A Pragmatic Approach to Teaching Model Based Systems Engineering: The PRZ-1

Mr. Michael J. Vinarcik P.E., University of Detroit Mercy

Michael J. Vinarcik is a Senior Lead Systems Engineer at Booz Allen Hamilton and an adjunct professor at the University of Detroit Mercy. He has over twenty-five years of automotive and defense engineering experience. He received a BS (Metallurgical Engineering) from the Ohio State University, an MBA from the University of Michigan, and an MS (Product Development) from the University of Detroit Mercy. Michael has presented at National Defense Industrial Association Ground Vehicle Systems Engineering and Technology Symposia, International Council on Systems Engineering and American Society for Engineering Education regional conferences, and a tutorial at the 2010 INCOSE International Symposium. He was a Featured Speaker at the 2016 No Magic World Symposium and is one of two Keynote Speakers at the 2017 No Magic World Symposium. Michael has contributed chapters to Industrial Applications of X-ray Diffraction, Taguchi's Quality Engineering Handbook, and Case Studies in System of Systems, Enterprise Systems, and Complex Systems Engineering; he also contributed a case study to the Systems Engineering Body of Knowledge (SEBoK). He is a licensed Professional Engineer (Michigan) and holds INCOSE ESEP-Acq, OCSMP: Model Builder – Advanced, Booz Allen Hamilton Systems Engineering Expert Belt, ASQ Certified Quality Engineer, and ASQ Certified Reliability Engineer certifications. He is a Fellow of the Engineering Society of Detroit, chaired the 2010-2011 INCOSE Great Lakes Regional Conferences, and was the 2012 President of the INCOSE Michigan Chapter. He currently co-leads INCOSE's Model Based Conceptual Design Working Group and is the President and Founder of Sigma Theta Mu, the systems honor society.

©American Society for Engineering Education, 2017

A Pragmatic Approach to Teaching Model Based Systems Engineering: The PRZ-1

Abstract:

Model Based Systems Engineering (MBSE) is transforming how systems engineering is practiced. System modeling with SysML (the Systems Modeling Language) drives rigor and crispness into the formulation of system behavior, structure, and parametrics. The author has introduced MBSE into the Systems Architecture and Systems Engineering courses that are part of the MS Product Development (MPD) program at the University of Detroit Mercy. This presentation will discuss lessons learned over the course of several years, culminating in the capstone project from the Spring 2016 Systems Engineering course.

In that course, students were required to model a polar exploration submarine, starting from a handful of system elements provided by the instructor. Over the course of the exercise, the students matured the model, increasing its detail and complexity through organic growth. The final outcome was a respectable fraction of the size of large, professionally executed efforts (such as the 30 Meter Telescope model still under development).

The significant advantages in clarity, consistency, and overall integrity of a model-driven systems engineering effort will be highlighted; an emphasis will be placed on derived work products (tables, matrices, and derived properties) and their ability to provide relevant content to stakeholders.

The MS in Product Development (MPD) Program at the University of Detroit Mercy

The MPD Program at the University of Detroit Mercy began in the late 1990s (the seventeenth cohort completed its course of study in January 2016). This program is derived from the System Design and Management Program/Product Development Track at the Massachusetts Institute of Technology (MIT), and was developed in parallel to similar programs at Rochester Institute of Technology (RIT), and the Naval Postgraduate School (NPS) through the collaboration efforts of PD21 - the Educational Coalition for Product Development Leadership in the 21st Century. PD21 engaged four educational institutions (MIT, RIT, NPS and UDM), plus six corporate and government leaders (Ford, IBM, ITT, Polaroid, Xerox and the United States Navy) to develop a program aimed at future leaders of product development within large and small organizations.[1]

[1] (University of Detroit Mercy, 2017)

In the years since its inception, the MPD program has enriched its subject matter to better serve its students (typically mid-career engineers at Detroit automotive OEMs). The curriculum currently includes the following courses:

Required courses:

- MPD 5050 Systems Architecture
- MPD 5100 Systems Engineering
- MPD 5200 System and Project Management
- MPD 5300 System Optimization
- MPD 5350 Organizational Processes
- MPD 5400 Finance and Managerial Accounting
- MPD 5450 Marketing Management
- MPD 5500 Operations Management
- MPD 5600 Product Planning & Development
- MPD 5990 Capstone Thesis and Project

Elective Courses – (Two are selected by each cohort)

- EMGT 5460 Product and Process Improvement: Lean Six Sigma I
- ENGR 5790 Mechatronics: Modeling and Simulation
- EMGT 5040 Administration of Technical Businesses
- MPD 5750 Design for X

The thirty-three hours of coursework and a three-credit thesis are complemented by a two-week, full-time "January Experience" that jumpstarts the program and give the students a chance to form bonds while working on a challenging design project. This cohort kickoff is one of the most memorable experiences in the program and consistently receives positive feedback from students.[2]

As the program has matured, it has added new content to reflect the latest advances in the engineering discipline. The author was exposed to Model Based Systems Engineering (MBSE) at the 2010 International Council on Systems Engineering's International Symposium in Chicago; by 2011, he had begun the first steps to integrating MBSE into the MPD curriculum. Early attempts included the use of DoDAF (the Department of Defense Architecture Framework) in addition to SysML.[3] However, DoDAF did not add sufficient incremental value to warrant its use; it has been dropped from the courses and both Systems Architecture and Systems Engineering now focus solely on the use of SysML. Instead of requiring students to learn, in essence, two systems engineering languages, the courses instead focus on learning and applying only one. The tables and matrices derived from the SysML models during these courses have some similarities to the canned viewpoints in DoDAF but empirically resonate better with the students. This may be, in large part, due to them "seeing" how the content is extracted (and how it answers specific questions about the system of interest).

[2] The author, a 2004 graduate of the Fifth Cohort, still has fond memories of his January Experience.
[3] (Vinarcik, The Ultra Survey Mission: Crafting A Systems Architecture Design Project, 2013)

Early Experiences Teaching SysML

Early efforts to introduce SysML into the curriculum were challenging both for the instructor and the students; the instructor had limited practical experience with system modeling and the existing reference materials, while complete, were difficult to "tack on" to an existing curriculum. Despite these challenges, students did complete term projects and successfully architected systems such as autonomous trash haulers and planetary survey missions. However, the lack of personal practical experience did hamper the instructor's ability to communicate nuances of system modeling. It is strongly recommended that any instructor teaching SysML have at least six months' worth of hands-on experience with a modeling tool under the guidance of a more experienced modeler who can provide feedback and guidance.

By the Fall of 2015, the instructor had several years of practical experience using SysML to architect and engineer systems and *SysML Distilled* by Lenny Delligatti had been published. This enabled the use of a textbook targeted at beginning system modelers as well as the application of hard-won lessons in applying SysML pragmatically. Numerous practical papers had also been published that shaped the course approach.[4] The capstone project for the Systems Architecture course that term required the students to architect a personal sustainment pod capable of keeping four people alive for an extended period (in support of disaster relief or exploration).

This paper will focus on the project used in the MPD 5100 Systems Engineering class that immediately followed and concluded in April 2016. Although *SysML Distilled* was available for use in MPD 5050, the author deferred introducing it until MPD 5100. In retrospect, this was a mistake; its content would have been immediately useful at the start of the students' exposure to SysML. Future Systems Architecture and Systems Engineering classes will both require this text so that students have a SysML reference readily available.

Essential Elements

SysML offers the experienced user a wealth of options and a rich metamodel capable of rigorously defining a system. However, the same complexity that enables it to model complicated systems can overwhelm novices. A relatively small subset of SysML can be used to model fundamental elements of any system-of-interest. This subset is less intimidating and provides sufficient value that students can understand the benefits of MBSE without drowning in details.

