name is a specific name that you give to a particular cell. For instance, in the case of the Vehicle Length we named the value which is cell B3 as Vehicle_Length. This is important we you use Model Center because Model Center only recognizes range names when you import the Excel file. If you don't have a range name then Model Center will not read that there was a variable that was suppose to be there and you will be missing one of your inputs. So before you do any of the output calculation, you will need to assign range names to each of the input cells. Each input has their own specific range name that is associated with that particular cell.



Design Parameters

Item	Value	Units
Vehicle Length	650	mm
Vehicle Height	260	mm
Vehicle Width	450	mm
Vehicle Density	30	kg/m^3
Power Consumption	890	whr
Voltage	25	VDC
Run Time	4	hours
Density of Fluid	997	kg/m^3
Gravity	9.8	m/s^2
Velocity	1.19	m/s
Drag Coefficient	0.15	
Vehicle_Volume	0.07605	m^3
Area_Facing_Fluid	0.117	m^2
Vehicle Mass	2.2815	kg

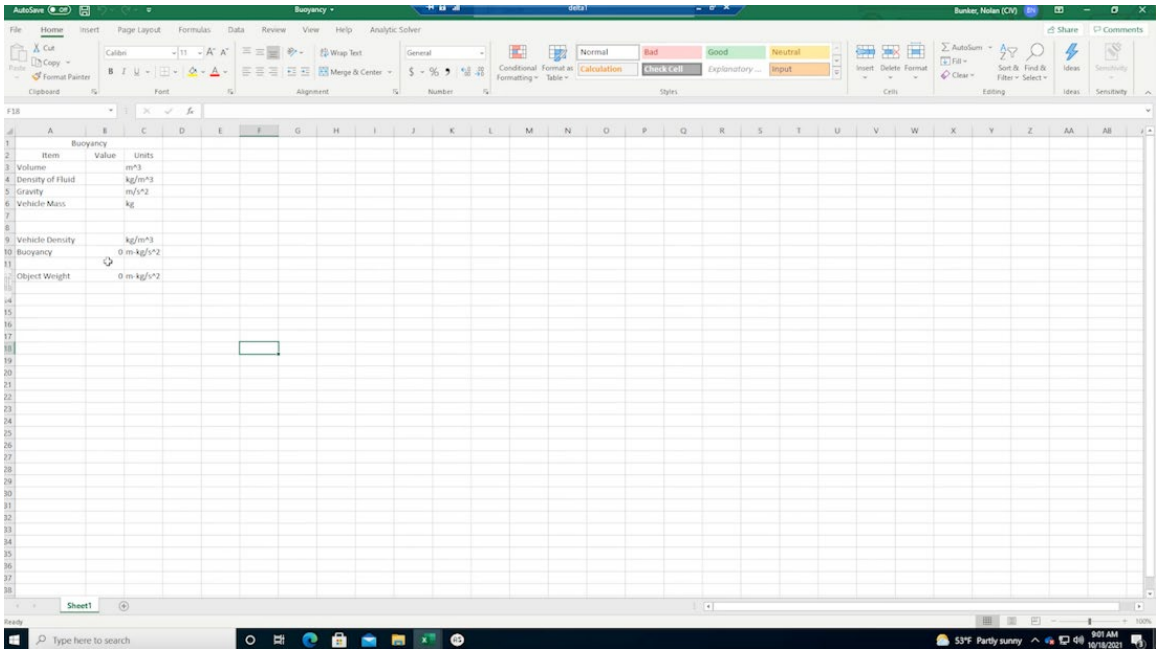
Once you do the range names you can do as it was mentioned above the standard calculations using the range names and it will give you a value. And this can be done for all output values.



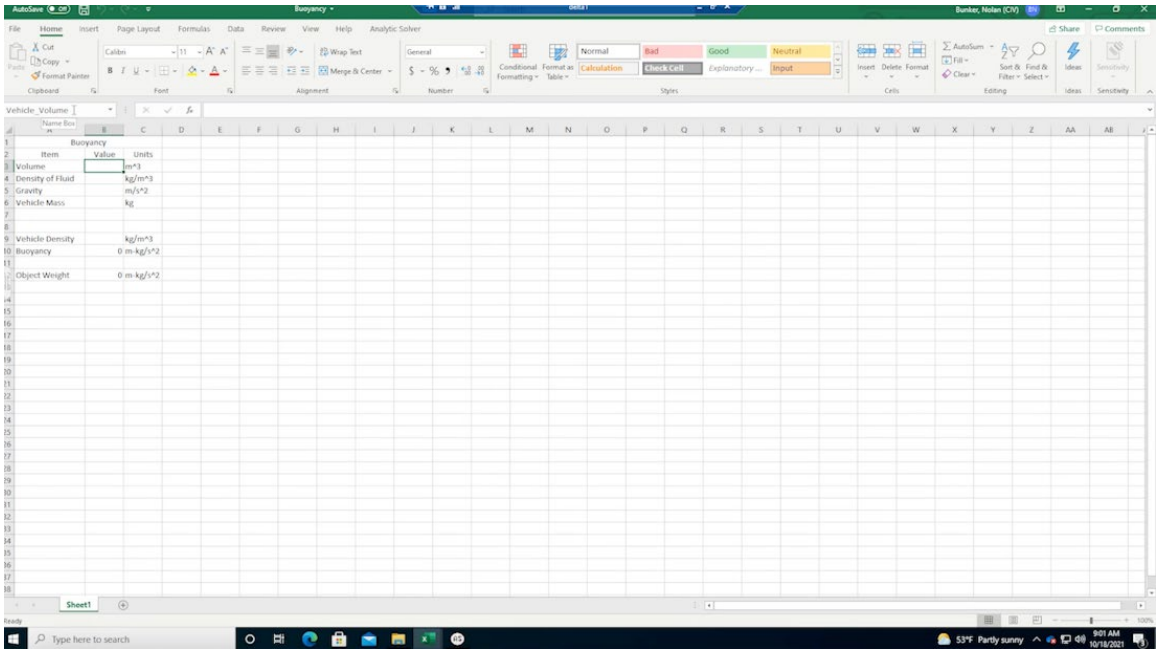
Design Parameters

Item	Value	Units
Vehicle Length	650	mm
Vehicle Height	260	mm
Vehicle Width	450	mm
Vehicle Density	30	kg/m^3
Power Consumption	890	whr
Voltage	25	VDC
Run Time	4	hours
Density of Fluid	997	kg/m^3
Gravity	9.8	m/s^2
Velocity	1.19	m/s
Drag Coefficient	0.15	
Vehicle_Volume	0.07605	m^3
Area_Facing_Fluid	0.117	m^2
Vehicle Mass	2.2815	kg

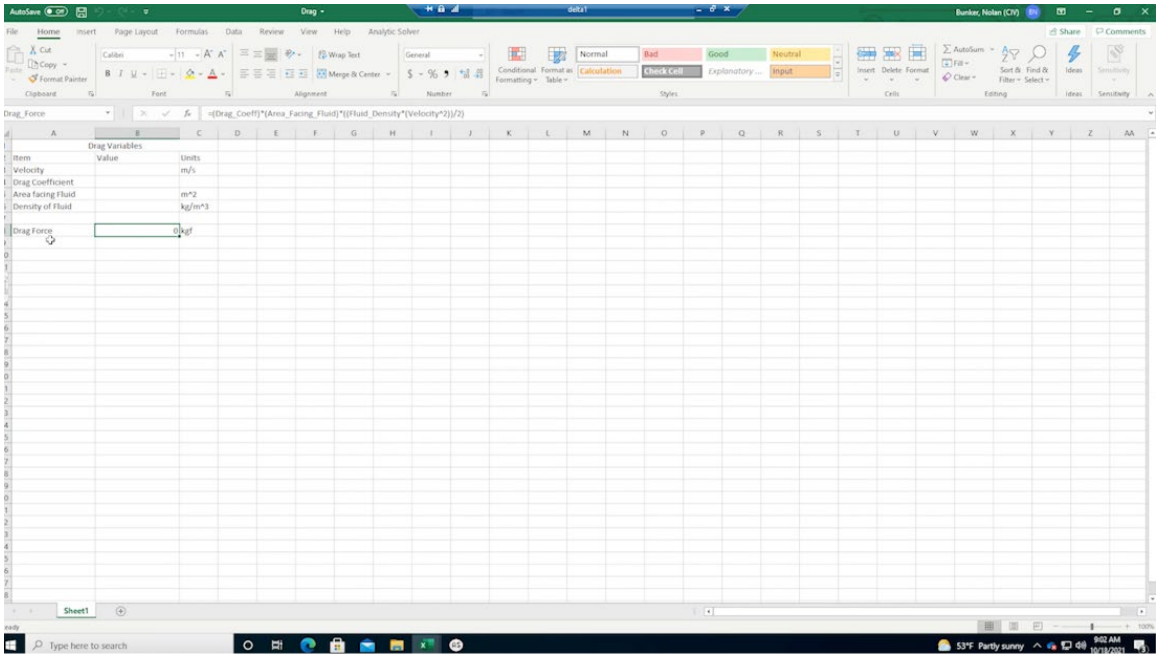
Now we will bring up other excel sheets that we will be using in Model Center that will help you create your model. So one of the values we are trying to solve for is Buoyancy. For this Excel sheet you can notice it is a bit different that the Parameters Excel sheet because we don't have any values listed in the inputs nor the outputs.



The reason for not listed values in the Buoyancy Excel sheet is that when it is loaded into Model Center you will link these values to the inputs of the Parameters Excel sheet and they will automatically populate within Model Center. Also it will run the calculations that were set in the Excel sheet. That being said we recommend that you name each of the cell values the same way as you named them in the Parameters Excel sheet. For instance in this Buoyancy Excel sheet we named the value of Volume as Vehicle_Volume. For this tutorial the range names in the Buoyancy Excel sheet matched the range names used in the Parameters Excel sheet, except for Vehicle Mass. This was done to show later in the tutorial to show you that it will still work if for some reason you accidently not named it the same.



For the outputs you will need to need to type your calculations. It may not have any variables being populated as they are all zeros, but the Buoyancy calculation is crucial for when you import it into Model Center. Otherwise, that particular calculation won't solve for anything. Another excel sheet that we will look at is the Drag Variables Excel sheet. Again, the same concepts apply where the values are not entered into the Excel sheet.



However, the Drag Force calculation has all the range names entered in. That is essentially the only thing you have to do with the Excel sheets and save each one as an individual Excel file. Therefore, every time you want to solve for something make sure you have a separate file. Otherwise, it becomes very tedious and confusing inside Model Center as it will bring up every range name that you have in that Excel sheet regardless of if you have different sheets within the Excel file. Make sure that before you go into Model Center that you have your Excel file exactly formatted the way you want it. If you make a change in your Excel file it is sometimes difficult to get that change to show up within Model Center. Therefore, sometimes it is easier to delete the excel file from Model Center model and re-input your Excel file. This concludes the first part of the Model Center tutorial. Thank you.

Model Center Integrate Tutorial II

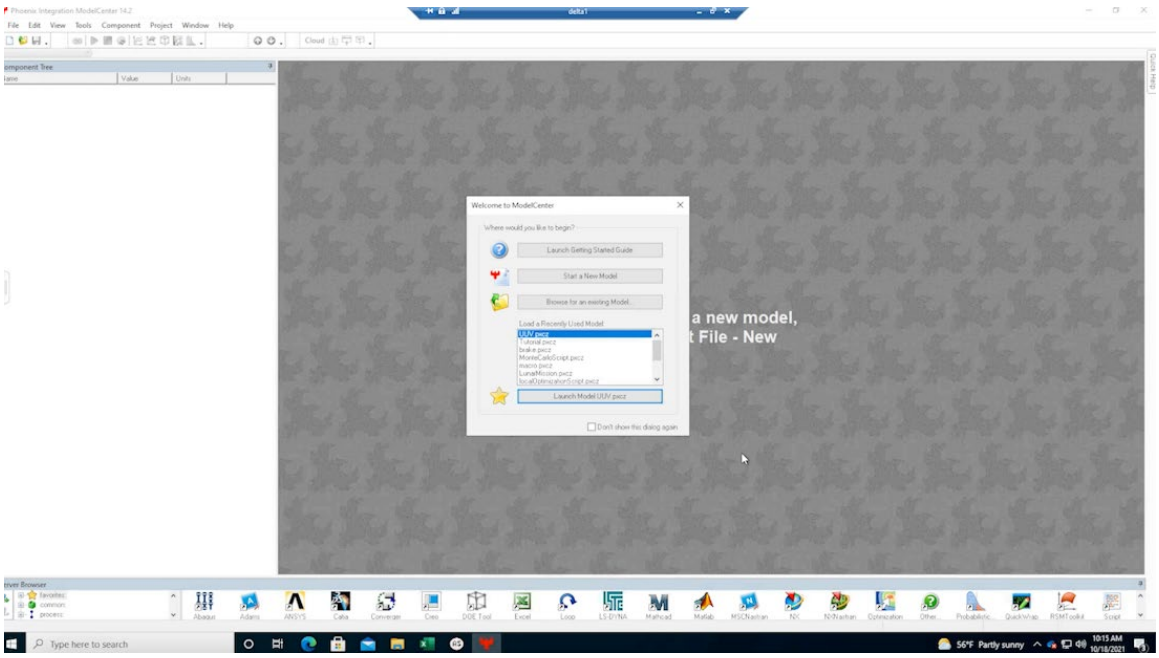
Background:

In this tutorial, we will build upon the steps that were taken in part 1. However, for this tutorial, we will dive into ModelCenter and begin linking the excel files created in order to conduct specific calculations. These analyses created in this step are integral to the integration with Cameo and ModelCenter MBSE. Which in turn leads to the ability to conduct optimization studies for the model created within Cameo.

Step-By-Step Guide:

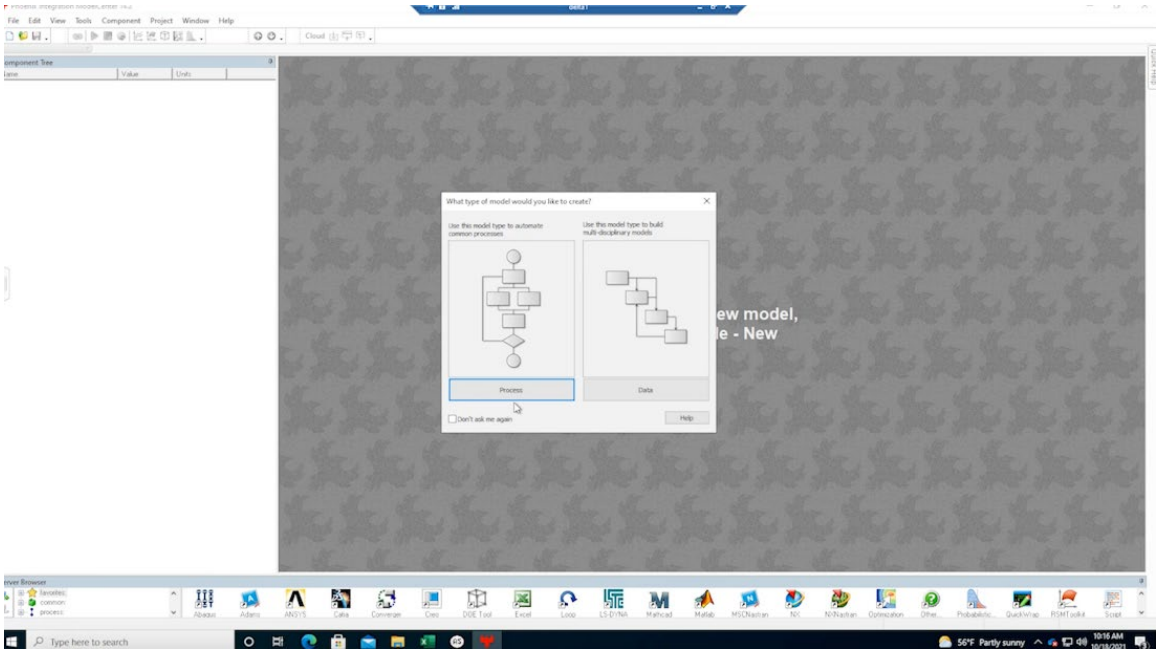
Welcome back to the tutorial on incorporating Microsoft excel sheets to Model Center. Last time we talked about how to setup the Analyses Server where you can save the results from Model Center to your computer. We also talked about how to create the excel file that will be imported into Model Center and what to look out for when you create these excel sheets. So that now those files are created and have the Analyses Server linked,

we will continue onto Model Center for this portion of the tutorial. So if this is the first time you have opened or created Model Center, you can go by first opening the Model Center application. In this tutorial we have the version Model Center 14.22. Once you click on it to open it you will have to let the Model Center application load in order to get the next screen. As mentioned before, if this is your first time opening Model Center, it will bring a welcome screen as shown in the screenshot below.

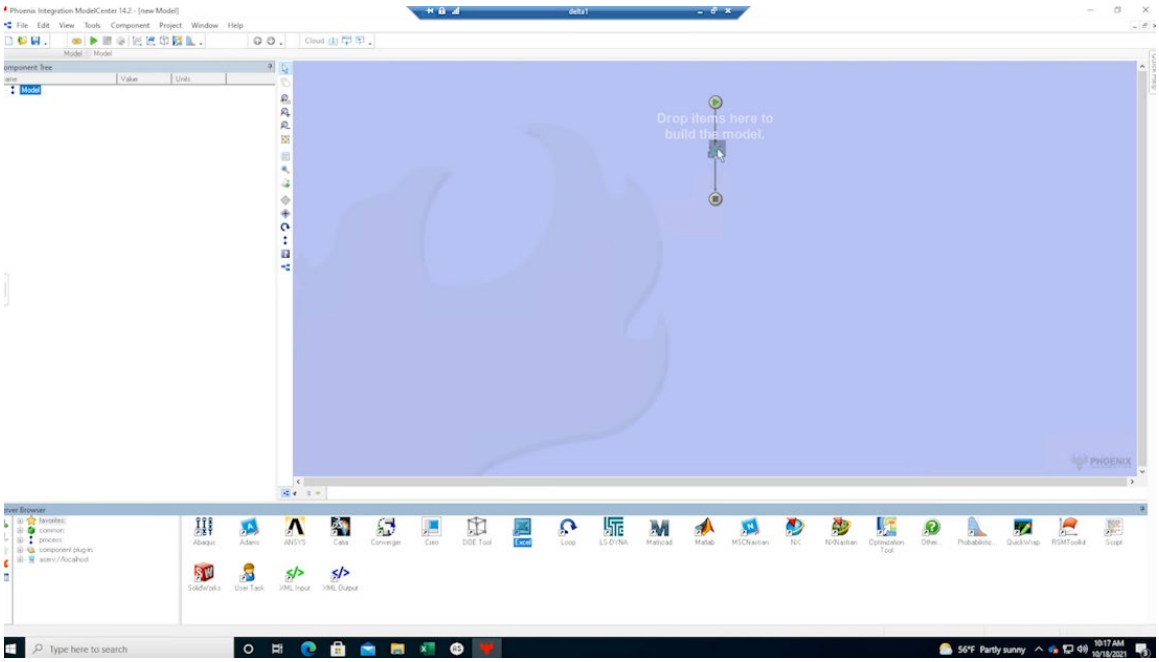


If this is indeed your first time then you will click on the Start A New Model button on the Welcome To Model Center popup window. In this tutorial we will start a new model, therefore, we will be pressing the Start A New Model button. If this is not your first time opening Model Center and you already have an existing model then you will look into the Load A Recently Used Model drop-down menu. For instance the model of the UUV that we created prior this tutorial, we will have to select the UUV form the drop-down menu and click on the Launch Model(UUV). For this tutorial we will walk you through on how to setup a similar model to our UUV model. First, you will need to click on the Start A New Model button and after it will bring up two choose for the different types of models. The question that Model Center poses to you will be What type of model would you like to create? The two choose are Process and Data. A Process model is used to automate

common processes. The Data model is used to build multi-disciplinary models. For this tutorial we will be choosing the Process model because it is the most relevant to what we will be doing in this tutorial.

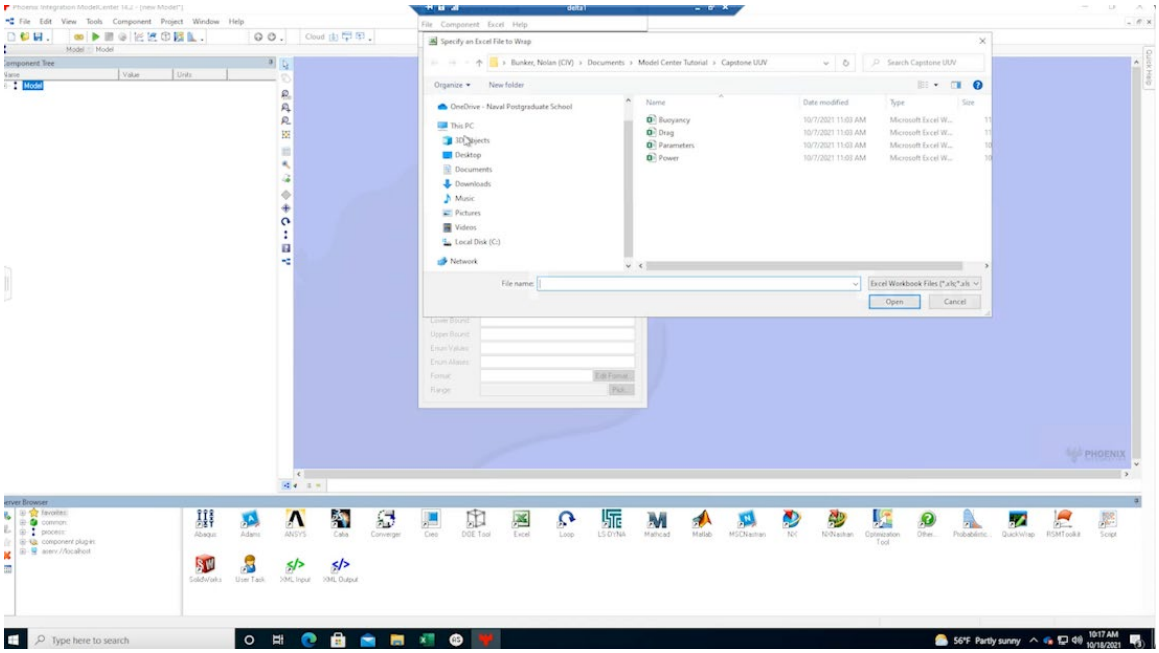


It will bring up the following screen as shown in the screenshot below. If you increase the bottom portion of the screen, you will be able to see the different tools Model Center allows you to use. For instance program will SolidWorks, Excel, Matlab, and etc.

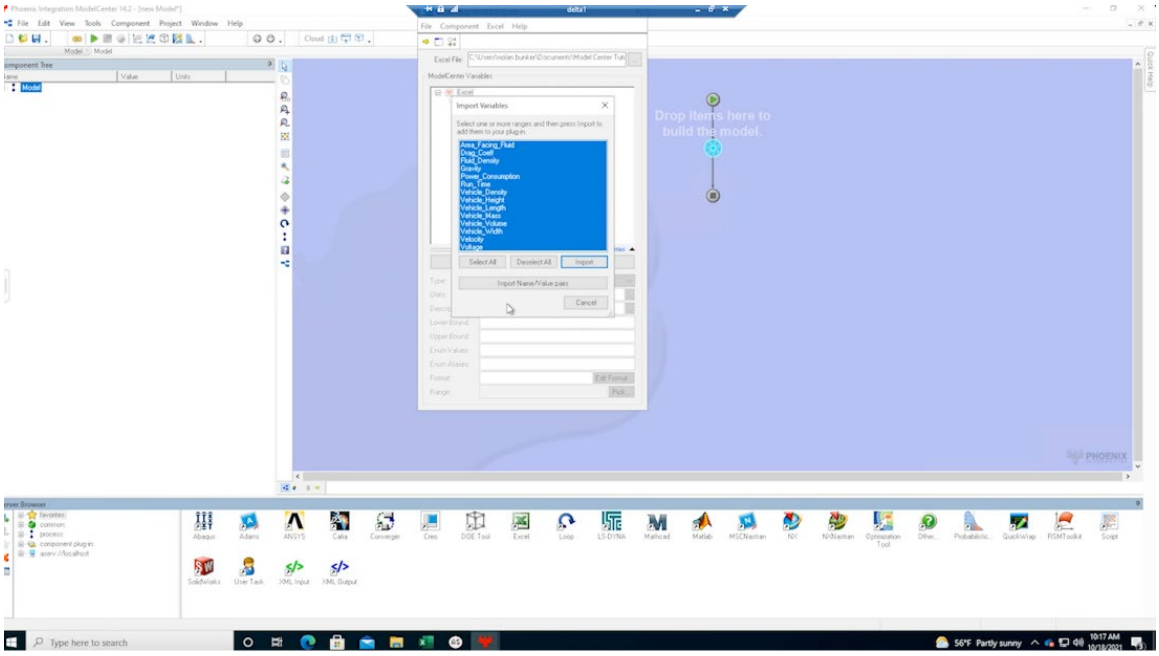


For this tutorial will be using excel and in order to start this whole process you will click and drag the excel icon to into the model.

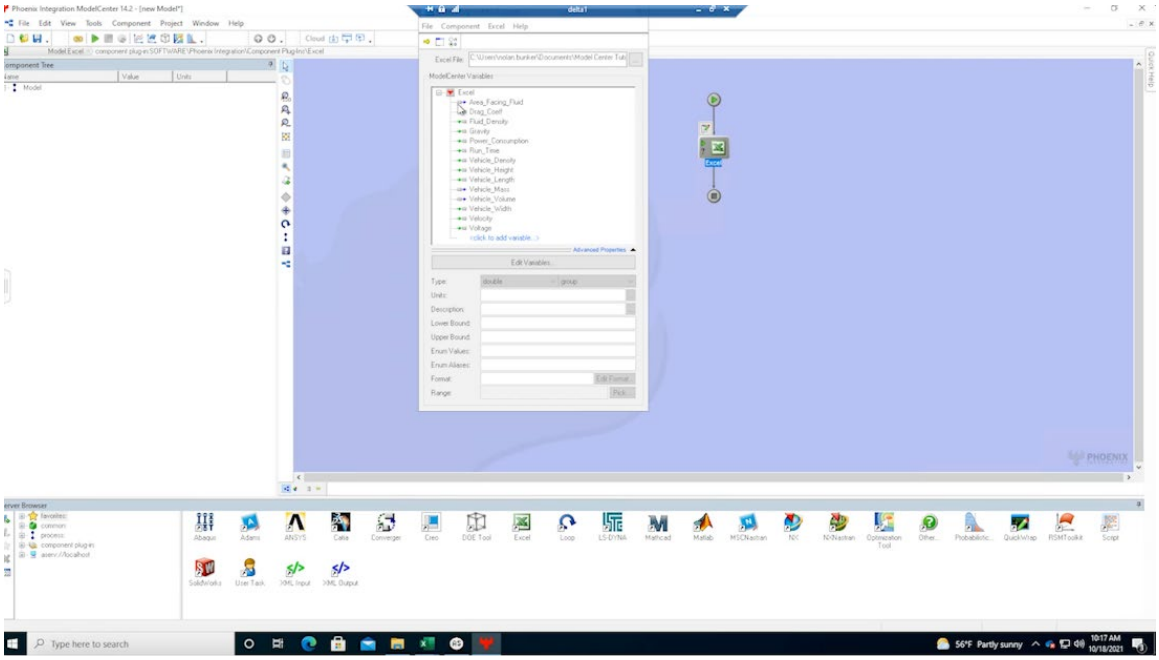
As you can see it will highlight the Excel icon and on the model, a little white circle pop up when hovering over the in the center. You will drag and drop the excel icon on top of the little white circle, which will then open a series of popup windows where you can link your desired Excel file. The popup window will request that you upload an Excel file.



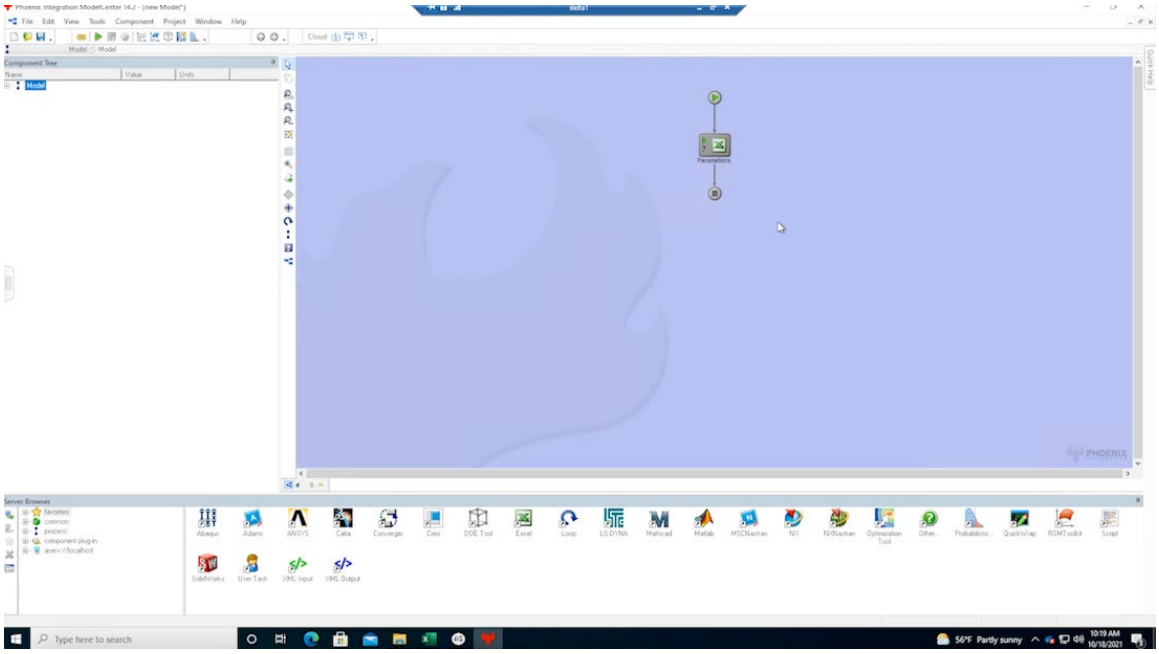
We will be choosing to upload the Parameters Excel sheet since it is the governing Excel sheet that feeds into the other three excel sheets. After selecting the Parameters Excel sheet you will need to click on the Open button. This will bring a popup window with a list of all the range names it found within the Excel sheet. You will just need to make sure that you have necessary range names that you are wanting and it can be found on the list. If the range name you are looking for is not found on the list then you will want to stop here with Model Center and go into your Excel sheet to figure out what you missed and then come back to Model Center.



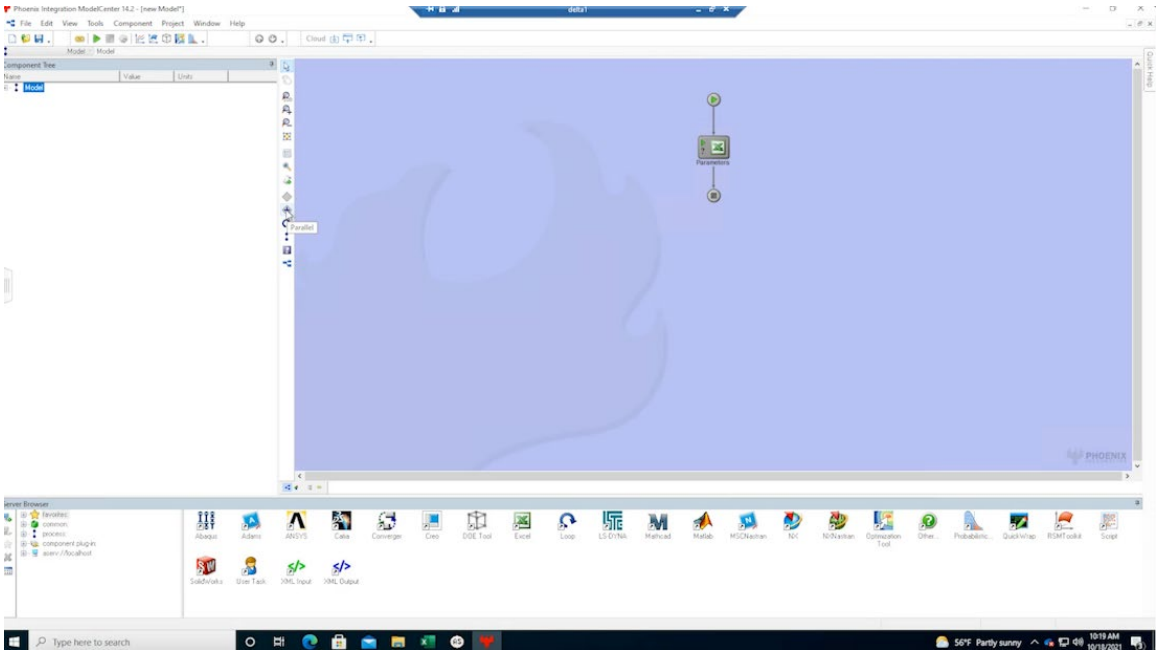
Assuming that everything is correct you will then hit the Import button and it will bring you to the next step as shown in the screenshot below. Within the popup window locate the Excel folder with the Model Center logo and under you will noticed a box with a blue arrow icon for Area_Facing_Fluid, which is for used to indicate an output. And the green arrow into the box represents the inputs. All inputs and outputs will have something there as it being pulled from the Excel sheet. For this one it is pretty straight forward you don't have to worry about much just ensuring that your outputs are assigned as outputs and your inputs assigned as inputs. If you don't have the calculations being conducted on your Excel sheet it won't see it as an output instead it will listed as an input.



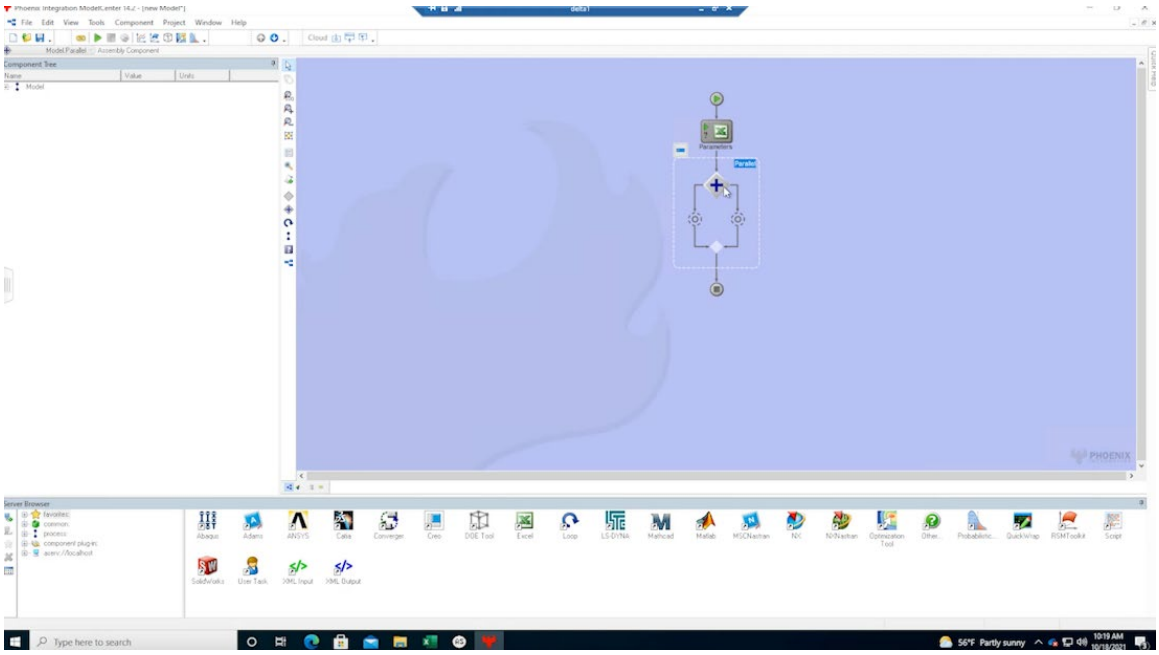
Other than that, you can input units if you want something about the units though that if you don't have the match between the other Excel sheets it will give you an error. It didn't seem to prevent anything from taking place in regards to the calculations, but it will give you an error saying that the units are not matching and that you need to correct that. Therefore, for this one we will leave the units off for now, but you can put units you just have to make sure that they are the same whenever you link the two inputs or two outputs. For now we will accept the applied changes and we will close the popup window. Double click on the text field named excel and label it Parameters.



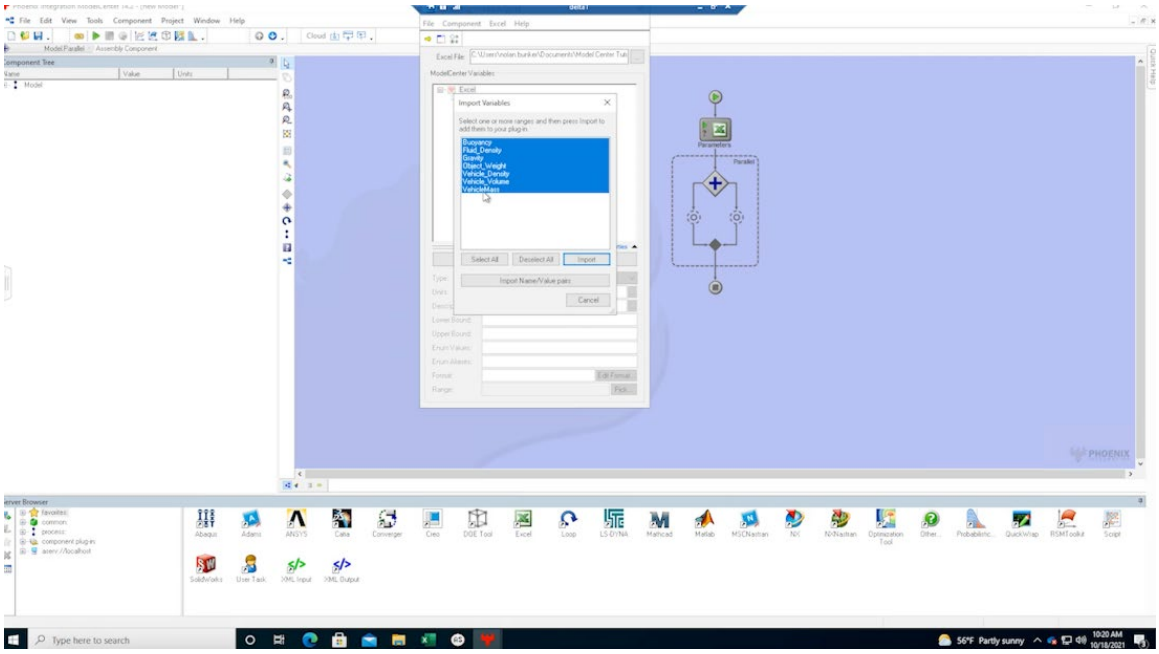
The way we set up the model since Buoyancy, Drag, and Power are all using the Parameters inputs as variables into their own calculations. We created a parallel using the Parallel button that is a square on its side and it has a cross in the middle.