Philosophy

One of the key aspects of the UDM systems architecture and systems engineering courses is that students are taught how to think about problems and to ensure they are framing them deliberately. *The Design of Design* by Frederick Brooks, numerous seminal papers on systems architecture and engineering, elegant design, systems thinking, and appropriate TED videos and case studies are woven into the courses. Students are required to write papers and research

[4] Including (Pearce & Friedenthal, 2013)

notable system architects and systems-of-interest; they are also required to author their own heuristics and the rationale that underpins them. The emphasis on essays and narratives (particularly about how teams are approaching their projects) is intended to enable students to articulate their application of course principles. Just as MBSE has a focus on making the implicit explicit, the written content in the courses is intended to trigger reflection in the students and force them to become self-aware about their systems approach.

MagicDraw Tool Ecosystem

MagicDraw with the SysML plugin is the system modeling tool used in the MPD Program. The author is an experienced user of the tool and believes it is the most capable SysML tool currently available.[5] The user-friendly interface, rigorous adherence to the SysML standard, and the error checking and prevention inherent to the tool are useful; most important, however, is that it may be used to create an integrated model.

It is the author's opinion that many system architects and modelers create a series of disjointed diagrams and believe they are creating a model. The emphasis on integrated modeling in the MPD classes shows students that although occasionally useful, diagrams are NOT models. The extra effort required to develop integrated models pays significant dividends in downstream analysis and content synchronization. For this reason, an early focus on tables and matrices is critical; the students are shown that these derived work products expose underlying relationships and model health.

Early system modeling efforts in the MPD program used file-based MagicDraw. By the Fall of 2015, students were attempting to merge independent work and experiencing difficulties (the instructor had assumed that they would model collaboratively by meeting face-to-face or using screen sharing software to discuss their efforts). Their desire to model truly collaboratively required the use of a server (originally TeamWork Server, now TeamWork Cloud). This allows students to model "live" with each other and collaborate on a single model. This is particularly important because approximately 50% of the students are located in Mexico.

It has also simplified grading; the instructor may create individual models for personal assignments and has instant visibility into all of the personal and group models. This eliminates the need for students to post models to the course's Blackboard site and reduces administrative drag. It also allows on-the-spot help sessions to correct issues and provide guidance because the instructor may view and adjust the student's model directly using the shared server.

Note: Other SysML tools are commercially available; it is beyond the scope of this paper to discuss them. Any of them can be used to teach system modeling and most are sufficiently advanced to apply the principles discussed in this paper (albeit at varying levels of effort); it is recommended that the collaborative solution from any selected vendor be used for the reasons described above.

---

[5] See the author's YouTube Channel, Systems Architecture Guild (http://tinyurl.com/showmethewow), videos "Why No Magic?" and "Why SysML?" for details.

Architectural Approach

There is a growing consensus in the modeling community that the traditional flowdown of requirements and the systems engineering "V-model" do not accurately reflect real practice nor do they represent the best value proposition for programs. Textual requirements are inherently less rich, less rigorous, and more ambiguous than competently executed system models; although they are still appropriate for some contractual and cultural reasons, they should be derived from a rigorous, mature system model. Efforts to continually synchronize text requirements with a rapidly-changing system model waste effort better spent on higher model fidelity.[6]

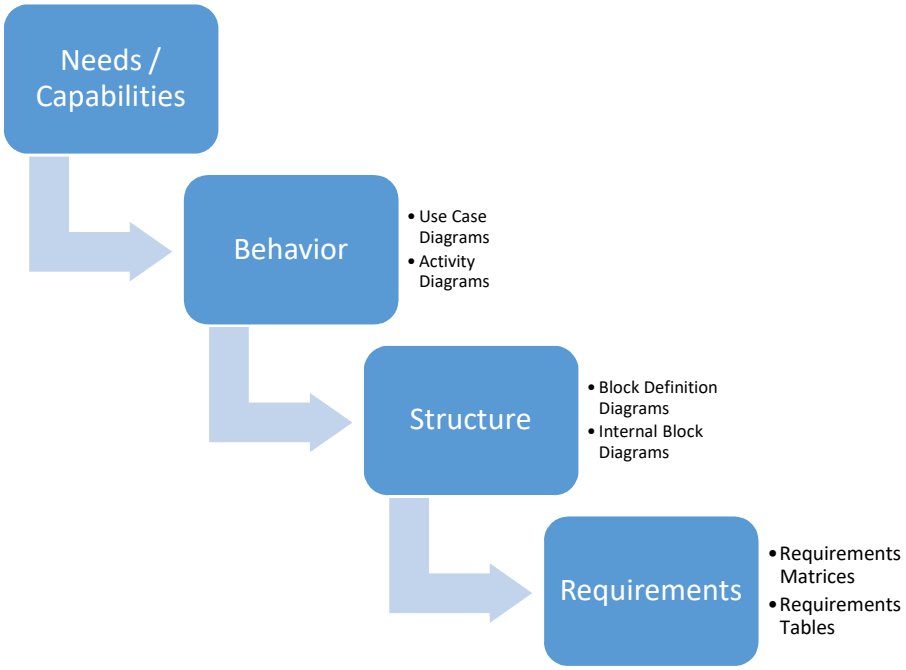


Figure 1: Content Relationships

Note that this is not a strictly linear process; as a modeler adds detail and refinement in one area, it often exposes gaps or weaknesses in another that should be addressed. A system model grows organically as detail and fidelity is added. Tim Weilkiens coined the term "zig zag method" to describe shifting between layers of abstraction and functional/logical/physical architectures.[7] This concept applies to the organic growth as well.

It should be noted that there is a very active systems engineering community on LinkedIn; there are numerous groups devoted to specific tools, SysML, and Model Based Systems Engineering. The *MBSE, Model Based Systems Engineering* group has over 7,700 members and is particularly worthy of note.[8]

---

[6] See (Vinarcik, Requirements Churn: The Hidden Drain on Systems Engineering, 2016)
[7] (Weilkiens, 2012)
[8] https://www.linkedin.com/groups/4036633

SysML Content Used

Use Case Diagram

The basic behavioral diagram used is the Use Case Diagram. Use cases are a convenient modeling element for capturing high-level behaviors or capabilities; the set of SysML elements on use case diagrams, including boundary systems, actors, and environmental effects enable sketching of system context and capability. Most students can quickly grasp the utility of the diagrams and the relatively simple syntax makes them a good introduction to SysML.

Students are given the following guidance:

Use Cases:

- Behavioral sketchpad to show behaviors/capabilities.
- <<extend>> use cases are triggered by extension points
- <<include>> use cases are always executed by the use case to which they are connected
- May be more fully described by activity diagrams

Exten	Name	Documentation	Included Use Cases	Extension Point	Extended Use Cases	Owned Behavior
1	Arctic Exploration	Arctic Exploration This is one of the main purposes for the usage of the PRZ-1 being executed by the scientific crew: the study of environment (range of temperatures, atmosphere, O2 levels, etc.), animal diversity (and their behavior), vegetation, temperature. The areas covered by this region are geographically located in Alaska (US), Canada, Finland, Greenland (Denmark), Iceland, Norway, Russia, and Sweden.		Need to Transfer Infor	Communicate	
2	Be able to emerge	Rise out of water/ice				
3	Be able to submerge	Go fully underwater				
4	Provide buoyancy	Maintain level submarine				
5	Maintain watertight integrity	Have no leaks in the hull				
6	Provide safe structure	Will not be damaged by ice or animals				
7	Provide structure and integrity	Strong enough to withstand pressure under water	Provide safe structure Travel through water Maintain watertight integrity Provide buoyancy	Submerge Be able to emerge	Be able to submerge Be able to emerge	
8	Travel through water	Be hydrodynamic				
9	Calculation and transmission of position and o	Use case condition to calculate position of submarine as it travels through the ocean. Primarily will use two different methods, an Inertial Navigation system that uses accelerometers and gyroscopes to measure the distance traveled, position and orientation of rotation of sub as it travels through the ocean.				Calculation and transmissic
10	Collect Physical Samples	collect ocean soil/artifact/ice samples				Collect Physical Samples

Figure 2: Use Case Table

Figure 2 is an extract from a table listing every use case in the PRZ-1 model, with its definition, included use cases, extension points, extended use cases, and owned activity diagrams listed.