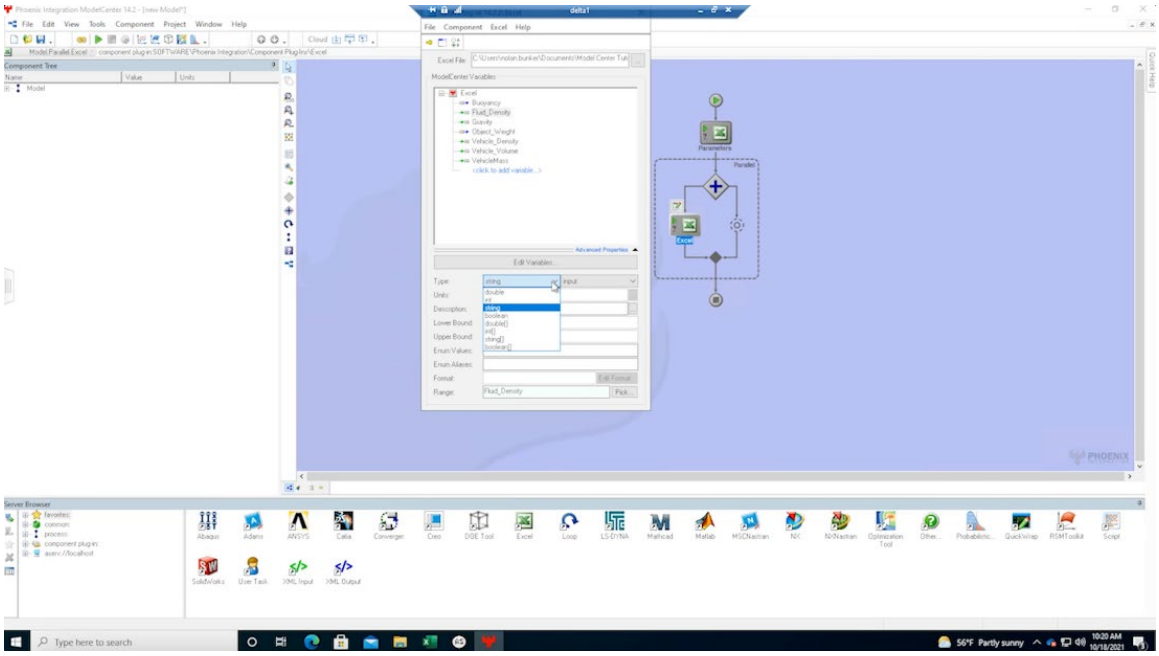
Once you select the Parallel button you have to drag it on top of the Parameters Excel model icon and then it will create a parallel branch.



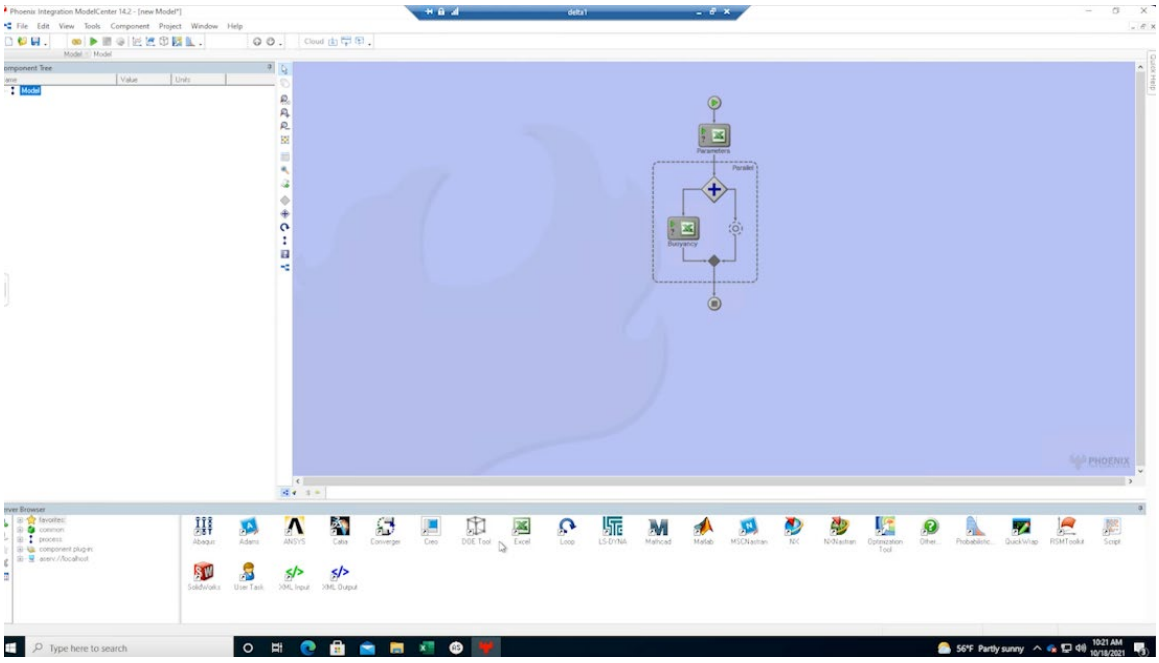
Then from there we can drag the new Excel sheets to the white circles. For the first Excel sheet will be adding will be Buoyancy and again make sure all the range names are the same or how you wanted them to be. As I mentioned before that Vehicle Mass is different between Parameters and Buoyancy Excel sheets. We will show you how to adjust that discrepancy later in the tutorial.



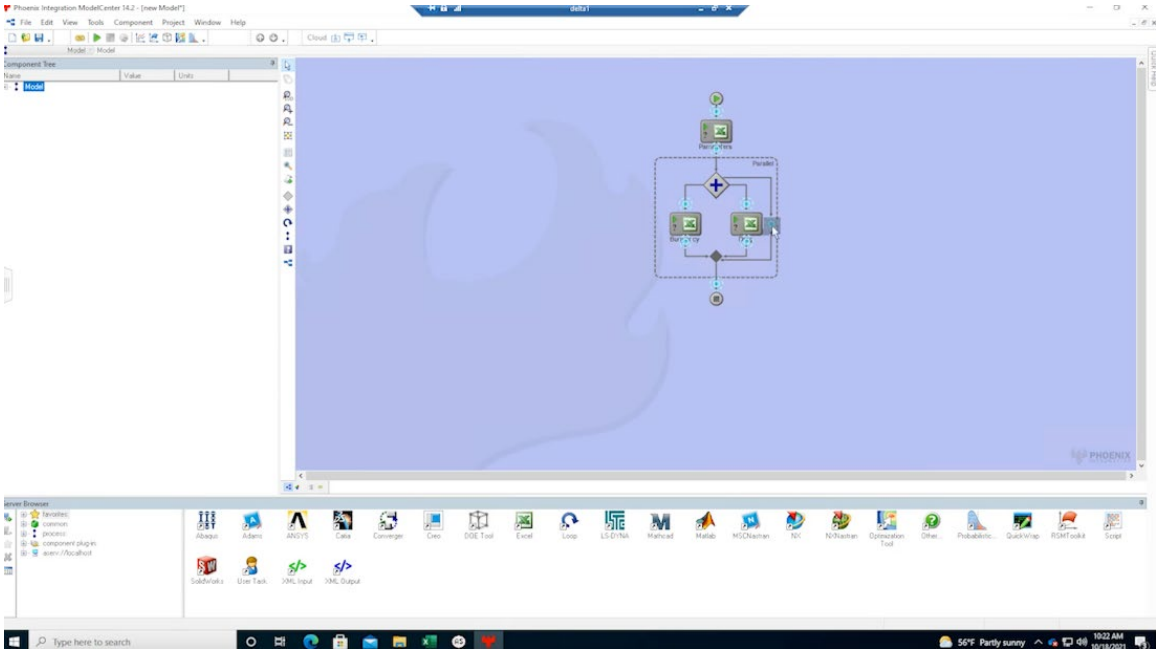
Everything looks good, therefore, we will select the Import button to import all the inputs and outputs. We have to check that import went well and the inputs are inputs and outputs are outputs with the attached arrow with the box icon. One thing to noticed though that if you are doing units, for instance, the variable type says String which when using a string type you cannot input to the units. So in order to input into the units you will need to click on the type drop-down menu and select Double. After changing the type to double, you are able to change the units to whatever you want. You will have to change the variable type from String to Double for each of the inputs. The outputs remain at double, therefore, there is no need to change the variable type. This is the process needed whenever you are doing units, but it is also a good practice to change the inputs variable types to Double so if you ever decide to do units that you do not need to go back make any changes to the variable type.



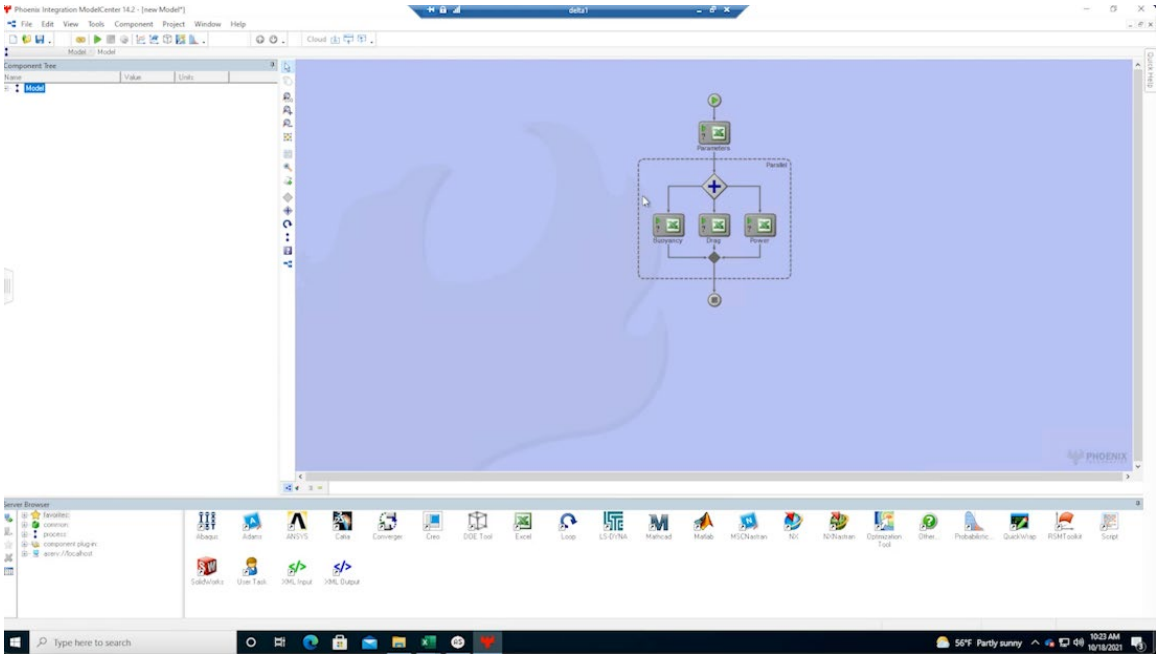
So now again we will go to apply changes and close out the popup window. Once again, we will double click on the text field named excel and label it Buoyancy.



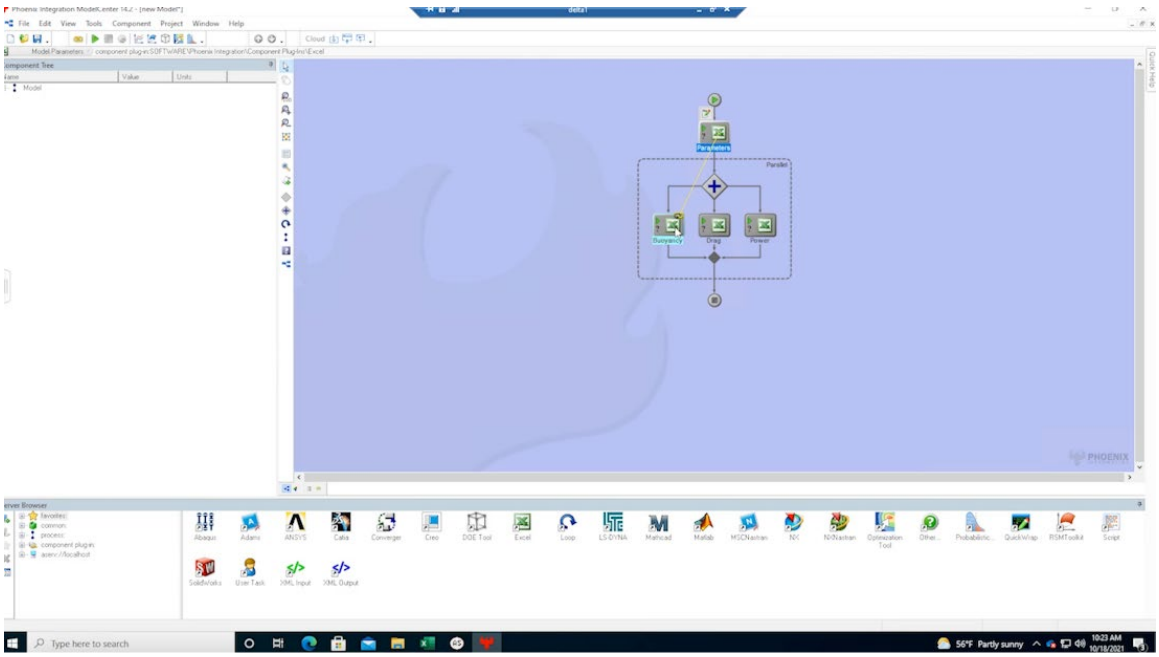
We will repeat this process for the remaining Excel sheets which are Drag and Power. And don't forget to change the variable type for all the inputs from String to Double. To add any additional parallel branches just drag the Excel icon to the white circle next to the Drag Excel icon in this instance.



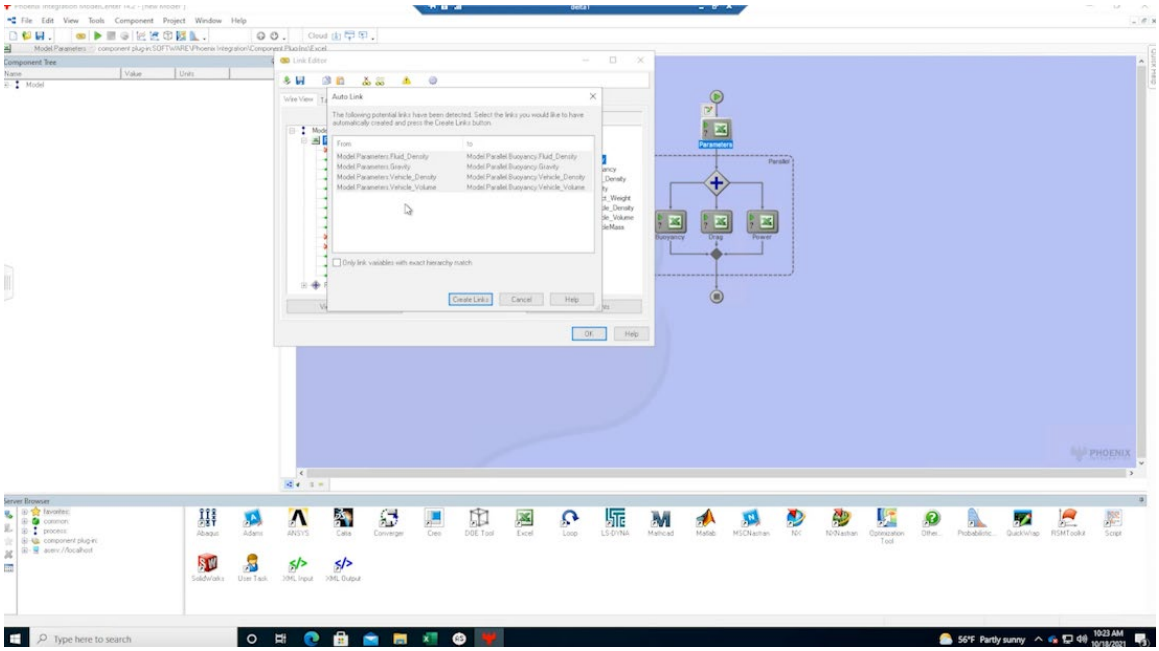
Then you repeat the sets mentioned above for the Power Excel. When the Power Excel is imported to Model Center the variable types by default are Double to the fact that they are not used in any other excel sheet. In this instance it is unusual because for all other Excel sheets that were imported the input variable type was changed to Strings. This has been consistent every time we have imported that Excel sheet, therefore, you don't have to worry about this one. Once again, we will double click on the text field named excel and label it Power. Therefore, the Model Center model will look like the screenshot below.



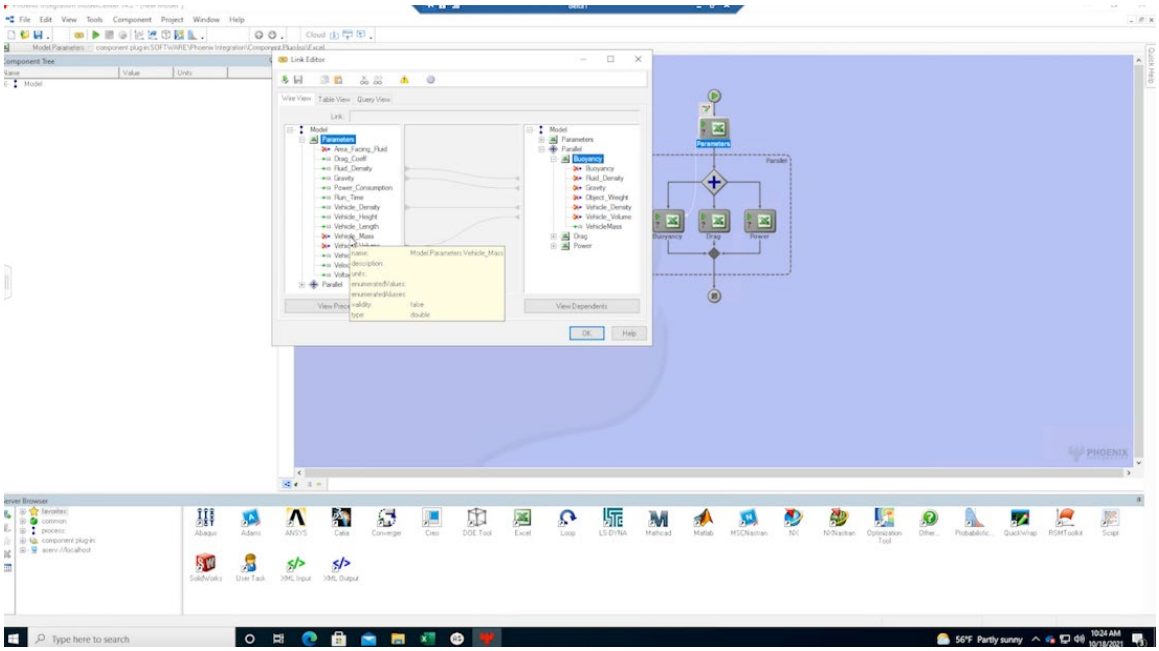
Now we are completed on our model as far as the inputs, outputs, and Excel sheets being in the model. The next step that we will show is how to link the model together. It is really simple you will have to click on the Parameters Excel icon in the model and then you will see the link icon which you will drag to the other Excel sheet icons.



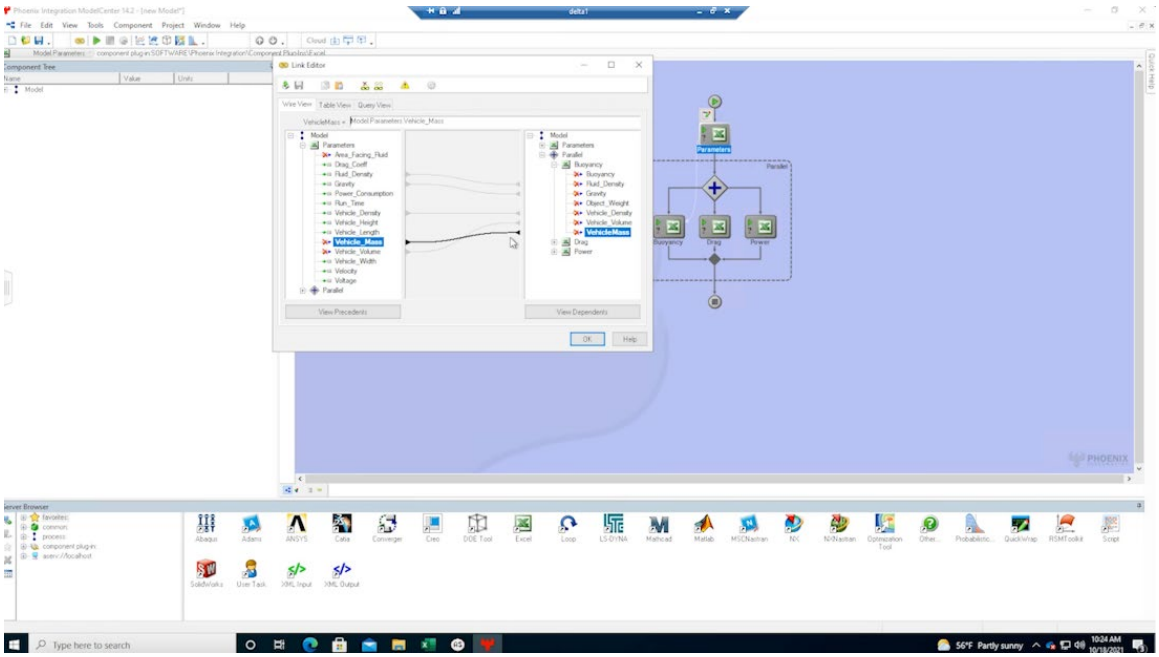
Once you drag and click on the Buoyancy Excel sheet icon for this instance, then the auto link popup window will appear. This is where the range names are important.



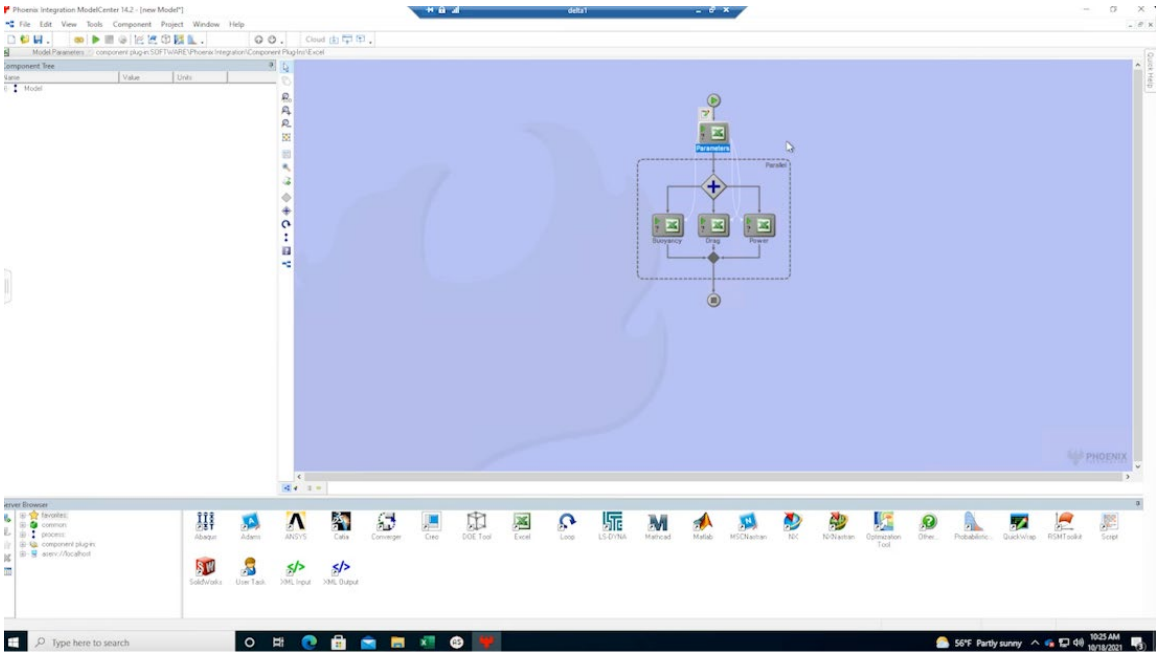
If you labeled the range names the same in the two different Excel sheets, then it will identify that each one has the same here and it will automatically create the link for you when you press the Create Link button. You will see that it created the links automatic with range names that are the same. Now the one thing that I mentioned earlier is that Vehicle Mass in the Parameters Excel sheet and the Vehicle Mass in the Buoyancy excel sheet are not the same. It is not a big deal if you made that mistake of not reusing the range names in multiple excel sheets.



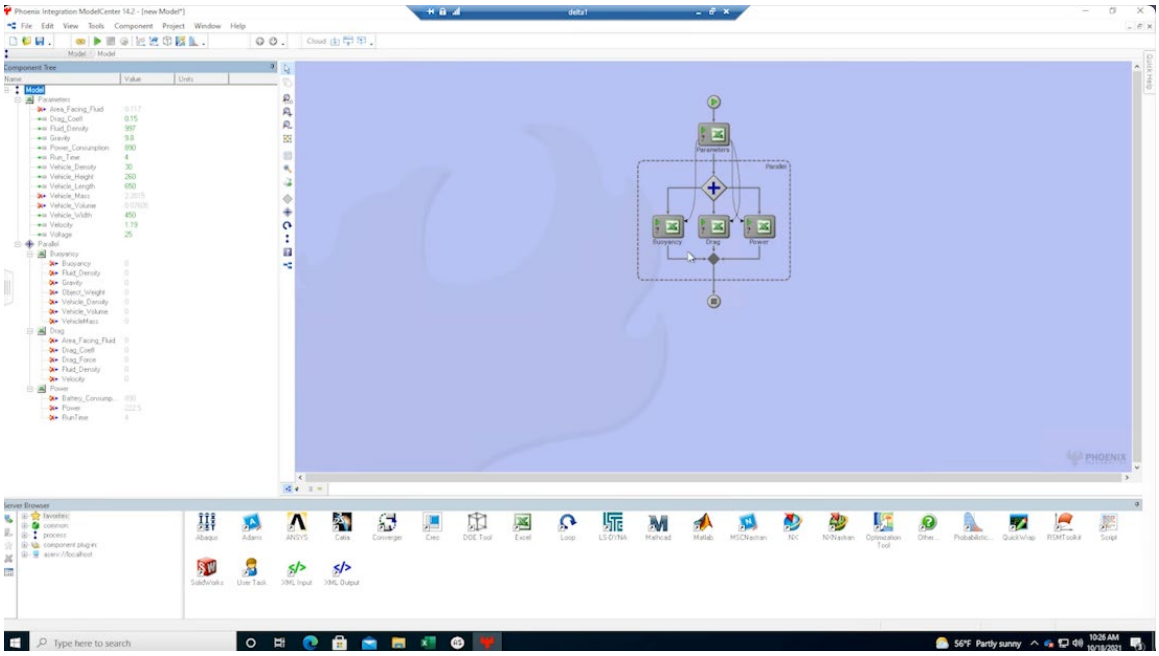
This is how you do this if you did not label them the same. You can click and drag the two labels that are not the same. In this case we will click and drag the Vehicle Mass in the Parameters excel sheet and the Vehicle Mass in the Buoyancy Excel sheet. This will create a manual connection link.



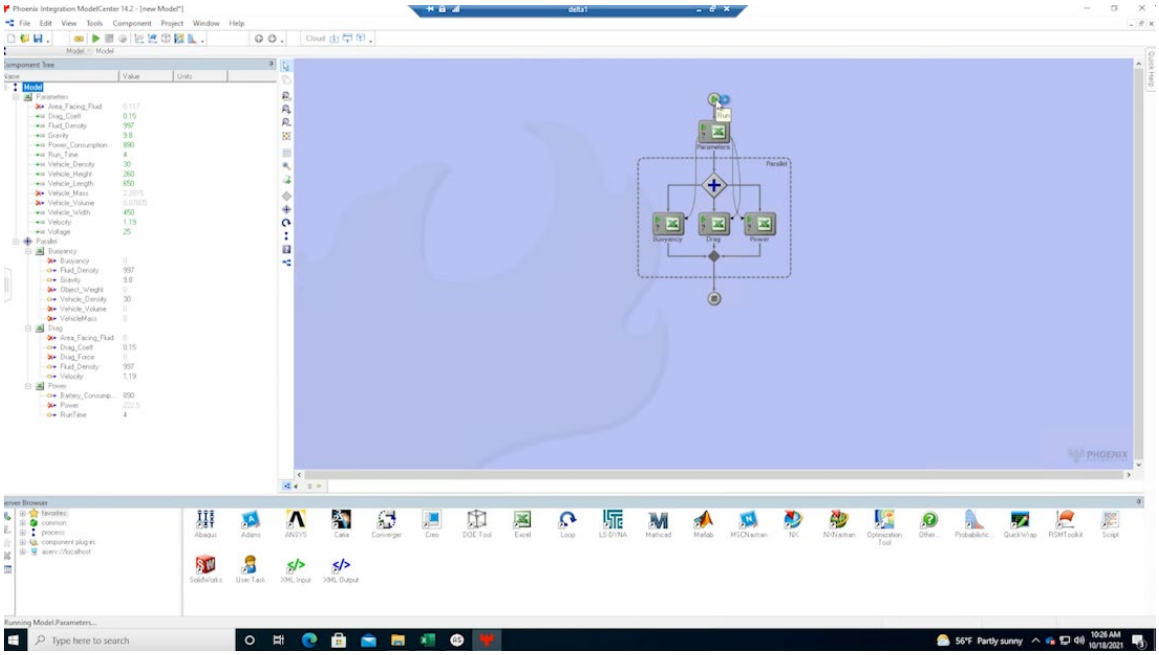
Therefore, if you didn't happen to name the same range name on each Excel sheet, that is fine and you don't have to worry about it. You will just have to manually link each occurrence of that. That's all you have to do on this one and now you have a linked Excel sheet. You will have to repeat the process for each Excel sheet in order to link them to the Parameters Excel sheet in Model Center. For Power Excel sheet we will need to manually link the Run Time to the Run Time of the Parameters Excel sheet. We then need to link the Battery Consumption in the Power Excel sheet to the Power Consumption in the Parameters Excel sheet. Now once everything is linked then the model will look like the screenshot below.



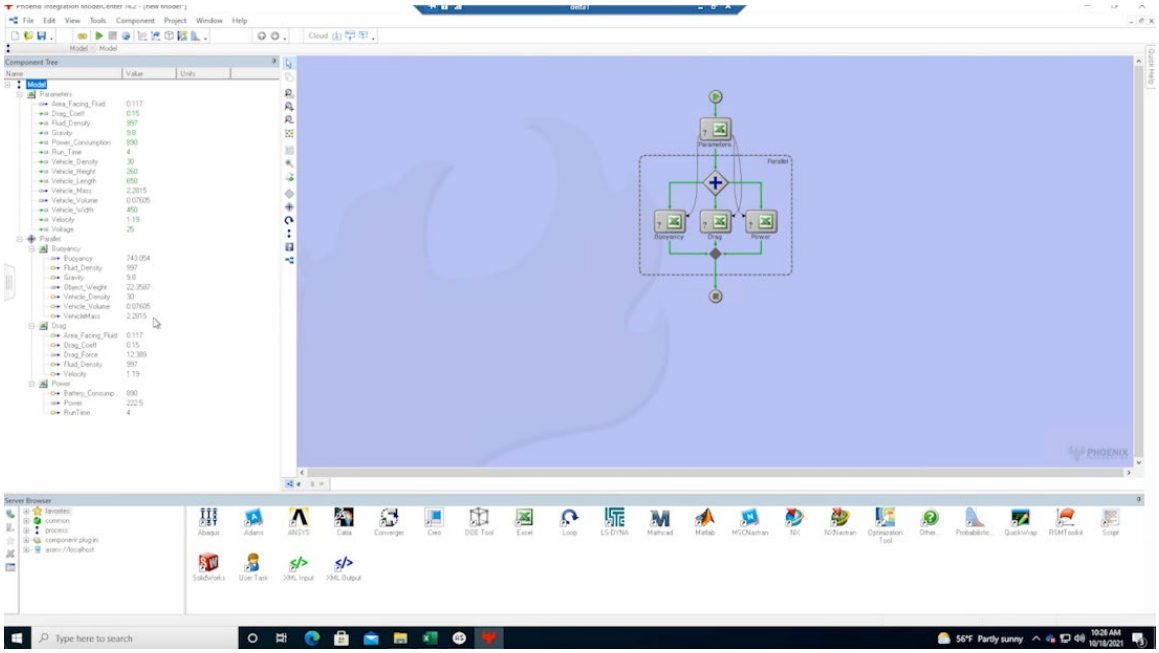
Now you have all three things done. So the next part you will head over to the left side menu and extend the tabs. This is where you have all your Parameters and all the parallel values that consist of Buoyancy, Drag, and Power.



Now you have all your stuff link together and now the next step will be to run the model by clicking the play button on top of the Parameters excel sheet.



Once you hit the play button then it will start going through the values. And as you can see on the left-hand side column each input variable that you had are all green arrows and had a value in there. And then you populated the link values which gives you the numbers.



You can go into the inputs and change the values and each time you change it and rerun the model, the calculations are updated with the newly input value. It grays out all of your outputs. Each time you change a variable you will need to rerun the model. So this is the tutorial for Model Center to run. You can then save it and save your file wherever you saved your Analyses file. And this is how you create this and on the next tutorial we will going through how to do the optimization of Design of Experiments where you can choose different input values that will find the best results. Thank you.

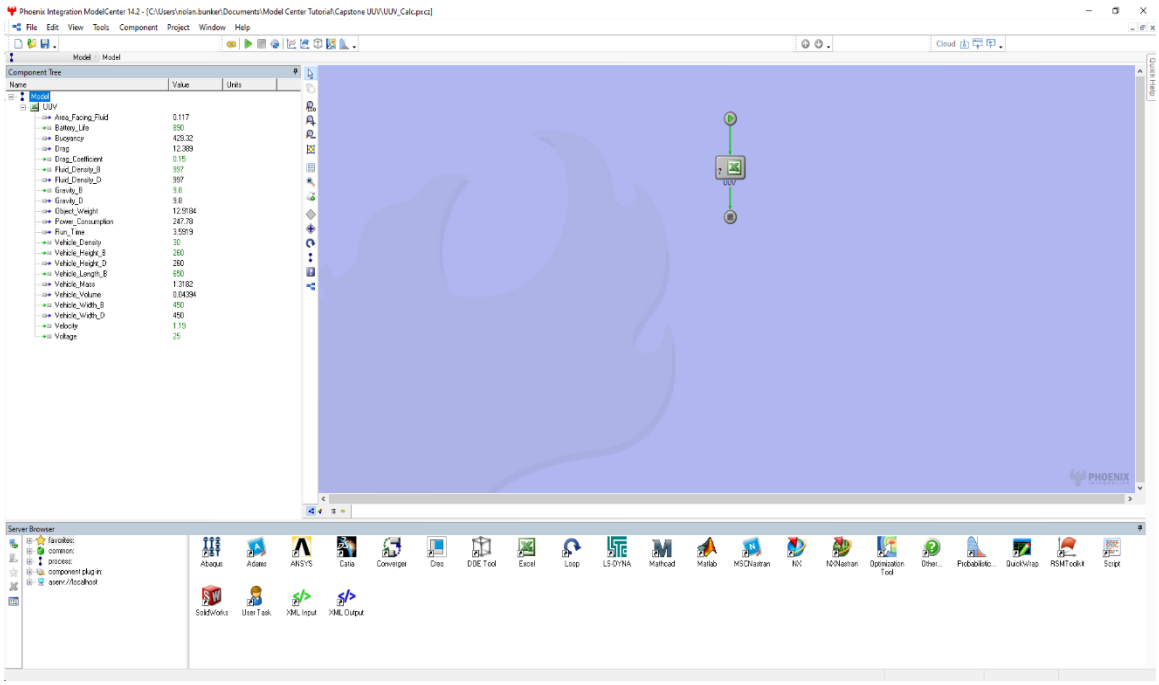
ModelCenter MBSE Tutorial Part 1

Background:

In this tutorial, we will go over how to incorporate the ModelCenter analysis and the Cameo model that has previously been created. This will be accomplished by the use of a Cameo tool called ModelCenter MBSE. This tutorial will give a detailed description of the steps required to set up a basic connection between ModelCenter and Cameo. More complex connections will be made in a later tutorial.

Step-By-Step Guide:

To begin, we are going to discuss a change to the ModelCenter Model that was created in the previous tutorials. While the model created is a good example of the connections and functionality that can be accomplished within ModelCenter, it was discovered that Cameo does not allow for some of those connections within Cameo itself. This error will be discussed in a later tutorial but is relevant now since the analyses had to be re-done for this tutorial.



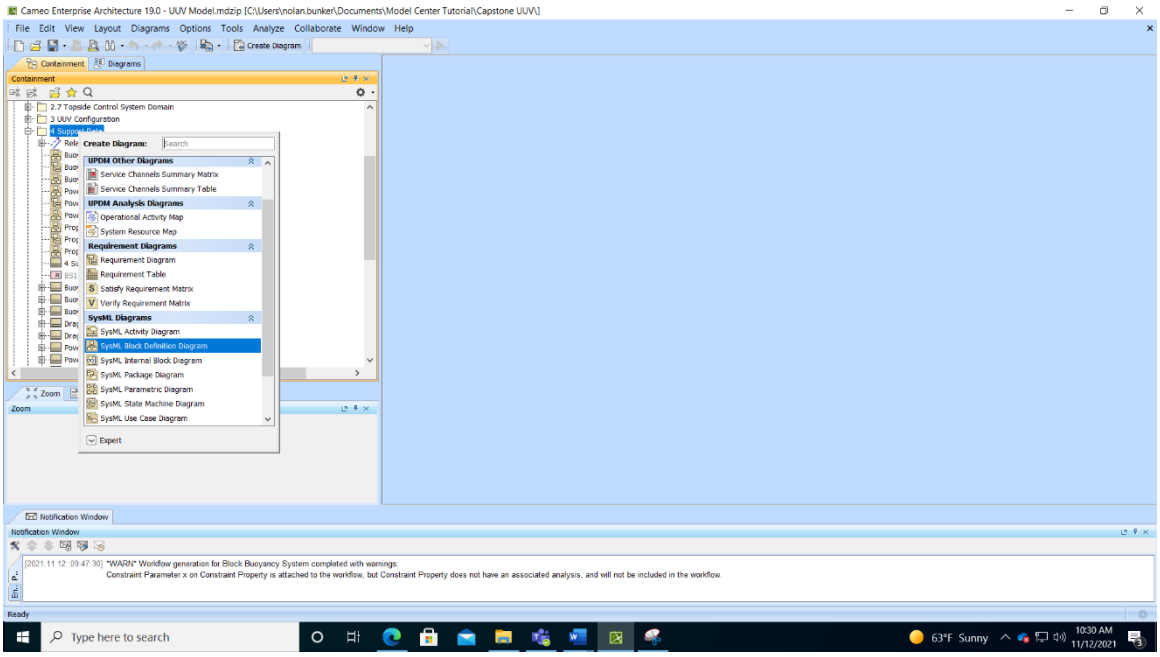
A quick look at the changes that were made to the ModelCenter analyses is shown above. Instead of having one ModelCenter analysis where one input Excel file feeds three output excel files, we combined all excel files into one single. To avoid the issue with a single input being linked to multiple outputs, we created an input for each output. For instance, Buoyancy and Drag both use vehicle width as an input variable, to ensure that the width changes for both later on during the optimization, we created Vehicle_Width_B and Vehicle_Width_D (width for buoyancy and drag respectively). We then made Vehicle_Width_D equal Vehicle_Width_B so that they will change together. The screen shot above shows the analysis for the model. This allowed for the integration into Cameo without causing errors. As mentioned, this will be important in a later tutorial. For this tutorial, we will continue with the integration into Cameo.

Now that the change in the ModelCenter Models has been covered, let's move into Cameo. Open up the Cameo model of your choice. For this tutorial we are going to use the model that was created in the Cameo model tutorials. To allow for easier tracking of the

concepts being applied, we are going to create a new package so that all of the values we need are located in one spot and we do not have to go search for them. If in your model you choose not to do this, that is ok as well.