MagicDraw makes it trivial to create tables of this sort and students are encouraged to use them to assess model quality and completeness. For example, one guideline students are given is that every major model element should be documented (even with just a few phrases or sentences) when it is created. This type of table makes it very simple to determine if any undefined elements exist.

Activity Diagrams

Activity diagrams are the cornerstone of the modeling method taught at UDM. They are one of the most generally useful of the SysML behavior diagrams and inform the development of rigorous functional decompositions.

Each action node on an activity diagram is required to be either a *call behavior* node (invoking a complete *activity*) or a *call operation* node (invoking a specific function owned by a *block*). Because *activities* and *operations* both rigorously capture inputs and outputs, they force students (and modelers) to define the inputs and outputs of each function. The use of the multiplicity concept allows for optional outputs (for example, a diagnostic function may have *fault report* and *health report* outputs, each with a [0..1] multiplicity). Judicious use of optional outputs can eliminate the need for decision nodes, since object flows out of these can invoke alternate paths downstream. Because *operations* are strictly owned by individual system *blocks*, swimlanes are expressly forbidden. It is the author's strong opinion, confirmed by personal experience, that swimlanes do little except waste space and pander to diagram-centric thinking. Ownership of functions is instead shown within the *call operations* nodes that are owned by *blocks*. This provides the same information as a swimlane, reduces wasted space on diagrams, and focuses attention on *ownership* of functions rather than *allocation*.

It is expected that students define *signals* to type inputs/outputs/results of *operations* and that each *operation* is owned by a specific *block*. As behaviors are modeled with activity diagrams, a rich set of messages (*signals*) and *blocks* (various system elements) is constructed as a result.

Students are given the following guidance:

Activity Diagrams:

- Flowcharts of behavior; describe *activities* that are made up of *actions*
- *Call behavior* actions execute other activities (activity diagrams)
- *Call operation* actions execute "leaf node" functions (the smallest behaviors we will model)
- *Send* and *accept* event actions model messages flowing into/out of activities and may be assigned to *ports*
- Complicated logical behaviors may be modeled (decision nodes, forking, etc.)

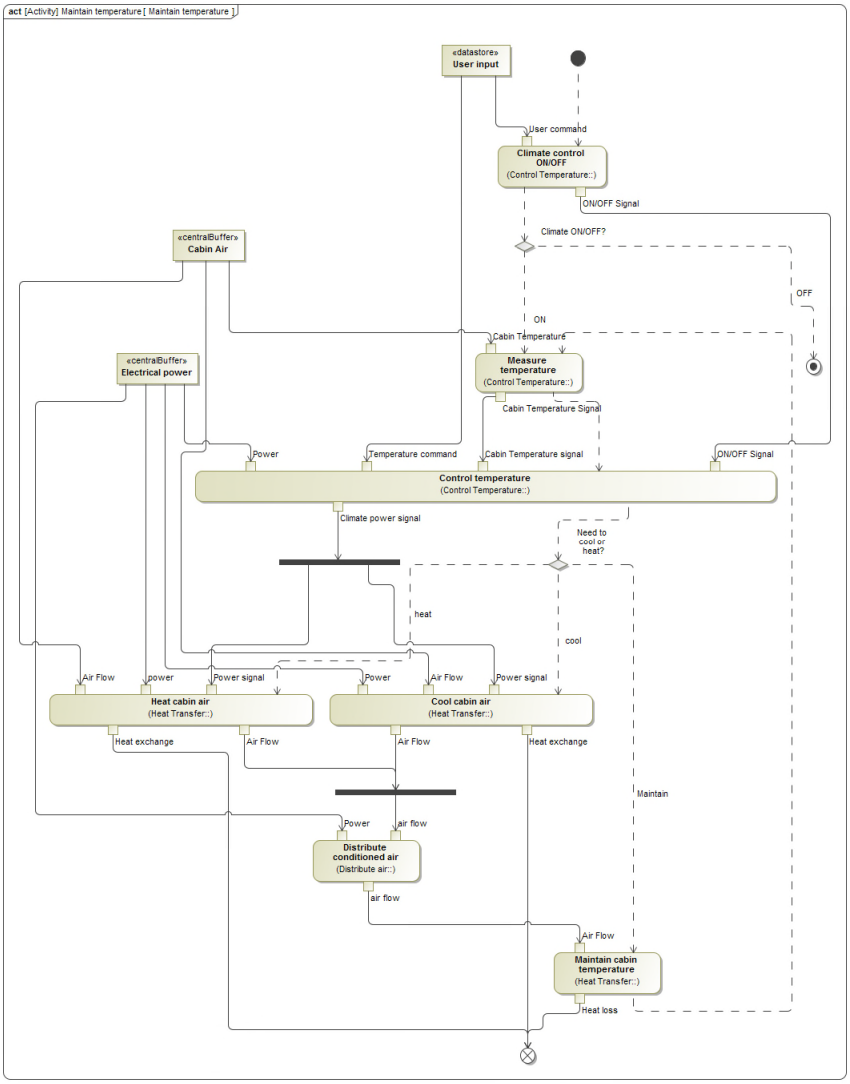


Figure 3: Activity Diagram Example

Note that no swimlanes are needed; each *call operation* node identifies the function called and the block that owns it. If multiple blocks may perform the same function, the operations may be copied from one to the other and the *call operation* nodes then identify the appropriate owner.

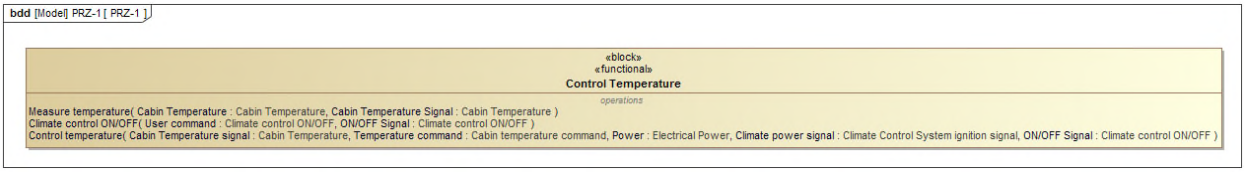


Figure 4: Example of Operations on a Block

State Machines

The author has found state machines are one of the most useful ways to integrate disparate behavioral diagrams into the system model. Use cases/capabilities and other activity diagrams can be knitted into a coherent behavioral structure using state machines. However, students

often struggle with constructing these and the instructor has found it necessary to act as a "chief architect" during term projects and construct the system state machine for the students. Student development of state machines was deferred until MPD 5100 Systems Engineering; in that course, students will be expected to construct state machines for each major system and subsystem in their architectures and to show clearly which *signals* or *operations* trigger transitions between states.

Block Definition Diagrams

Block Definition Diagrams (BDDs) are useful for sketching system context and other compositional relationships. Tables of part properties with ownership identified are often more compact representations of this model content; however, beginning users are usually more comfortable with BDDs. BDDs are composed solely of *blocks* and *composition* relationships. This results in *blocks* with *part properties*. *Aggregations* are not used because they result in *reference properties*. Any *reference property* should be represented as a *part property* at the "next level up" in the emergent hierarchy of the system and its context. This makes it clear which elements make up a system or subsystem and which are "outside" of it and connect to it via clearly defined interfaces.