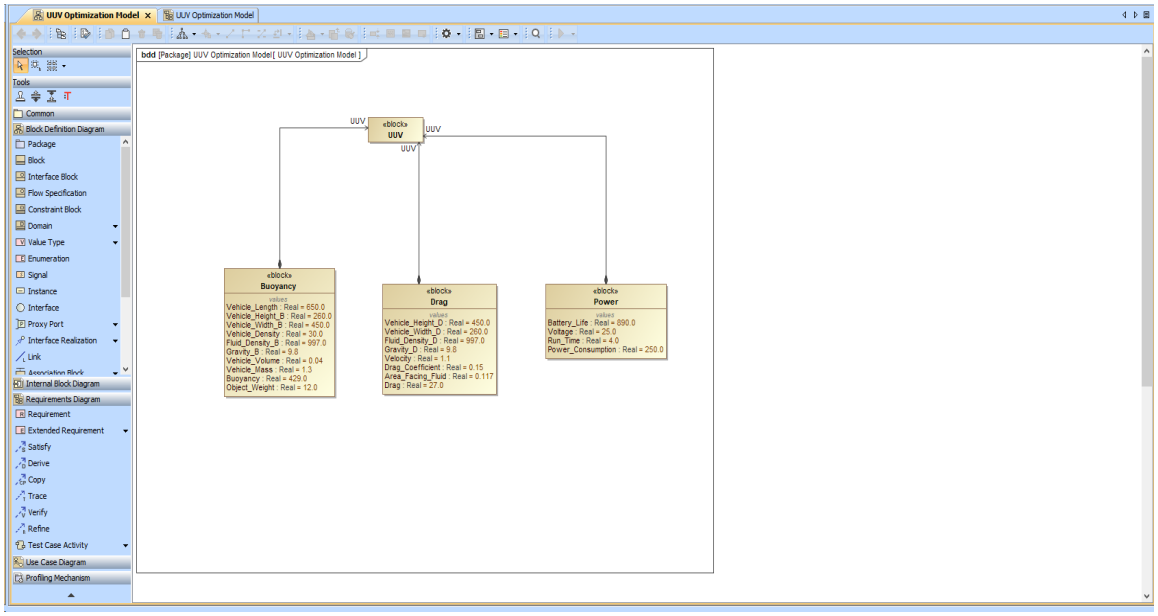
To begin, create a new package by right clicking on the Model > element > package. We are naming our package as 4 Support data. This package will be used to house all of the data, elements, and diagrams we are going to create in the next few tutorials.

Now that you have created a new package, create a Block Definition Diagram by right clicking on 4 Support Data > Diagram > SysML Diagrams > SySML Block Definition Diagrams.



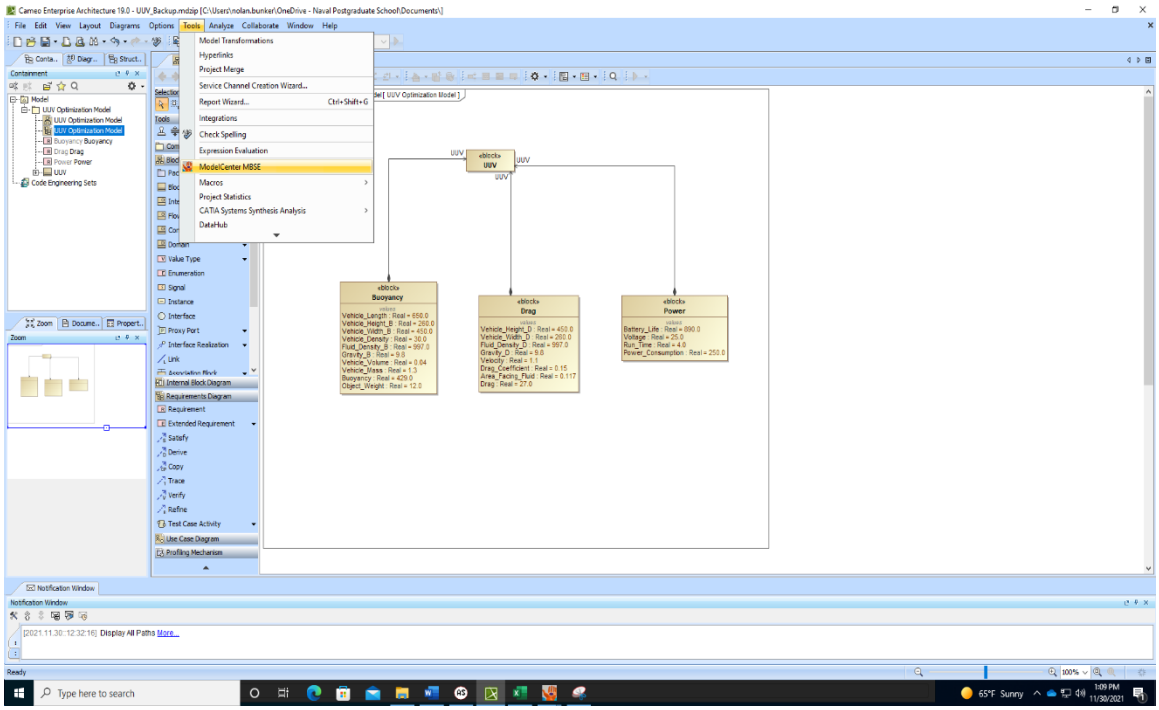
Inside the Block Definition Diagram, the user can create some blocks that allow them to input value properties. The user can click and drag the block icon into the model window. The first block was labeled as the Buoyancy. The other two blocks that were created were Buoyancy Drag and Power. The buoyancy block is where the input and output values can but entered in using value properties. The user can enter in the names they choose once they create the value property. It is assumed that the user knows how to create the value properties based on previous tutorials. The information can be entered directly

into the block when the value property is created. A format that can be used is as follows: Vehicle_Length: Real = 650. Using a format similar to this allows you to enter each value propety without having to go into the properties window for that value.

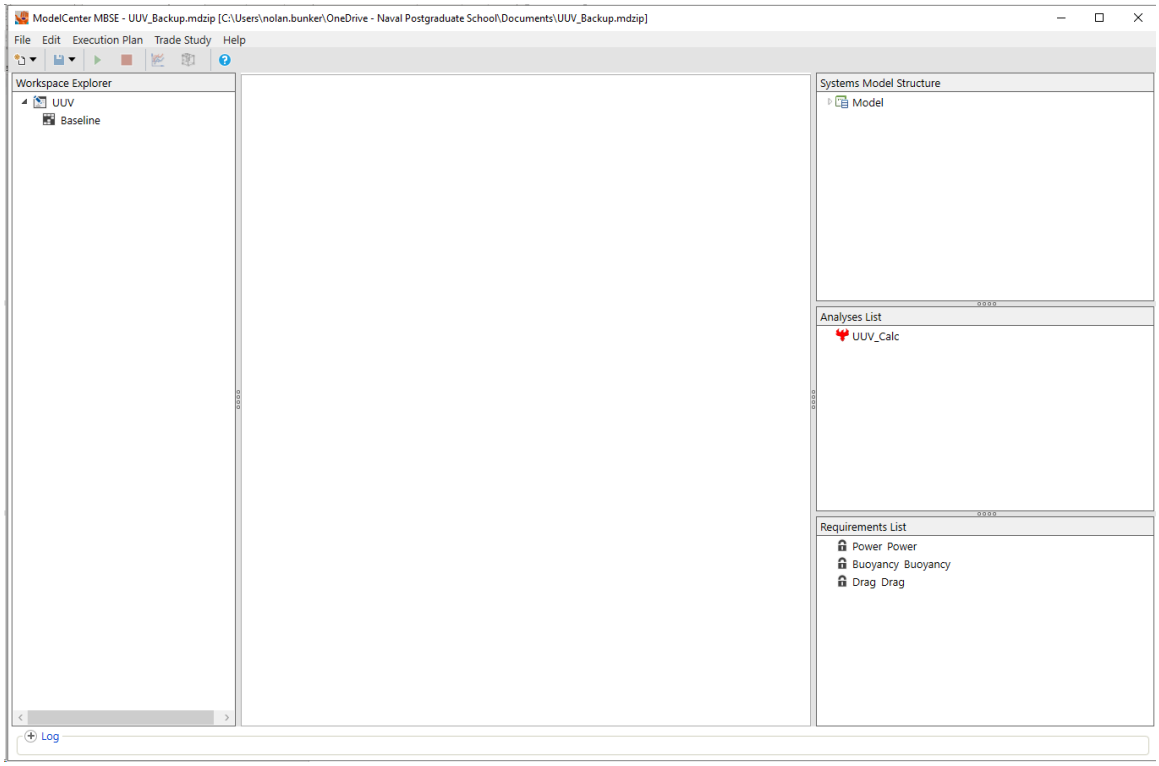


The values listed within each of the blocks should match the excel files that were created for the ModelCenter analysis, this will help with the linking of the two softwares later in this tutorial. As can be seen in the screenshot above, the Buoyancy block have the necessary input and output values. It should be of note that a default value must be entered in each value property entry otherwise ModelCenter MBSE will give an error when the simulation is executed.

Once the block definition diagram is created it is time to start up the ModelCenter MBSE tool. This can be accomplished by going to tools > ModelCenter MBSE in Cameo. The screenshot below shows the location.



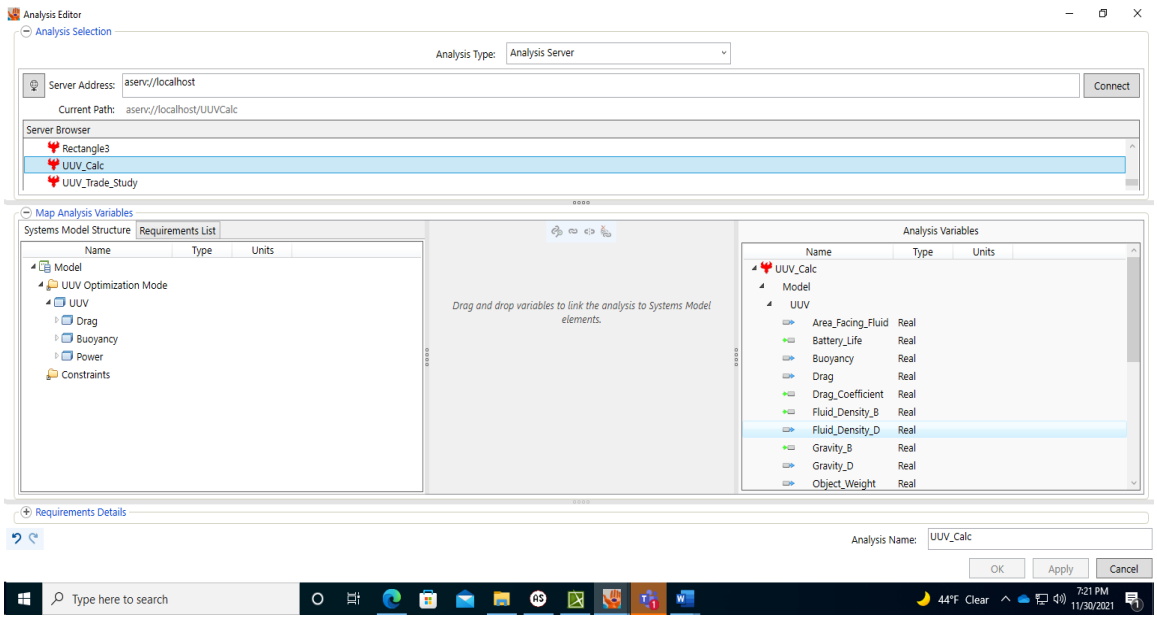
Open up ModelCenter MBSE and a new window will appear. This window will look something like the screenshot shown below; however, because our model has already been set up, we have some features already included. We will discuss these additional items in this tutorial and in future tutorials. For this tutorial, we will focus on the basic functionality. The window that will be seen by a first-time user will have an Execution plan 1 on the top left of the screen. The rest of the window should be blank.



If you do not already have an Execution Plan 1 you can create on by going to File > New Execution Plan. If the Execution Plan 1 is there, move to creating a new analysis. This process is similar to the execution plan except you select “new analysis,” File > New Analysis. When the new analysis option is selected, the window shown below will appear.

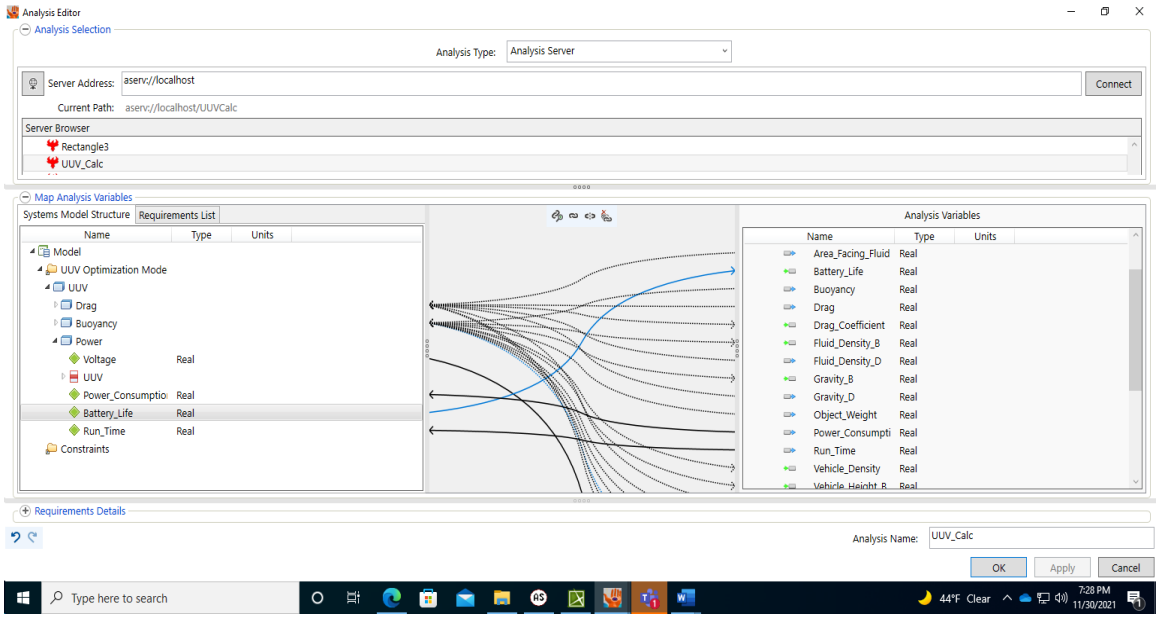
This is where it is important that the analysis server has already been activated as it will be used in this step specifically. In this window, the user will link the Model block definition diagram, more specifically the blocks and their value properties to the respective analysis. In order to be able to select the appropriate analysis, the user must select the “connect” button on the top right-hand side of the analysis editor window. Various options will appear in the Server Browser window. You will be able to tell if the analysis server is running if there is an error shown within this window when the connect button is selected.

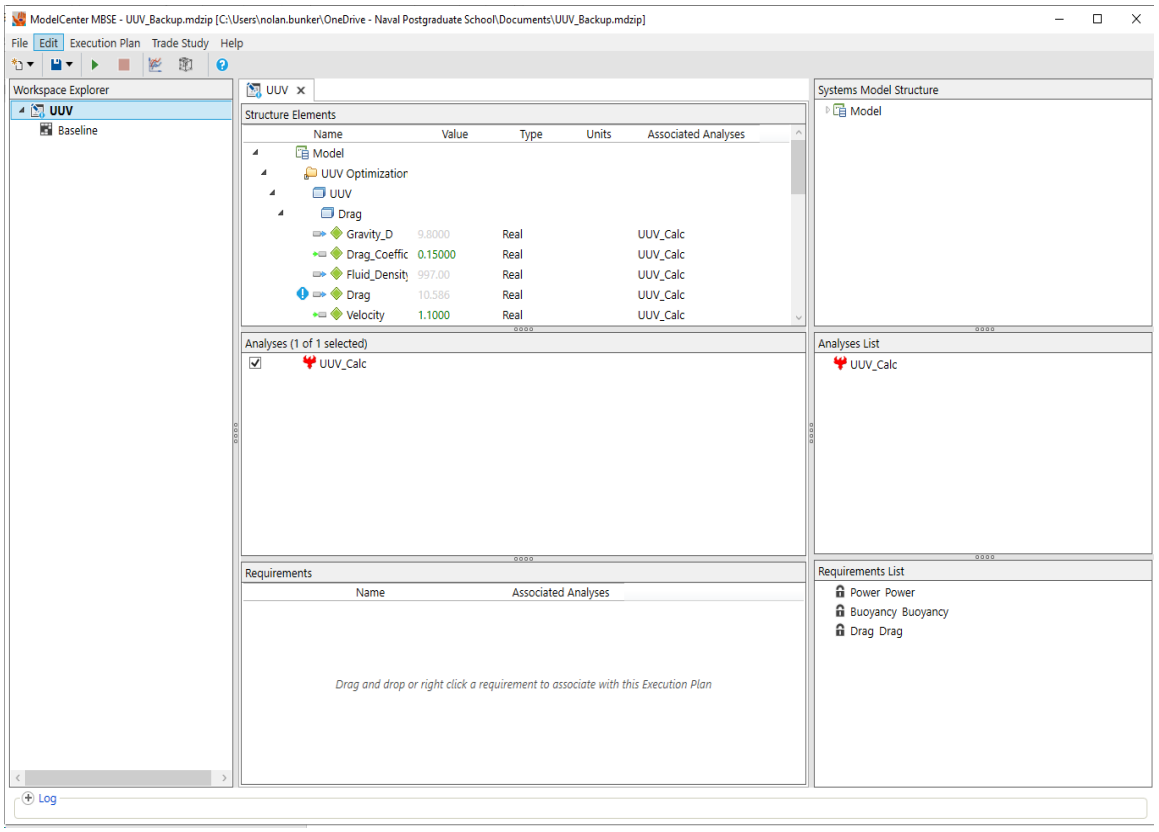
When the analysis server is active and the connect button is pushed, the server browser window will show various options, among these options will be the analyses that were created. The screenshot below shows what the window should look like



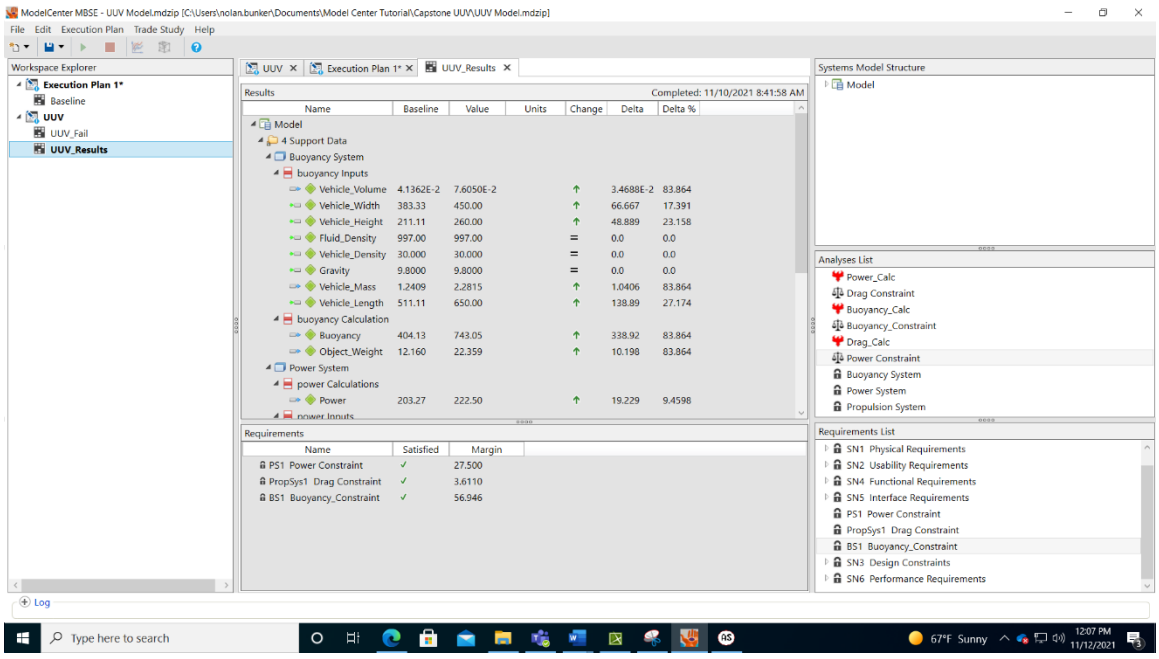
Select the appropriate analysis you wish to set up. In this instance, we are selecting the UUV_Calc analysis. Double-Click your option, if it does not appear in the analysis variables window on the bottom right initially, double click it again or until it appears in the analysis variable window. On the left-hand side of the window, you will see your model name. Click the drop-down arrow and find the location in which the block definition diagrams you are using are located. In this example, as we mentioned, we are using UUV Optimization Model. Click the drop-down arrow for UUV Optimization Model. You will see the Drag, Buoyancy, and Power Blocks. Similarly, to the ModelCenter model setup, begin linking the appropriate values between the model and the analysis. Ensure that all input variables are linked on the analysis variables side. This is why it is beneficial to create the blocks to reflect what the analysis shows. This allows for easy linking if all variables match the Excel file. Once all of the variables are linked, press apply and ok. Repeat this process for any remaining analyses that are required. For our example, this was done for both Drag and Power. Now that the analyses are linked to the model, they will appear in the Analyses List in the ModelCenter MBSE window. To run the execution plan, double

click on the execution plan. The window will change to what is shown in the screenshot below.