Internal Block Diagrams

Internal Block Diagrams (IBDs) are used to show the relationship between system elements. Ports, connectors, and information/item flows may all be shown on IBDs. IBDs, with varying levels of granularity, are some of the most detailed work products generated by the student modeling efforts. *Item flows* and *information flows* are then used to fully describe flows across connectors.

Omitted Diagrams

Sequence diagrams are not used in the MPD program. It is the author's opinion that they are less generally useful than the other behavioral diagrams and their omission allows students to focus on mastering other content. Although they tend to be favored by individuals with a software-centric background, they are not as readily understood by non-software-centric students and can lead to confusion.

Parametric diagrams were omitted from the Systems Architecture and Systems Engineering courses (although a thesis team used them to great effect in integrating several disparate models). They will be included in a future Systems Engineering class with the long-term goal of integration of SysML parametric diagrams and the Systems Optimization class taught during the same term.

Requirements diagrams are not used; they are inefficient in representing requirements linkages (low density of information). Requirements are instead represented in tables/matrices with appropriate relationships displayed.

Tables and Matrices

One of MagicDraw's strengths is its ability to generate tables and matrices on demand of direct and multi-order relationships between elements. Students are shown how to use tables and matrices to investigate model consistency, completeness, and as a basis for rich self-exploration of a system model.

#	Name	Owner	Documentation	Call Operation	Activity Diagram
48	Cool cabin air	Heat Transfer	Transfer heat out from the air to reduce the temperature	Cool cabin air	Maintain temperature
49	Crew Evacuation	Crew Evacuation		Crew Evacuation	Emergency Evacuation
50	Crew informed and updated	Keep team informed	Final operation where crew has been informed and updated on any information they need to know and that has been transmitted through the appropriate communication channels.	Crew informed and updated	Keep team informed
51	Crew member considered as trai	Get training	Operation to set crew members as accepted after reviewing their exam results.	Crew member considered as trained	Get training
52	Crew member rejected for missic	Get training	Operation to set crew members as rejected after reviewing their exam results.	Crew member rejected for mission	Get training
53	Crew Preparation	Crew Preparation		Crew Preparation	Emergency Evacuation
54	Crew takes exam	Get training	Operation to test crew with fundamental and key knowledge of the ship, the ship´s systems and subsystems for the mission, with special emphasis in each crew member´s main activities/duties.	Crew takes exam	Get training
55	Data Analysis	Sensors- Detect Environment	Operation to complete Data Analysis for environmental detection	Data Analysis	Detect Environment
56	Deccelerate Drone	Aerial Drone Propulsion	allow aerial drag to slow down drone	Deccelerate Drone Deccelerate Drone	Return To Base Maneuver Drone

Figure 5: Table of Operations

This table lists *operations* from the PRZ-1 model; it also shows the owning *block*, the definition of the function, what *call operation* nodes reference it, and what activity diagrams call it. This is an example of how the students are taught to view the model as an integrated whole and not just a collection of diagrams.

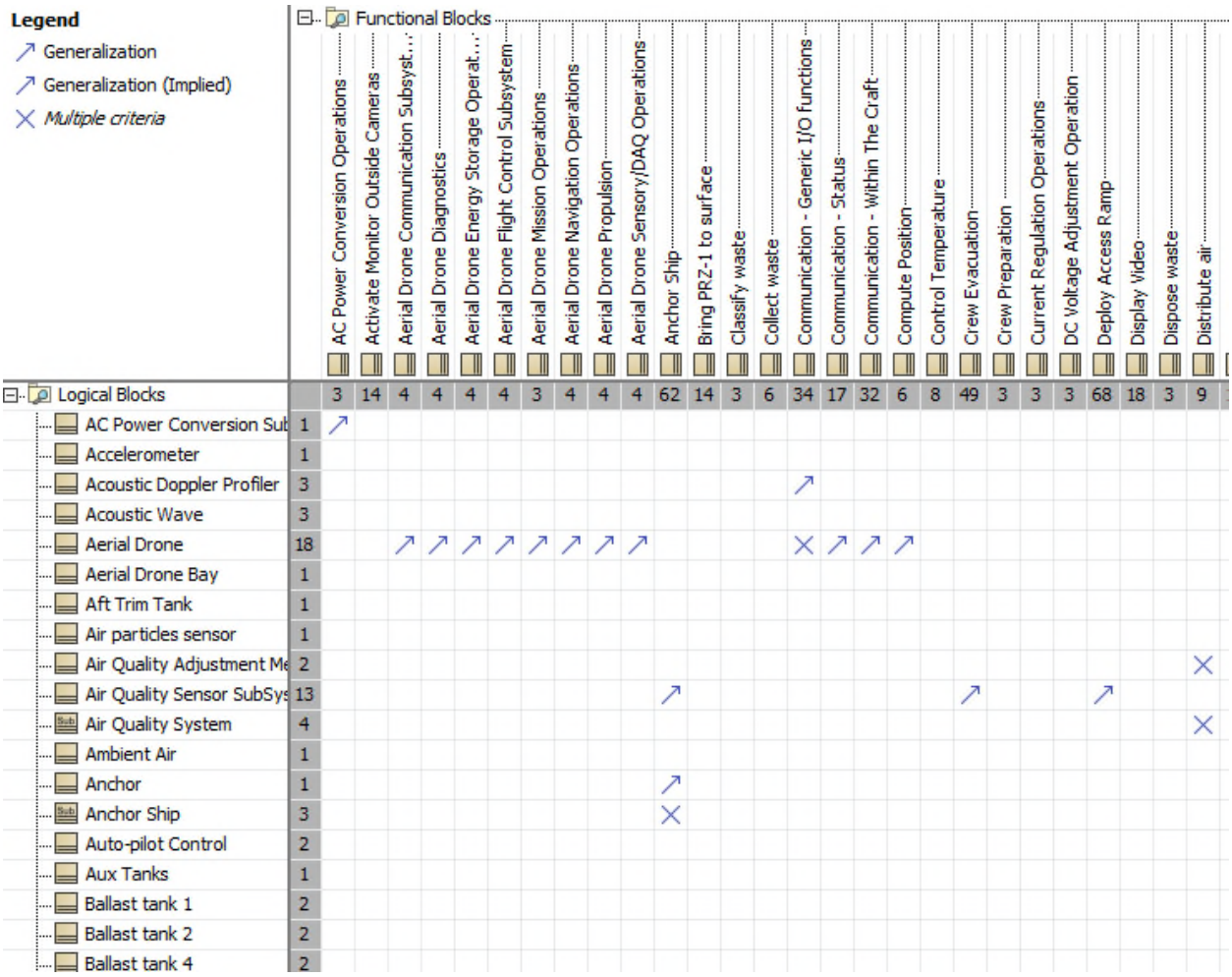


Figure 6: Matrix showing Logical Inheritance

MagicDraw's Dependency Matrix allows students to easily create matrices showing the relationship between model elements (this can also be combined with Metachain Navigation[9] to conduct complicated multi-step queries that are then displayed in matrix form).

As a further example, in the 2017 MPD 5100 Systems Engineering term project, a query was constructed that automatically displayed the linkage between science goals, objectives, and investigations and the physical elements that supported them. These linkages traced *operations* owned by physical blocks to the logical blocks, activity diagrams, and use cases that were linked to the goals (resident in a library). Constructing the system model using the relationships specified enabled rapid determination of which physical elements supported science investigations and goals.

[9] Metachain Navigation is the term No Magic uses to name its structured expression language that allows rapid development of complex model queries.

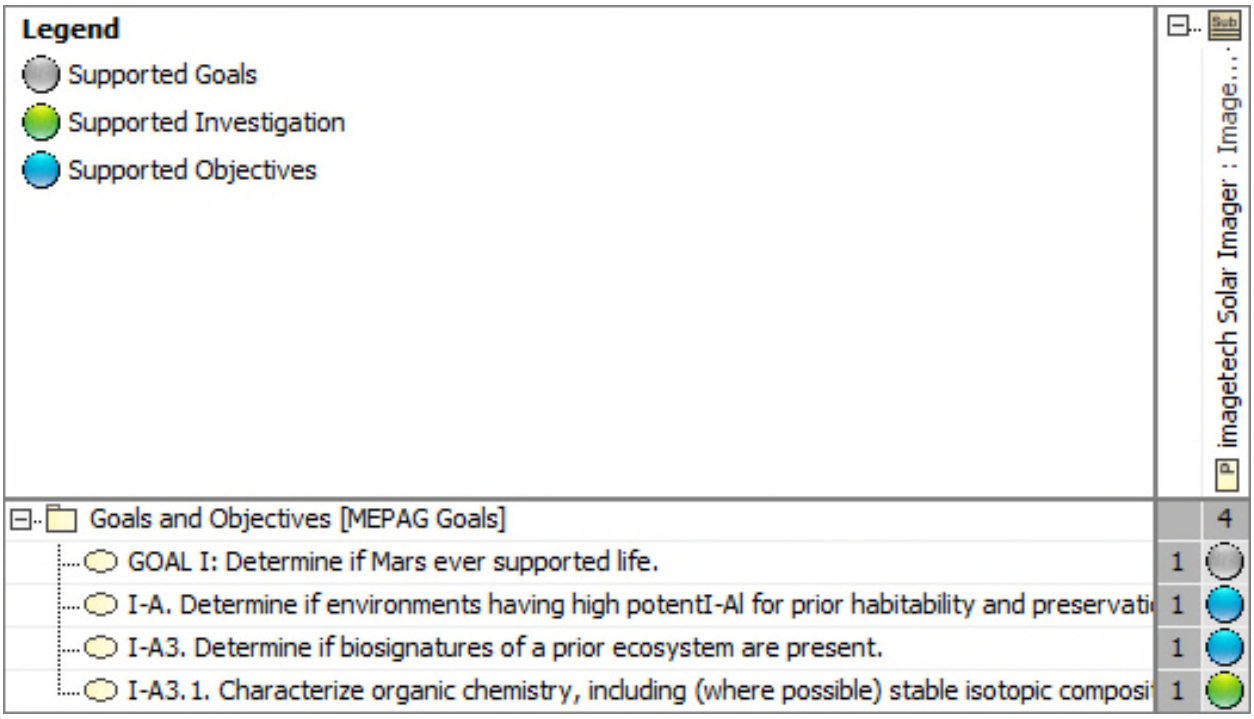


Figure 7: Physical Element to Goals/Investigations/Objectives Trace Matrix

#	Name	Physical Element
1	GOAL I: Determine if Mars ever supported life.	Imagetech Solar Imager
2	GOAL II: Understand the processes and history of climate on Mars.	
3	GOAL III: Understand the origin and evolution of Mars as a geological system.	
4	GOAL IV: Prepare for human exploration.	
5	I-A. Determine if environments having high potentI-Al for prior habitability and preservation of biosignatures contain evidence	Imagetech Solar Imager
6	I-A1. Identify environments that were habitable in the past, and characterize conditions and processes that may have influen	
7	I-A1.1. Establish overall geological context.	
8	I-A1.2. Constrain prior water availability with respect to duration, extent, and chemical activity.	
9	I-A1.3. Constrain prior energy availability with respect to type (e.g., light, specific redox couples), chemical potentI-Al (e.g.,	
10	I-A1.4. Constrain prior physicochemical conditions, emphasizing temperature, pH, water activity, and chemical composition.	
11	I-A1.5. Constrain the abundance and characterize potentI-Al sources of bioessentI-Al elements.	
12	I-A2. Assess the potentI-Al of conditions and processes to have influenced preservation or degradation of biosignatures and	
13	I-A2.1. Identify conditions and processes that would have aided preservation and/or degradation of complex organic compou	
14	I-A2.2. Identify the conditions and processes that would have aided preservation and/or degradation of physical structures	
15	I-A2.3. Characterize the conditions and processes that would have aided preservation and/or degradation of environmental i	
16	I-A3. Determine if biosignatures of a prior ecosystem are present.	Imagetech Solar Imager
17	I-A3.1. Characterize organic chemistry, including (where possible) stable isotopic composition and stereochemical configuratio	Imagetech Solar Imager
18	I-A3.2. Test for the presence of possibly biogenic physical structures, from microscopic (micron-scale) to macroscopic (meter-s	

Figure 8: Goal/Objective/Investigation Trace Table

Desired end state

The students are provided guidance for the desired end state of their system model. They are given as set of "quality check" tables and matrices that enable them to see the model content and edit it.

End state:

- All use cases decomposed by activity diagrams
- All activity diagram nodes either are *call behavior* nodes that trigger other *activities* or are *call operation* nodes triggering leaf-node *operations* on <<functional>> blocks
- Functional requirements are either <<satisfied>> by operations or by activities
- All leaf-node functions are operations on <<functional>> blocks with *in*, *out* and *result* parameters typed by signals.

- All <<logical> blocks specialize one or more <<functional>> blocks to inherit their functions[10]
- Ports have been added to the logical blocks and are typed by interface blocks
- Internal block diagrams have been created to show how logical blocks connect; all connectors have *information flows* showing what *signals* flow along them.
- <<physical>> blocks *realize* logical blocks and are used to *redefine* part properties of each physical architectural variant.

Requirements:

- All functional requirements are satisfied by operations or activities
- All interface requirements are satisfied by ports
- All physical and performance requirements are satisfied by value properties
- All design constraints are satisfied by blocks
- All requirements are *verified* by *test cases*

The PRZ-1 Project

When creating a term project, it is useful to have some knowledge of the relevant topic and underlying principles. However, allowing students to simply model comfortable systems devalues the experience; they may simply "file the serial numbers off" of work they have done in the past or that is part of their day-to-day employment experience. For that reason, it is preferred to create accessible scenarios outside the students' normal frame of reference. However, that places an increased burden on the instructor to craft believable projects. For that reason, one should select topics that are fresh in one's mind and for which one has sufficient knowledge of possible pitfalls and outcomes.

In this example, the instructor read *Nautilus 90 North* as a boy; it was a gripping account of the USS *Nautilus*'s first voyage to North Pole authored by the submarine's skipper.[11] Shortly before MPD 5100 began in January 2016, he read *The Ice Diaries,*[12] a more detailed account by the same author. The relevant discussion of the engineering and practical challenges related to exploration under the polar ice cap inspired a term project about the architecting and engineering of a notional polar exploration submarine.

To avoid forcing the students to investigate radiation shielding and other ephemera associated with nuclear reactors, the power plant for the submarine was defined as a "zero point reactor," a fictional multi-megawatt power source. This led to the project name PRZ-1 (Prototype, Reactor, Zero-Point, 1 Research Submersible).