Prior to running the execution plan, the user must bring in the model and respective analyses into the appropriate windows. The model must be click and dragged into the structure elements window at the top and each analysis to be calculated must be clicked and dragged into the analyses window. As shown in the screenshot above, the analyses are represented by the ModelCenter Icon of a phoenix. The other icons in that window will be discussed during a later tutorial. Now that the model and analyses are within the execution plan, the use can click on the green arrow to run the plan. A loading circle will appear that represents the calculation of all the analyses. When the calculation is complete, a results window will appear. It should say “Results 1.” You can change the name of this results once you save it should you choose too. We named ours UUV Results. Each time you run the execution plan a new set of results will appear. We also have a UUV Fails results based on a scenario we built that will be discussed later. Shown below is an example of what the screen will look like once the results are finalized. Based on the input values you entered inside the Cameo model, an up arrow, down arrow, or equal sign will be shown on each value. This indicates whether the value in the analysis is greater, lower, or equal to the value you entered in the Cameo Model.



If you double click on the execution plan again, you will be able to scroll down towards the bottom of the structure elements window. Once you see green numbers, stop scrolling. These green numbers mean that the user can change the values that are used for the analysis. For example, instead of using 650 for vehicle length, we can put in 600 and re-run the calculation. An arrow will appear in the new results indicating the change in value. If the user is satisfied with the values provided by the results, they may save the values in two ways. The first way is saving the values as the baseline. By saving the values in the results as a baseline, you will change the values saved within the Cameo model blocks. Doing this on the first iteration is useful if you did not enter the correct values within the blocks during their creation. The other option you may choose from is to save as a design instance. Saving as a design instance does not change the values within the blocks inside Cameo. Instead, it saves your results as a "possible" outcome within Cameo for you to reference later. To do either of these saves, right click on the blue folders for the blocks. Once you have right clicked two options will appear, save as baseline or save as design instance. If you wish to save as baseline, click that option and a message will appear within the screen saying it was successful. If you choose to save as a design instance, click that option. A window will appear requesting the location you wish to save it at. It is recommended to open the drop-down menu and select the same location you are pulling the data from. In our example, we selected the 4 Support Data location. This allows us to keep all necessary data in a central location. Now that you have been able to link the analyses from ModelCenter into ModelCenter MBSE and Cameo, we end this tutorial here. The next tutorial will discuss how to create more complex connects and allow for the creation of parametric diagrams that are used for Trade Studies. Thank you.

ModelCenter MBSE Tutorial Part 2

Background:

In the last tutorial, we created the initial link between ModelCenter and ModelCenter MBSE within Cameo. This link is crucial to the more in depth and complex functions that can be conducted utilizing ModelCenter MBSE and Cameo. This tutorial will guide you through how to create parametric diagrams, constraint blocks, utilize the optimization tool within ModelCenter, and finally how to take the values from the optimization study and incorporate those values into the Cameo model.

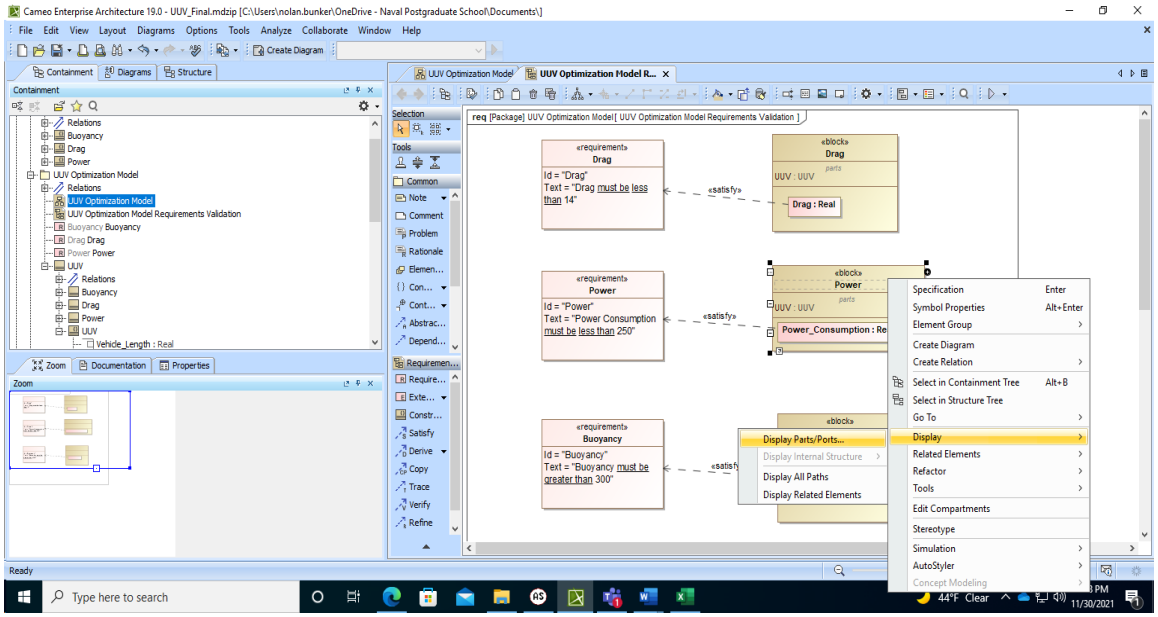
Step-By-Step Guide:

Let's begin! First off, this tutorial is assuming that the user has a basic understanding of how to use Cameo and create the necessary diagrams. This guide will show you how to create these diagrams anyways; however, the user needs to have a basic understanding as to their function and how they interact with other diagrams within Cameo. This tutorial is also assuming that based on you reading this tutorial you have completed the other ModelCenter tutorials that precede this one.

Picking up where we left off from the previous tutorial, you have just created the first link between ModelCenter and ModelCenter MBSE within Cameo. Moving forward, we will begin preparing our model for the more complex functions of ModelCenter MBSE. To begin, the user needs to create a requirements diagram. For this example, we created the the requirements diagram with all three constraints we are going to create and called it UUV Optimization Model Requirements Validation. The screenshot below shows this diagram.

The requirements we are going to create are: the Buoyancy must be greater than 300, Drag must be less than 14, and Power Consumption must be less than 250. The wording for this needs to be in the phrases like "greater than or equal to," "less than or

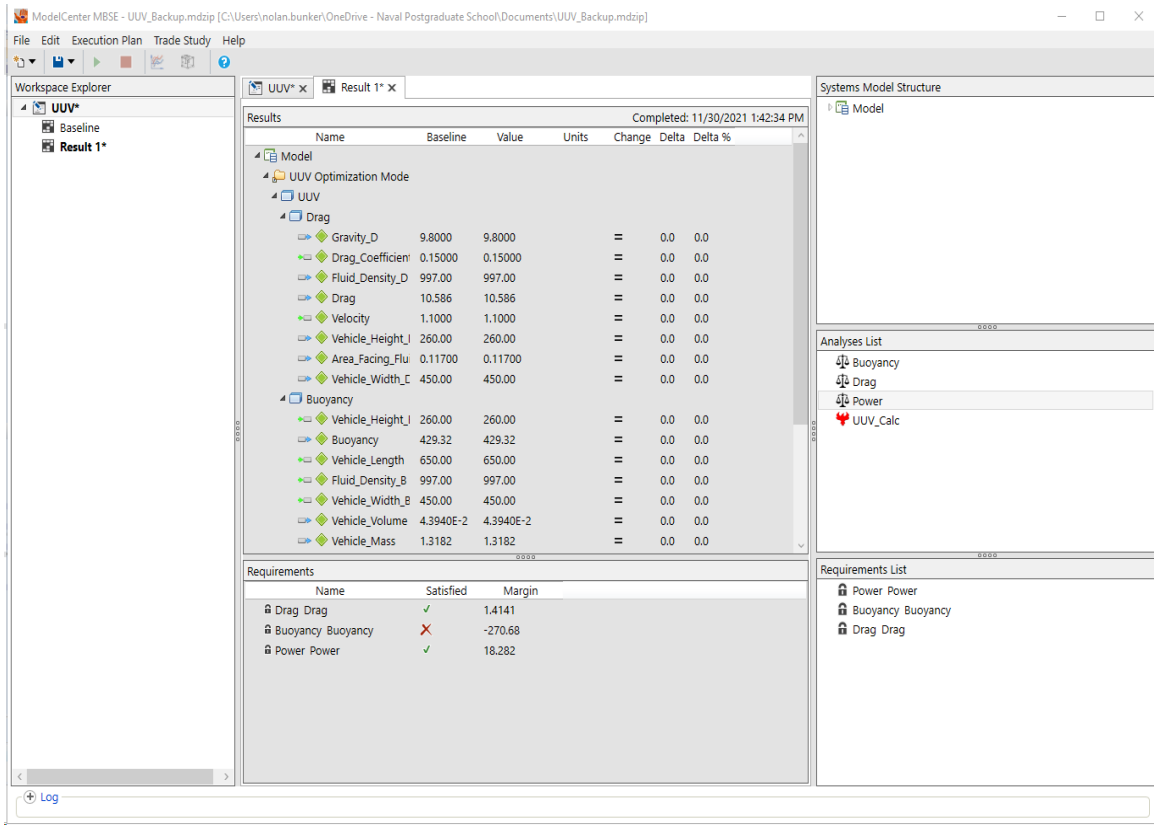
equal to," "greater than," "less than," "equal to," etc. This is critical to when you extract a constraint from the requirement. Repeat this step for Drag and Power, creating a requirement for each of these in their own requirements diagram.



To replicate this diagram, we created a requirements diagram and created three requirements within the model window. We named each of the requirements separately just to be able to track which one is which better. We then clicked and dragged in the buoyancy, drag, and power blocks into the model window. Now right click on the buoyancy block, go to display > display ports > left click.

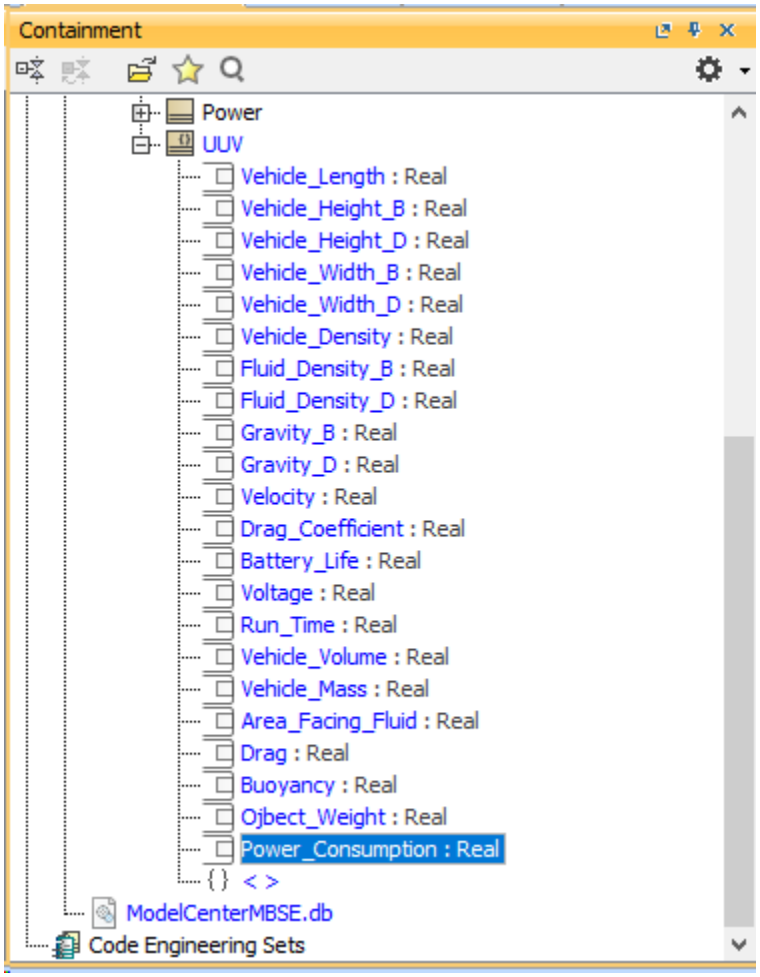
Once you left click it will bring up a popup window. Within this popup window will be the value properties that were previously entered in the block. For our example we will only select the value properties related to these requirements, e.g., buoyancy, drag, and power consumption. Click ok. Repeat this for each requirement. A small box with each value property will appear in their respective blocks. In the Selection pane between the containment tree and the model window, select the Satisfy option. Click on the buoyancy box and drag it to the constraint. If this is done correctly, the satisfy line will appear like shown in the screenshot above. Repeat this for the other two requirements. Once the satisfy link is created, you will need to go to Options > Project > General > General and make sure

that both “Use Requirements Term Glossary” and “Use Glossary” are checked as true. Next right click on the constraint and select tools > extract constraint block from requirement. Do this for Drag and Power as well. At this point, you can now go back into ModelCenter MBSE. Go to file and select import requirements. The three requirements you created will appear in the analyses list. These requirements will also appear in the requirements list. Click and drag each of the new requirements into the requirements window and the bottom middle of the screen. This will automatically put the requirements in the analyses list into the analyses window as well. If you run this execution plan, the values will check to see if your current numbers are successful in meeting your requirements. Shown in the screenshot below is our failed results in which the values did not meet the requirements.

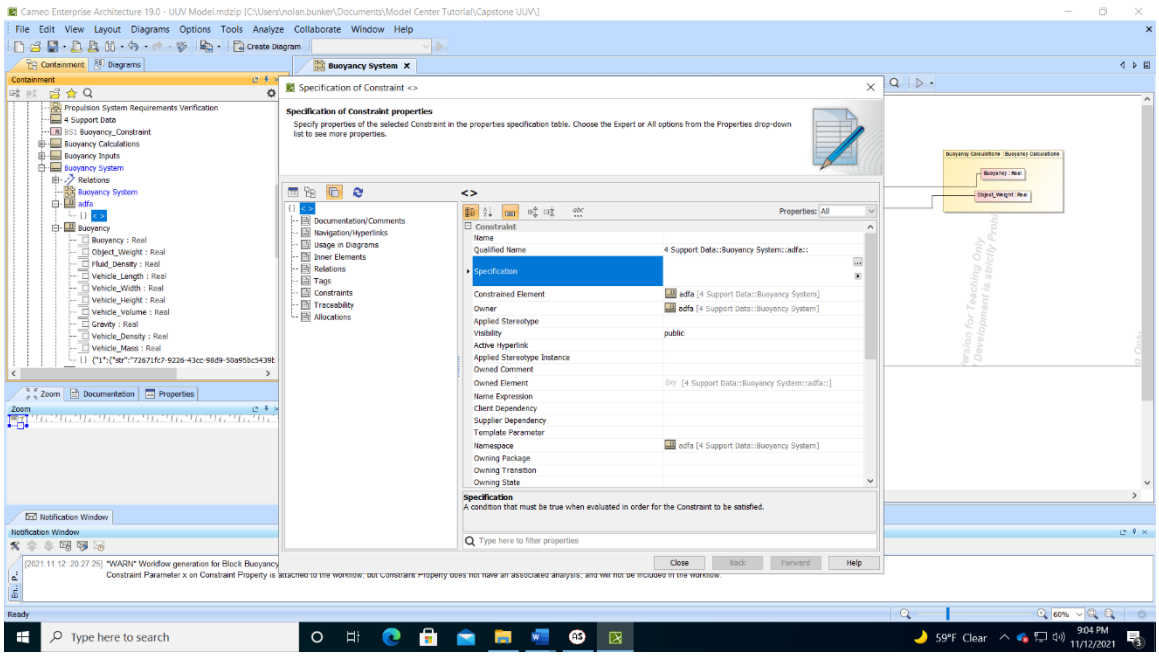


The goal now is to create a trade study to be able to optimize the design so that it will meet the desired requirements. You can do this by creating parametric diagrams and specification constraints. This process will be described in the coming paragraphs and screenshots.

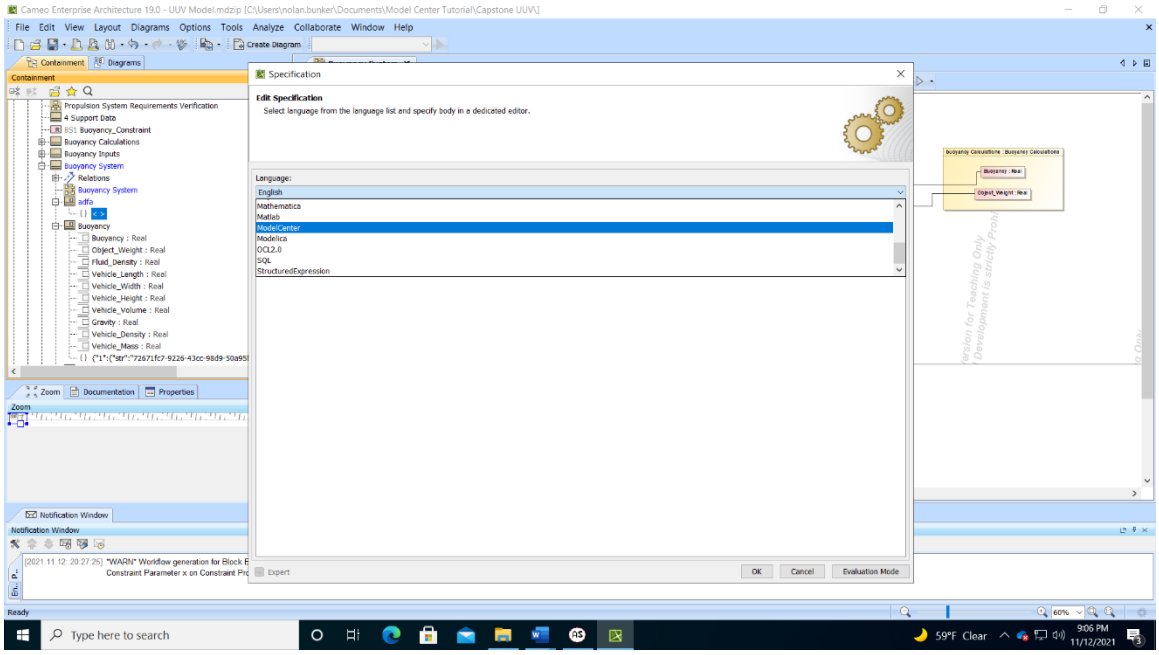
The first step for creating a parametric diagram is that you must create a new constraint block under the main block in this package. For example, we created a “UUV” constraint block under the UUV block. See the screenshot below. Once you create the constraint block you will need to enter in constraint parameters, essentially all of your input and output variables in your model will be entered here. The screen shot below shows a completed version of this step.



At the very bottom there will be an open slot that looks like this {} <>. Right click on this space and select specification. A window will appear that looks like the screenshot below.