[10] The author now prefers to eliminate the functional architecture and have <<logical>> blocks own *operations* directly. This eliminates some complications resulting from inheritance.
[11] (Anderson, Nautilus 90 North, 1959)
[12] (Anderson & Keith, The Ice Diaries: The Untold Story of the Cold War's Most Daring Mission, 2008)

The students were given a stub model with a handful of diagrams to communicate the high-level needs:

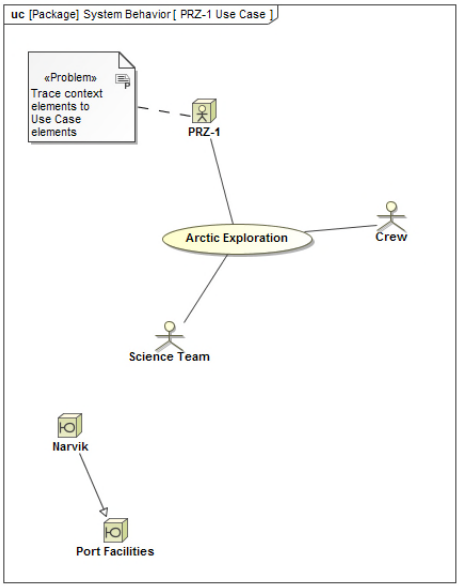


Figure 9: Initial PRZ-1 Use Case Diagram

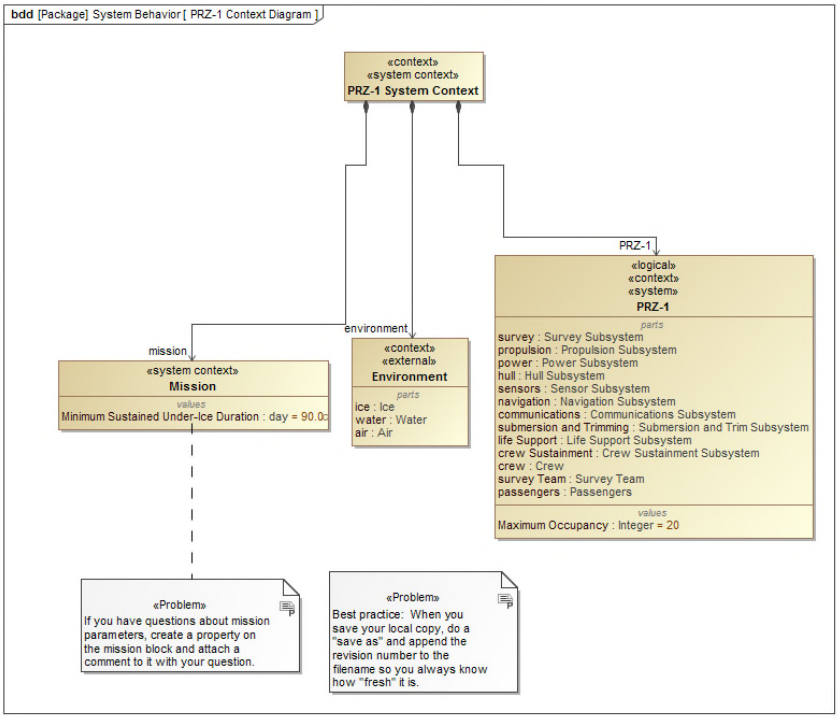


Figure 10: Initial PRZ-1 BDD

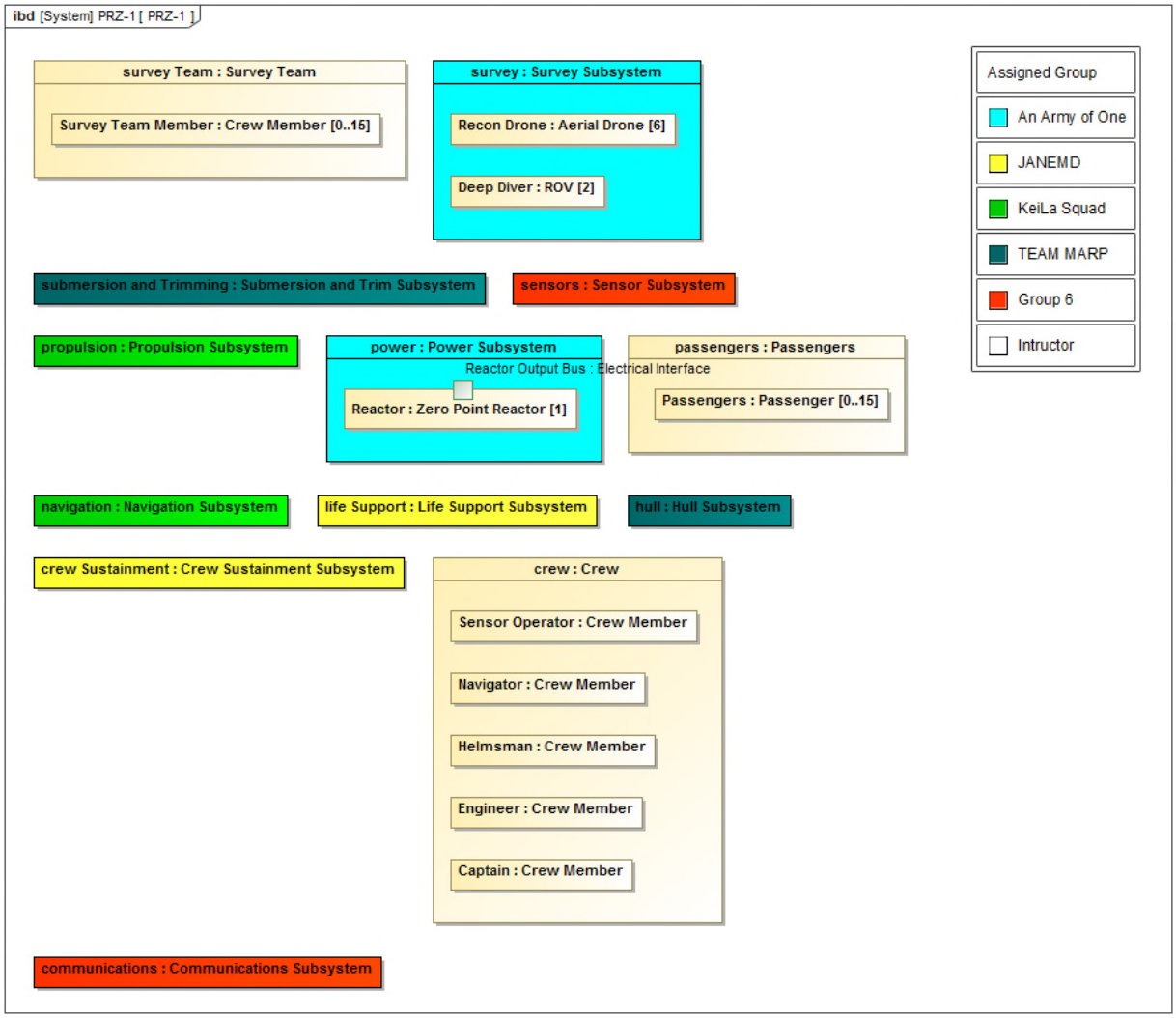


Figure 11: Initial Element Assignments

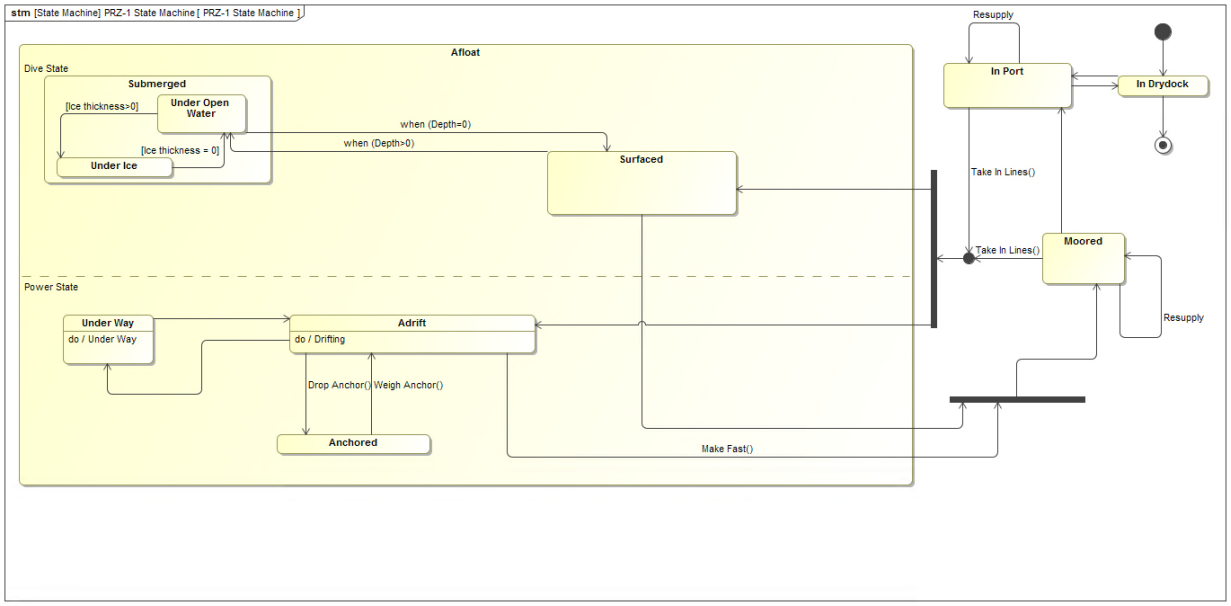


Figure 12: Final State Machine (by Instructor)

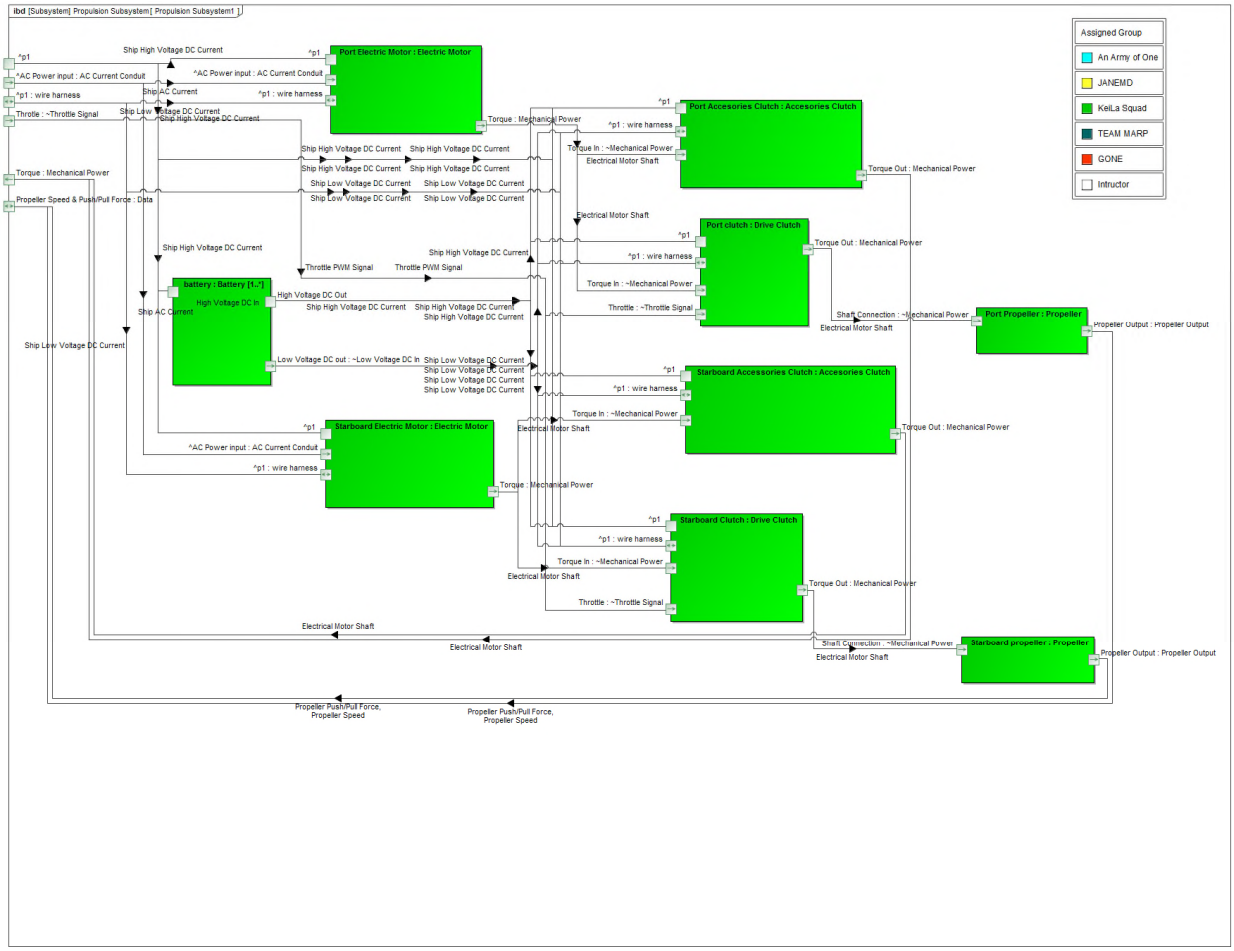
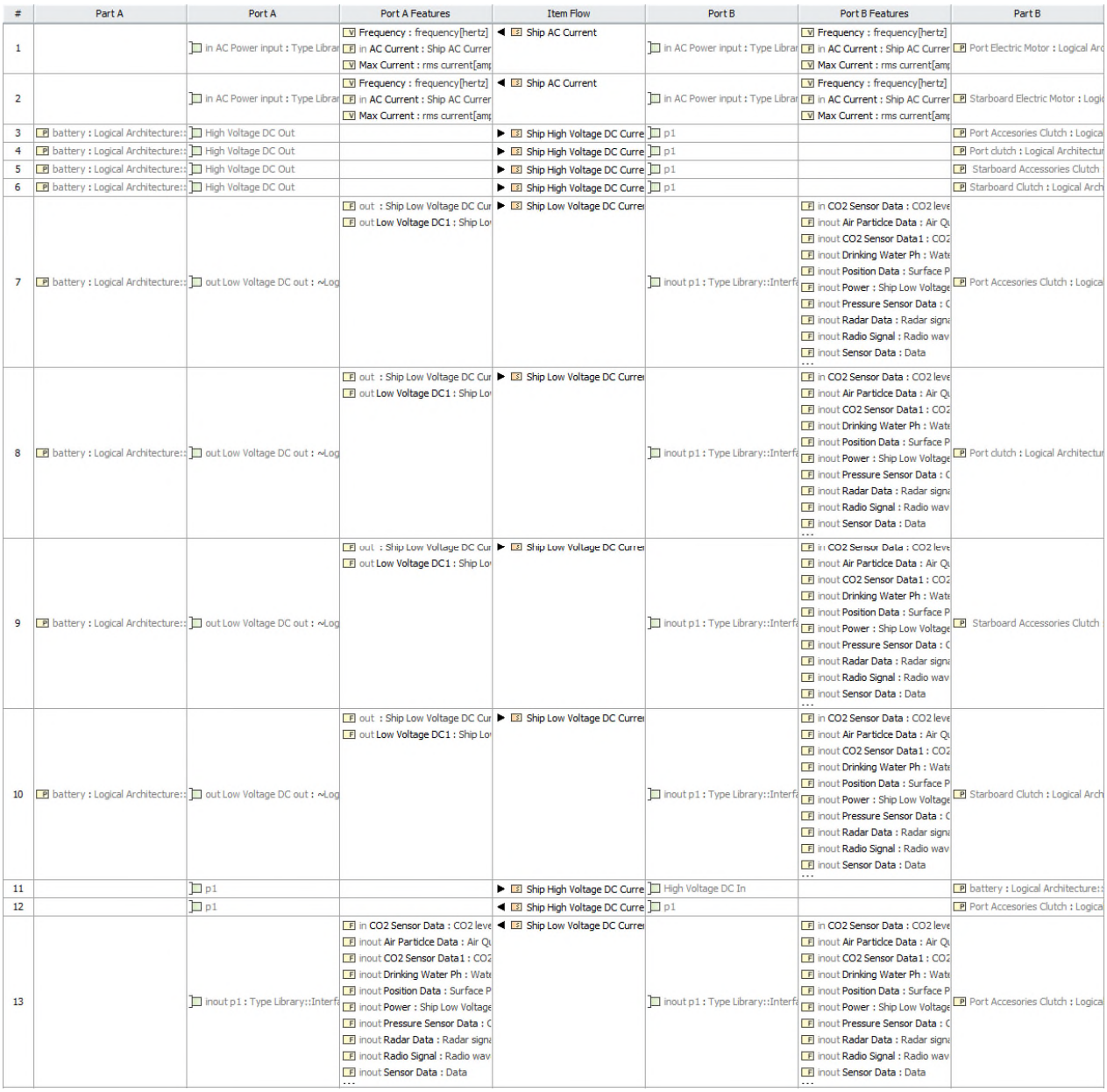