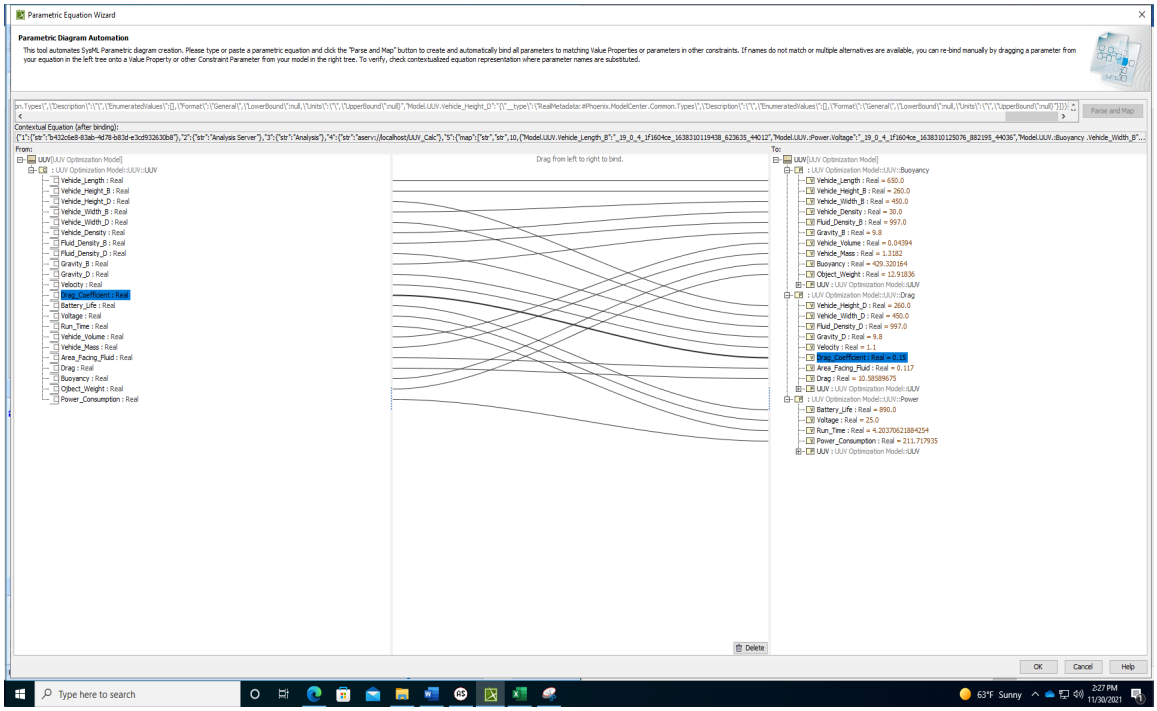


Within this window, select the three dots to the right under the specification option. This will bring up a window similar to that shown in the screenshot below.

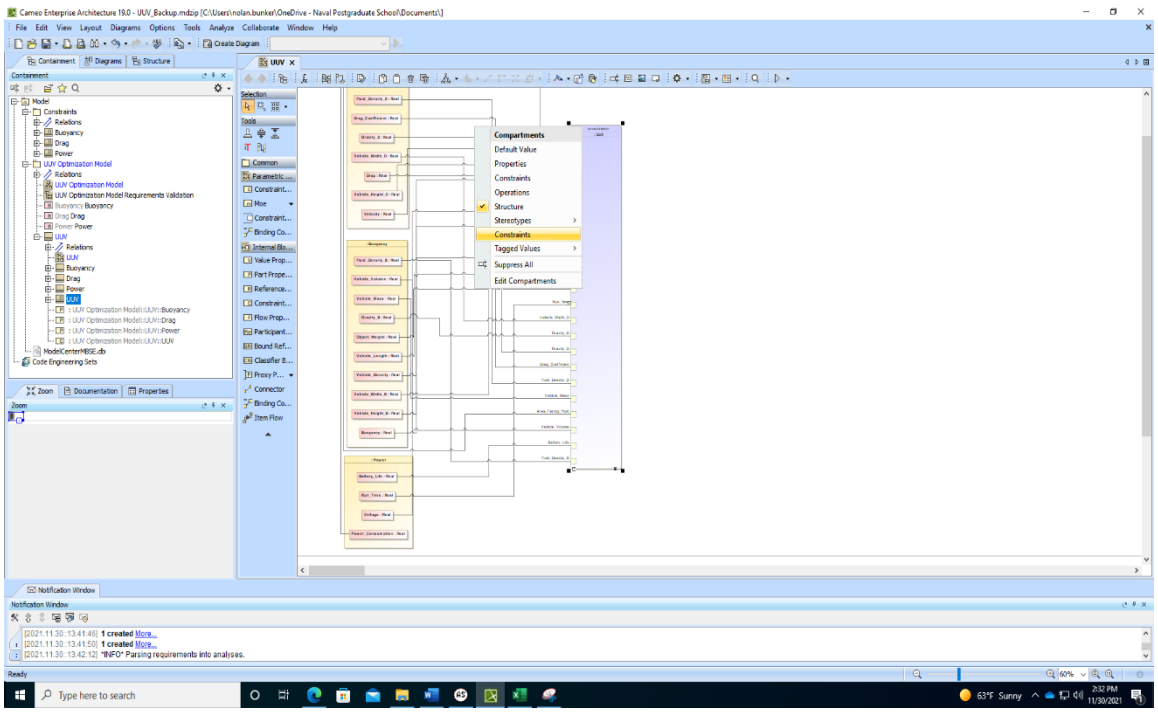


Click on the drop-down menu and search for ModelCenter. Select this option and click on the “launch analysis editor” button. The window where you link the model to the analysis will once again appear. Begin linking the input and output variables together like was done before. Click apply and ok. A new specification link is created. Once you have completed this step the {} < > space should now have an entry with a lot of different text

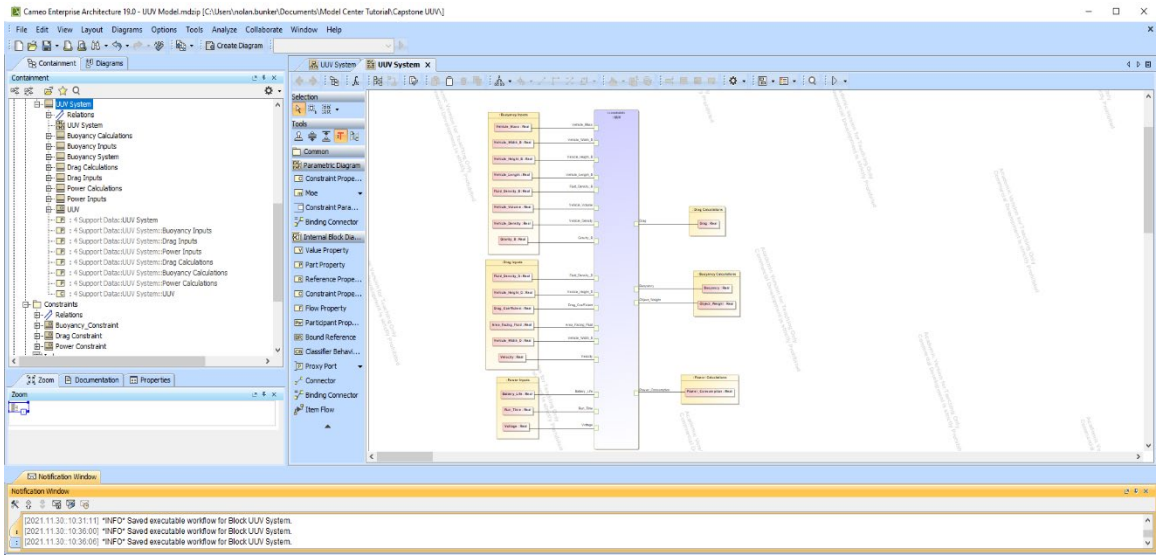
Once all the constraint parameters are created and the constraint specification is complete, right click on the UUV block and create a parametric diagram under the UUV block similar to how you created the constraint block. A blank modeling window will appear. Click and drag the three blocks into the window. You may be prompted to add in value properties, you do not need to do it at this time. You will now move the UUV constraint block into the modeling space. A window will appear that is similar to the screenshot shown below. Delete all existing connections and reconnect each parameter. Ensure that you connect the constraint parameters to the value parameters from the actual blocks.



Once all values are attached, click ok. It will generate yellow boxes (input/output blocks) and a purples box (constraint block). The purple block will be really large due to the text that is inside the block. To fix this, click only on the purple box, a small icon with three dots will appear in the upper left-hand side. Click on this and a smaller box will appear. Click on the down arrow and you will see a highlighted option for constraints. Uncheck this option. This is shown in the screenshot below.



You can then select the select the “quick diagram layout” button and perform some bookkeeping. This will result in the image show below.



The final step to ensuring that this model will work during optimization is that you need to create an executable workflow. To create a ModelCenter Executable Workflow

right click on the UUV block and select “Create ModelCenter Executable Workflow.” If everything is connected correctly, you will get a workflow was successful message. Save your model.

ModelCenter MBSE Tutorial Part 3

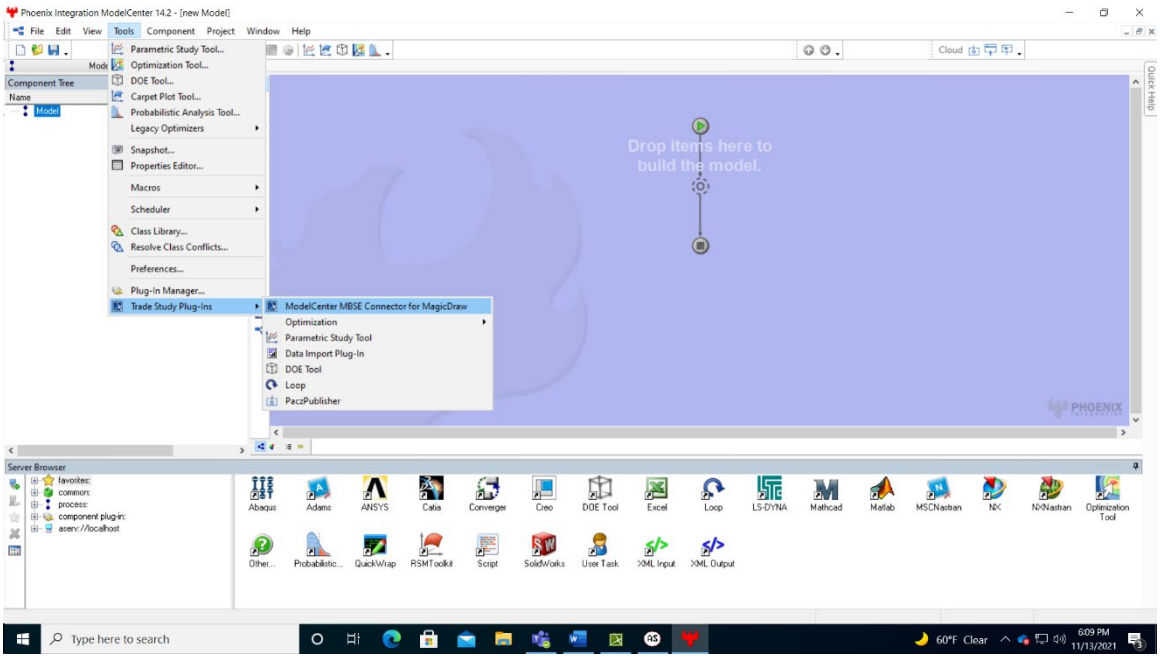
Background:

This tutorial is the final piece for the integration of ModelCenter and Cameo. In this tutorial we will show you how to import the executable workflow into ModelCenter. This will allow you to run an optimization study based on which outputs you wish to solve for using the inputs and constraints you created in Cameo.

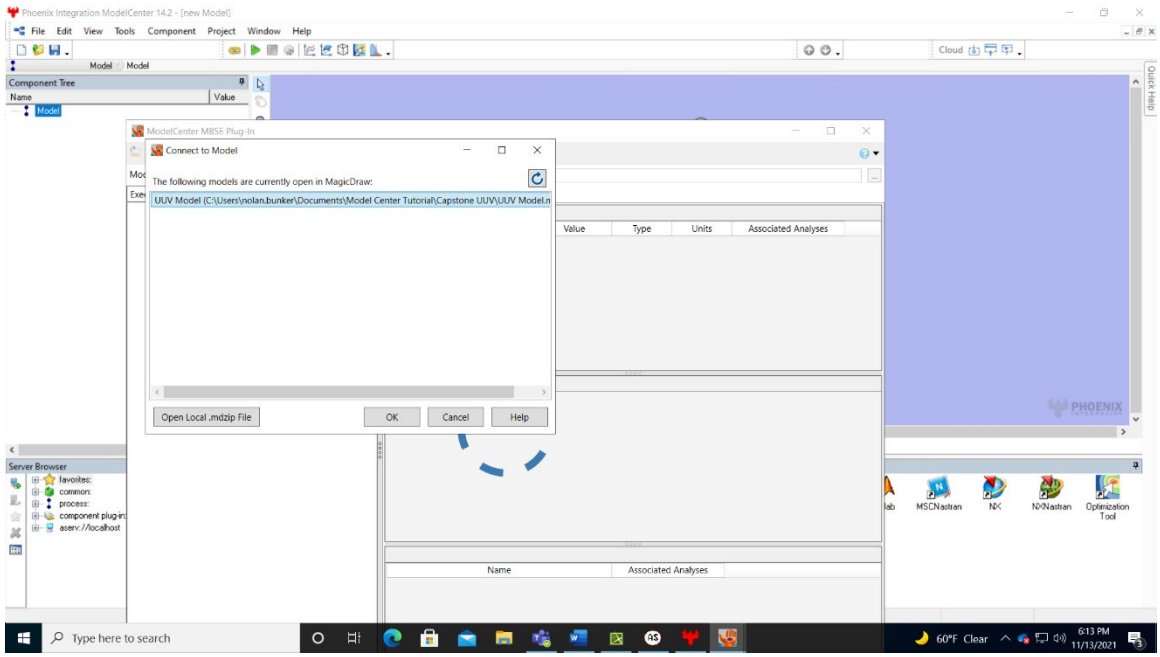
Step-By-Step Guide:

This tutorial should only be done if the other ModelCenter tutorials have been completed. This will ensure that your model is set up correctly and will function correctly when moving to the next few steps. This tutorial is going to show the user how to set up an optimization trade study so that they can get the optimal design inputs based on the outputs. This will be useful for when the user needs to tailor their design to meet certain criteria and will show what changes need to be made to meet that.

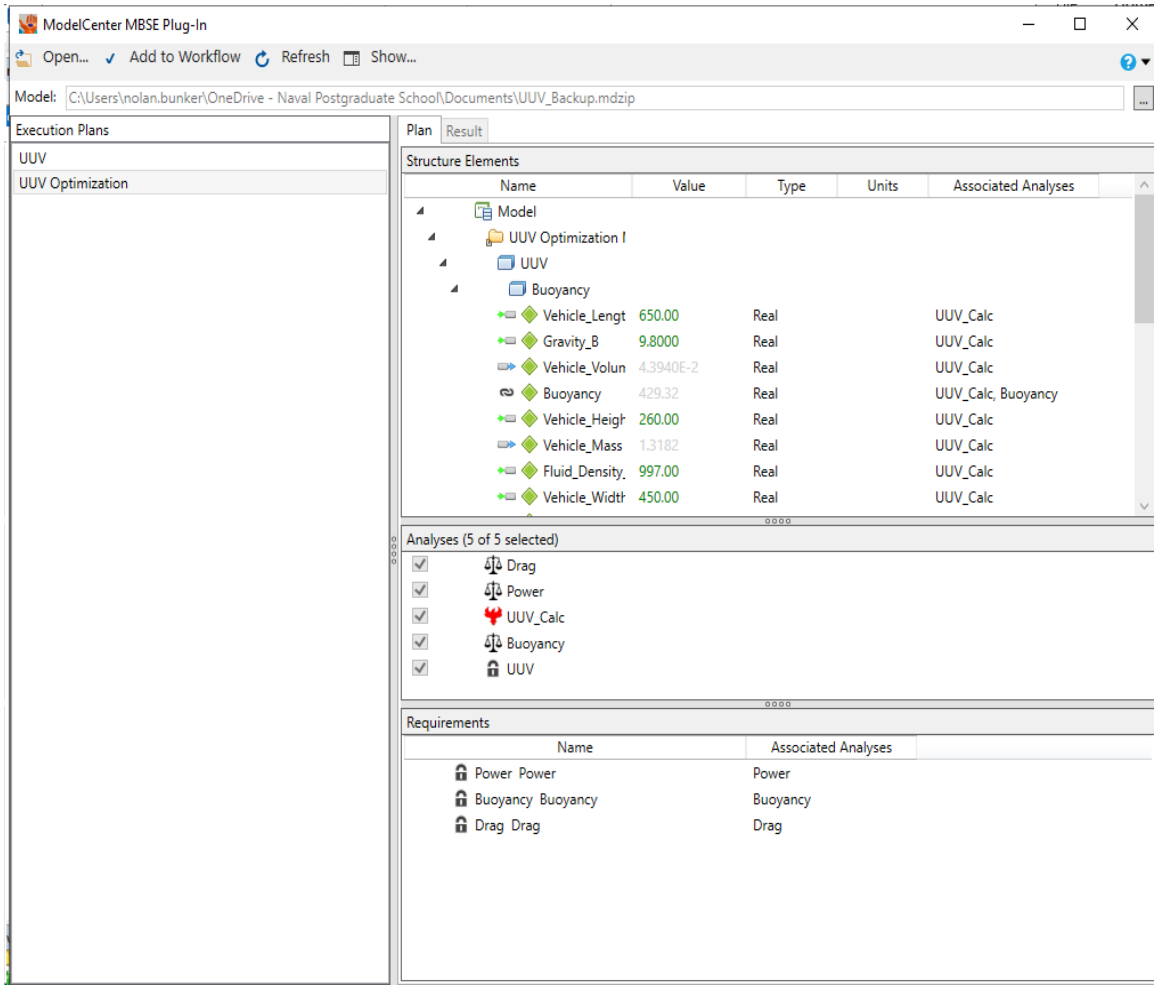
To begin, ensure that the analysis server is running. Once the analysis server is activated, open the ModelCenter application. Once the application is open, select “Start a New Model” and then select “Process.” Once the model opens up go to Tools > Trade Study Plug-Ins > ModelCenter MBSE Connector for MagicDraw. This is shown in the screenshot below.



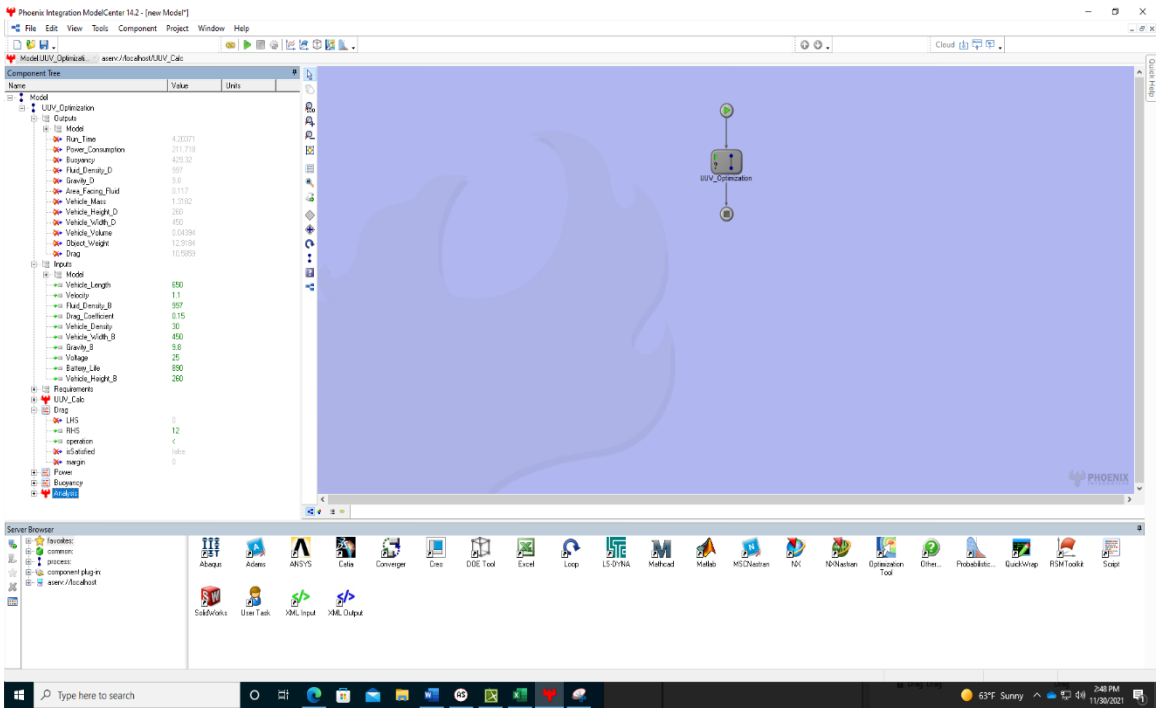
After you select the ModelCenter MBSE Connector for MagicDraw a window will appear. The window is shown in the screenshot below. Select the option that is shown within the window and press ok.



The ModelCenter Plug-in will appear, select the model you created. In our scenario, we will be choosing the file that says “UUV.” Double click on your selected w file and it will the windows will populate. Check to ensure that all the necessary information is there, e.g., inputs, outputs, constraints, and analyses.



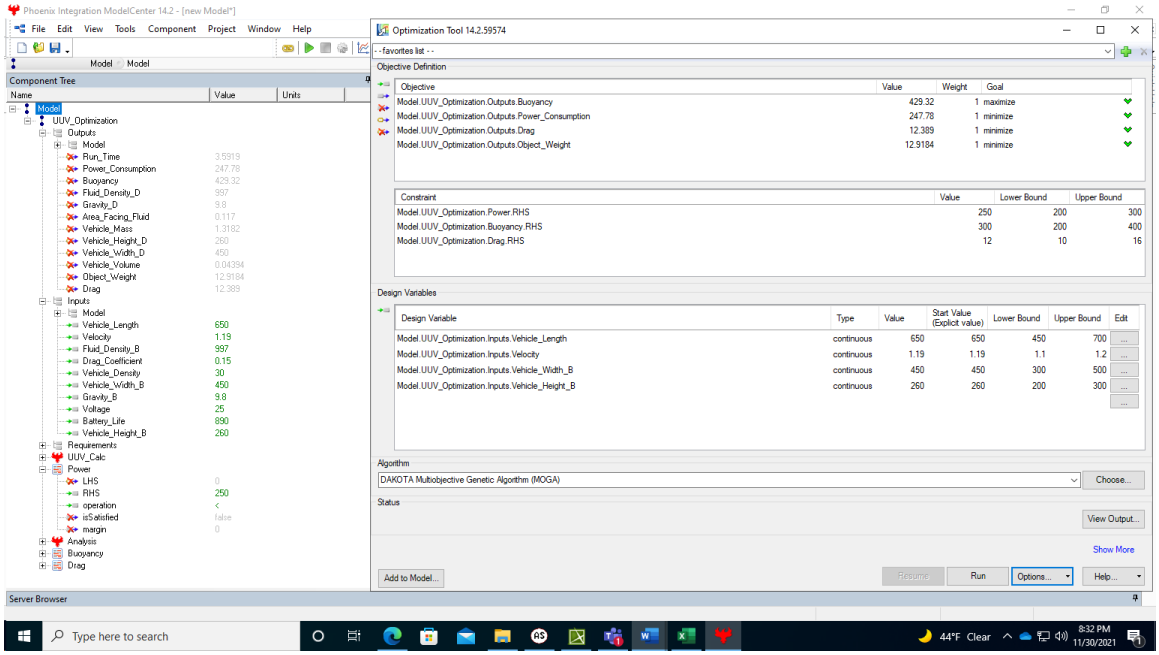
Click on Add to workflow. The model will begin to import into ModelCenter. Wait for the spinning circle to finish. Once it says “Update Workflow” close out of this window. You will now see a box within the ModelCenter build window, shown in the screenshot below. As you can see, this workflow now shows the inputs, outputs, requirements, and analyses in the left-hand side window.



You will now see a box within the ModelCenter build window, shown in the screenshot above. As you can see, this workflow now shows the inputs, outputs, requirements, and analyses in the left-hand side window.

To begin the optimization process. Go to Tools > optimization tool. The window shown in the screenshot below will appear. To add in the design objectives, click on the model plus button until you can get to outputs. Add in the outputs desired. For our example, we are concerned about buoyancy, object weight, drag, and power. These four are added to the objectives window. The same process is done for the constraints side; however, the user can open up the specific constraint. For our example, the buoyance constraint, power constraint, and drag constraint are used. When putting the constraints in the window, the value selected must be the one from the RHS. Finally, the input variables that the user wants to change in order to see what the best design will need to be selected. These values

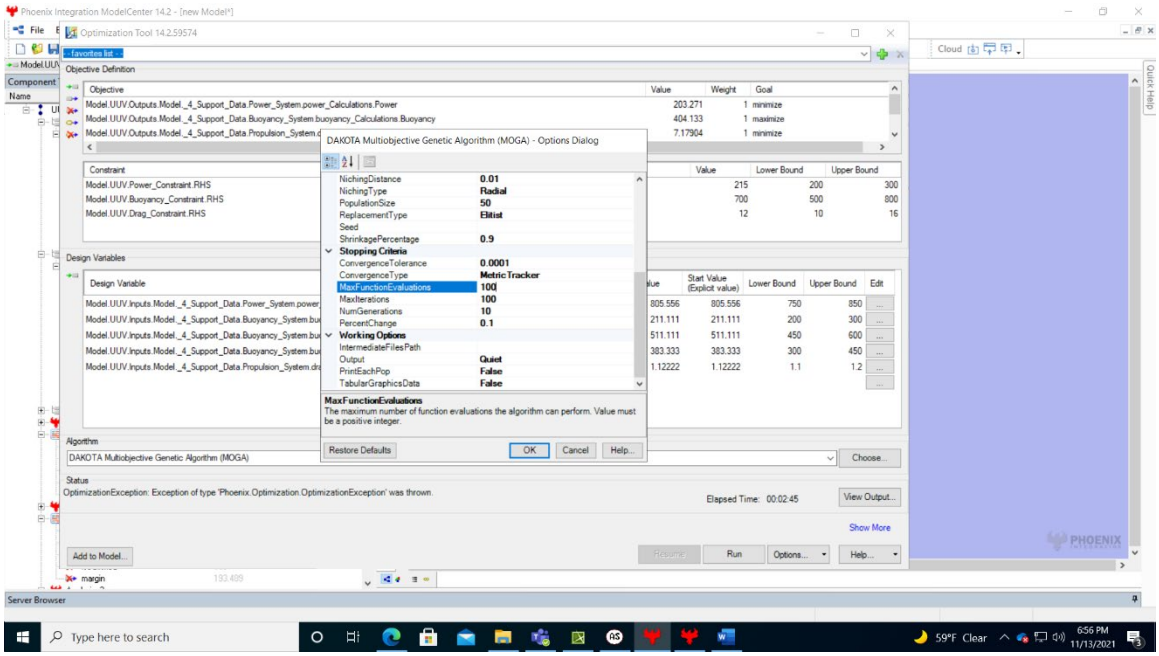
can be dragged and dropped from the inputs drop downs. Each variable that is going to be adjust needs to be added. The screenshot below shows and example of each of these steps.



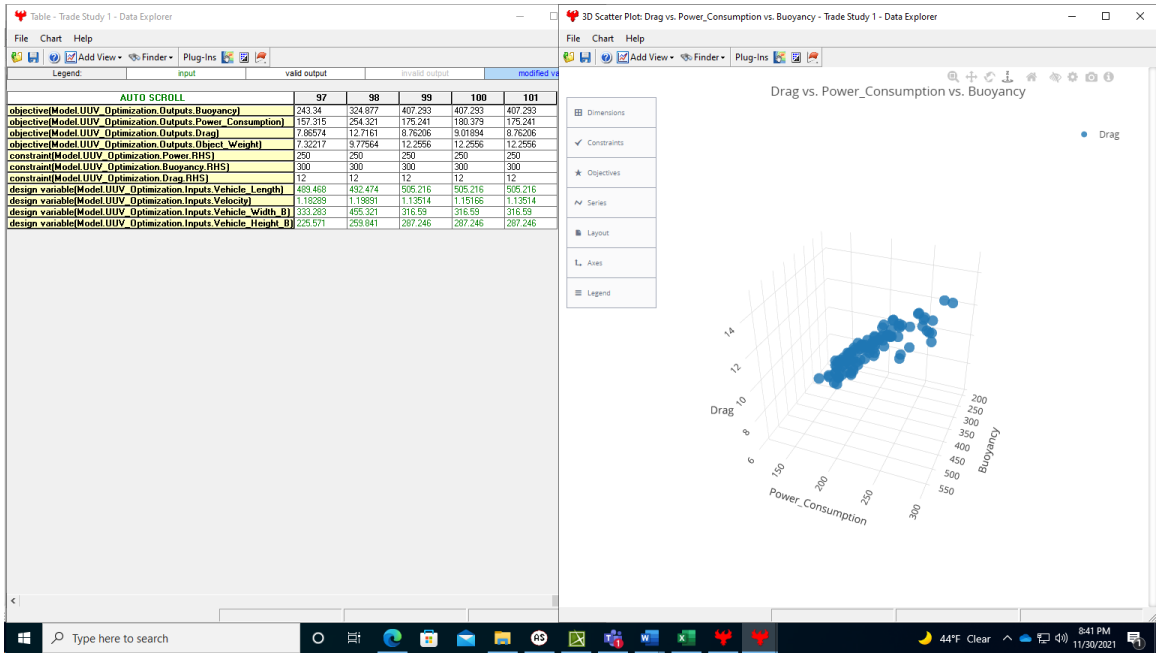
Now that the necessary values are in their respective windows, the user will need to select if they want to maximize or minimize the objectives. The buoyancy objective will be maximized, all other objectives are minimized. In the constraints window, you will notice that the lower and upper bound cells have alerts. The user must input a lower bound and upper bound for each constraint. For our scenario, the power constraint had 200 and 300 for lower and upper, respectively. For buoyancy, 200 and 400. Drag, 10 and 16. In the design variables window, the user can now do the same upper and lower bounds for each design variable. Show below are the values:

Vehicle Height (Buoyancy Input): 200/300
Vehicle Length (Buoyancy Input): 450/700
Vehicle Width (Buoyancy Input): 300/500
Velocity: 1.1/1.2

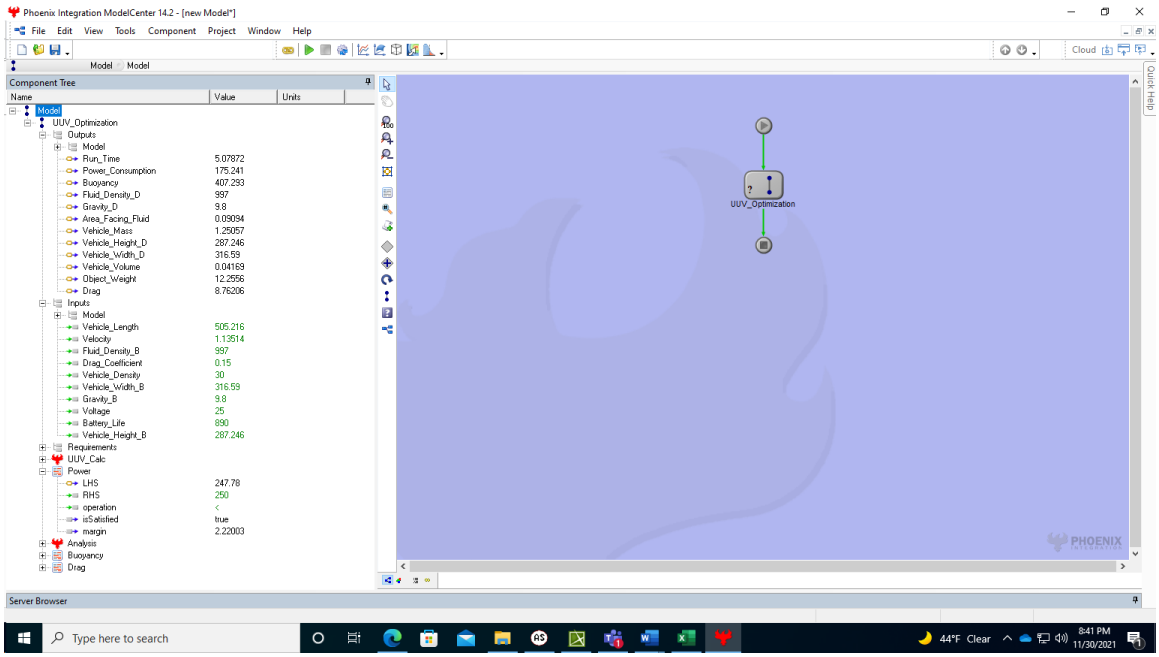
Once all the lower and upper bounds have been entered, the user will need to choose the solving algorithm. This can be done under the Algorithm section, clicking on the Select an Algorithm drop down, DAKOTA Coliny Direct was recommended for single objective simulations; however, for multi-objective simulations we decided upon DAKOTA Multiobjective Genetic Algorithm. This is not to say that any of the other 53 algorithms cannot be used, we just did not explore these other options. Click on the DAKOTA Multiobjective Genetic Algorithm (MOGA). Click on the "Options" drop down arrow, select Algorithm Options. With this window that has popped up, go to MaxFunctionEvaluations and change it to 100. Click OK. This is shown in the screenshot below.



You may now click "Run." A scatter plot will begin to populate on the screen. These dots are all the different results in an attempt to determine the best design based on what you entered. The calculation may take several minutes. Be patient and once the results are done you will have the optimal design for your system. The scatter plot and tally table that is created during this process is shown in the screenshot below.

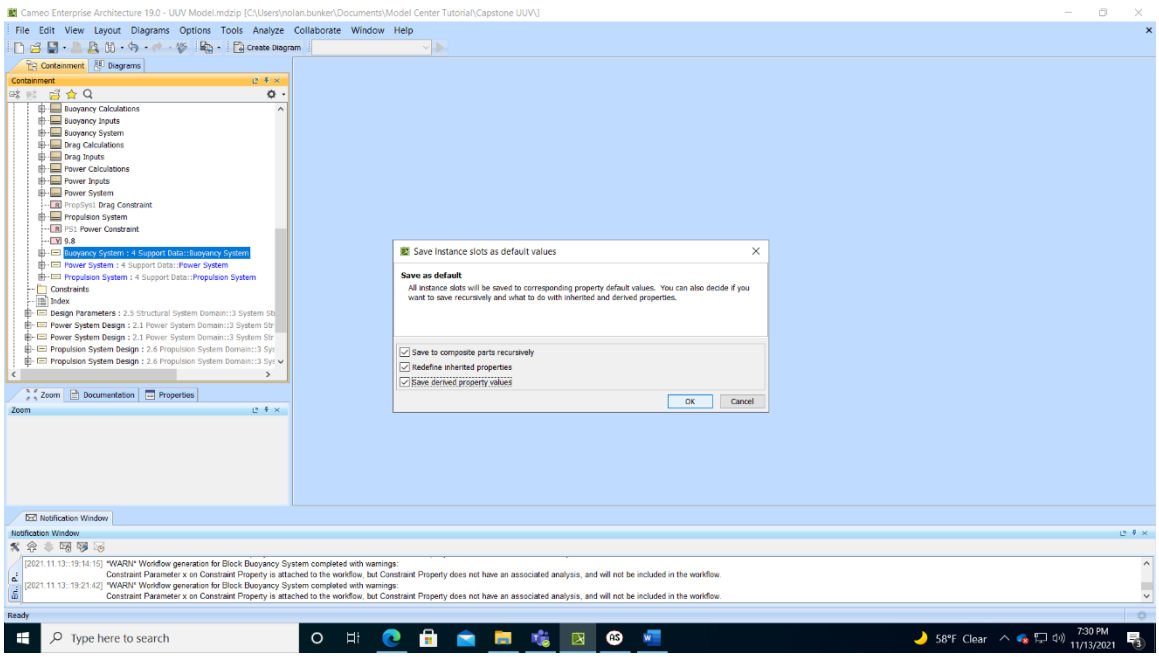


You can now close the scatterplot and the Trade study. You are not required to save it unless you wish to. You may also close out the Optimization Tool. The results will have been updated in the column on the left, this is shown in the screenshot below.



To get these values to update the values within the Cameo model click on Tools > Trade Study Plug-Ins > ModelCenter MBSE Connector for MagicDraw. Click on the results tab. Click on the “Get values from ModelCenter. This will update the values from the optimization tool you just ran. As you can see, all three of the constraints have now been met and have a green check mark. You can now either save these values as baseline values or a design instance. The process for this is the same as what was shown in the previous tutorial. At this time, we will save it as a design instance. Save it under 4 Support Data, do this for buoyancy, power, and propulsion. Close the ModelCenter MBSE Plug-in, save the ModelCenter Model if you wish. Then close out ModelCenter.

The final step is to save these design instances as the default values. Go back to your Cameo model and look under the containment tree. At bottom of UUV Optimization Model you will see a new addition. In our example they are labeled “UUV :UUV Optimization: : UUV.” Right Click on the respective system and go to tools > Save to default value(s). Once you click on save to default values this a window will appear. Ensure that all three boxes are checked, as shown below.



The final step is to check that the values have successfully updated within the model. Double click on the block definition diagram to verify the values were successfully transfered. This concludes the ModelCenter Tutorials. These tools are extremely powerful as it allows a subject matter expert to be able to do design studies without much complication. Significantly reducing the amount of time needed to fix design issues if requirements are changed. Thank you for learning with us.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. QUESTIONS FOR STAKEHOLDERS

To evaluate the effectiveness of these tutorials, a list of questions was prepared for potential end users. In order to not go into the realm of human subject research, the questions were submitted to the NPS Institutional Review Board for review. The following questions have been determined by the Institutional Review Board to not constitute human subject research and will be used to verify and validate the tutorial effectiveness:

1. Did the narrator of the video speak clearly and at a pace that was understandable?
2. Were the visuals in the video clear?
3. Were the steps easily followed (i.e., at a pace that one could follow themselves)?
4. Does the written tutorial follow the video tutorial?
5. Does the written tutorial add or subtract any value?
6. Is the scope of the video too specific or too broad?
7. Is the format of the video and written tutorials easily accessible?
8. Which aspects of the video or written tutorial were most helpful?
9. Which aspects of the video or written tutorial were least helpful?
10. Which aspects of the video or written tutorial need improvement?

These questions, along with a full example of the UUV Cameo model and tutorial packages, will be presented to stakeholders at NPS and NSWC PHD to determine the effectiveness of the tutorials as teaching tools.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Aleksandraviciene, Aiste, and Aurelijus Morkevicius. 2018. *MagicGrid Book of Knowledge.* Kaunas, Lithuania: Vitae Litera, UAB.

Hunt, John. 2014. *Guide to the Unified Process Featuring UML, Java and Design Patterns*. London: Springer.

Office of the Deputy Assistant Secretary of Defense for Systems Engineering. 2018. *Digital Engineering Strategy.* Washington, DC: Department of Defense. https://fas.org/man/eprint/digeng-2018.pdf

Pressman, Roger S., and Bruce R. Maxim.2019. *Software Engineering: A Practitioner's Approach.* New York: McGraw-Hill Education.

Shepard, David, and Julia Scherb. 2020. "What Is Digital Engineering and How Is It Related to DevSecOps?," *Software Engineering Institute* (blog). November 16. https://insights.sei.cmu.edu/blog/what-digital-engineering-and-how-it-related-devsecops/.

Undersecretary of Defense for Research and Engineering. n.d. "Digital Engineering." Accessed March 2021. https://ac.cto.mil/digital_engineering/.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California