Figure 13: Example PRZ-1 IBD (Propulsion System)



#	Part A	Port A	Port A Features	Item Flow	Port B	Port B Features	Part B
1		in AC Power input : Type Librar	Frequency : frequency[hertz] in AC Current : Ship AC Currer Max Current : rms current[amp	Ship AC Current	in AC Power input : Type Librar	Frequency : frequency[hertz] in AC Current : Ship AC Currer Max Current : rms current[amp	Port Electric Motor : Logical Arc
2		in AC Power input : Type Librar	Frequency : frequency[hertz] in AC Current : Ship AC Currer Max Current : rms current[amp	Ship AC Current	in AC Power input : Type Librar	Frequency : frequency[hertz] in AC Current : Ship AC Currer Max Current : rms current[amp	Starboard Electric Motor : Logi
3	battery : Logical Architecture::	High Voltage DC Out		Ship High Voltage DC Curre	p1		Port Accesories Clutch : Logica
4	battery : Logical Architecture::	High Voltage DC Out		Ship High Voltage DC Curre	p1		Port clutch : Logical Architectur
5	battery : Logical Architecture::	High Voltage DC Out		Ship High Voltage DC Curre	p1		Starboard Accessories Clutch
6	battery : Logical Architecture::	High Voltage DC Out		Ship High Voltage DC Curre	p1		Starboard Clutch : Logical Arch
7	battery : Logical Architecture::	out Low Voltage DC out : ~Log	out : Ship Low Voltage DC Cur out Low Voltage DC1 : Ship Lo	Ship Low Voltage DC Curre	inout p1 : Type Library::Interfa	in CO2 Sensor Data : CO2 leve inout Air Particlce Data : Air Qu inout CO2 Sensor Data1 : CO2 inout Drinking Water Ph : Wate inout Position Data : Surface P inout Power : Ship Low Voltage inout Pressure Sensor Data : C inout Radar Data : Radar signa inout Radio Signal : Radio wav inout Sensor Data : Data ...	Port Accesories Clutch : Logica
8	battery : Logical Architecture::	out Low Voltage DC out : ~Log	out : Ship Low Voltage DC Cur out Low Voltage DC1 : Ship Lo	Ship Low Voltage DC Curre	inout p1 : Type Library::Interfa	in CO2 Sensor Data : CO2 leve inout Air Particlce Data : Air Qu inout CO2 Sensor Data1 : CO2 inout Drinking Water Ph : Wate inout Position Data : Surface P inout Power : Ship Low Voltage inout Pressure Sensor Data : C inout Radar Data : Radar signa inout Radio Signal : Radio wav inout Sensor Data : Data ...	Port clutch : Logical Architectur
9	battery : Logical Architecture::	out Low Voltage DC out : ~Log	out : Ship Low Voltage DC Cur out Low Voltage DC1 : Ship Lo	Ship Low Voltage DC Curre	inout p1 : Type Library::Interfa	in CO2 Sensor Data : CO2 leve inout Air Particlce Data : Air Qu inout CO2 Sensor Data1 : CO2 inout Drinking Water Ph : Wate inout Position Data : Surface P inout Power : Ship Low Voltage inout Pressure Sensor Data : C inout Radar Data : Radar signa inout Radio Signal : Radio wav inout Sensor Data : Data ...	Starboard Accessories Clutch
10	battery : Logical Architecture::	out Low Voltage DC out : ~Log	out : Ship Low Voltage DC Cur out Low Voltage DC1 : Ship Lo	Ship Low Voltage DC Curre	inout p1 : Type Library::Interfa	in CO2 Sensor Data : CO2 leve inout Air Particlce Data : Air Qu inout CO2 Sensor Data1 : CO2 inout Drinking Water Ph : Wate inout Position Data : Surface P inout Power : Ship Low Voltage inout Pressure Sensor Data : C inout Radar Data : Radar signa inout Radio Signal : Radio wav inout Sensor Data : Data ...	Starboard Clutch : Logical Arch
11		p1		Ship High Voltage DC Curre	High Voltage DC In		battery : Logical Architecture::
12		p1		Ship High Voltage DC Curre	p1		Port Accesories Clutch : Logica
13		inout p1 : Type Library::Interfa	in CO2 Sensor Data : CO2 leve inout Air Particlce Data : Air Qu inout CO2 Sensor Data1 : CO2 inout Drinking Water Ph : Wate inout Position Data : Surface P inout Power : Ship Low Voltage inout Pressure Sensor Data : C inout Radar Data : Radar signa inout Radio Signal : Radio wav inout Sensor Data : Data ...	Ship Low Voltage DC Curre	inout p1 : Type Library::Interfa	in CO2 Sensor Data : CO2 leve inout Air Particlce Data : Air Qu inout CO2 Sensor Data1 : CO2 inout Drinking Water Ph : Wate inout Position Data : Surface P inout Power : Ship Low Voltage inout Pressure Sensor Data : C inout Radar Data : Radar signa inout Radio Signal : Radio wav inout Sensor Data : Data ...	Port Accesories Clutch : Logica

Figure 14: Whitebox Interface Control Document Table (Propulsion System)

Outcome

At the end of the project, the twenty-four students (organized into six team) made 1,017 commits to the TeamWork Server. The model grew to nearly 44,000 elements; for comparison, the 30-Meter Telescope Model[13] has 186,000 elements. The PRZ-1, constructed during one term, is 23.6% of the size of a model that has been under development by professionals for several years. The 30-Meter Telescope Model is more detailed, which a much greater emphasis on parametric diagrams/relationships and more detailed activity diagrams (as one would expect from a multi-year project undertaken by experienced modelers). However, the structural definitions made by

[13] (No Magic, 2016)

the students were generally good (since modeling of system structure seems to be more readily grasped by first-time modelers).

Element	Count
Blocks	560
Diagrams	259
IBD Flows	204
Object Flows	1,026
Operations	303
Parameters	822
Ports	370
Requirements	135
Signals	181
Use Cases	112
Total Elements	43,921

Figure 15: Model Element Count

Model size does not directly correlate with model quality; however, several years of project experience suggest that by the end of two terms of modeling, students are generally performing as well as full-time modelers with three to four months' worth of experience. This demonstrates the utility of hands-on projects in developing modeling competency.

Student Learning

Students readily grasp the fundamentals of modeling and the tool user interface; although complex, system modeling is less complicated than other engineering tools (such as CAD or complex mathematical analysis packages). In general, student modeling success is roughly equivalent to their success in other aspects of the course (those who write excellent papers tend to create excellent models). Most students are productive in the modeling tool (at least at a basic level) within a month.

The creation of *blocks*, *ports*, and associated interfaces is readily mastered by most students. Use case diagrams also are rarely problematic; the activity diagram seems to be the most difficult for students to effectively master. Although they are like flowcharts in appearance, the nuances of adding *object flows* and *send event* or *receive event* elements causes some weaknesses in the behavioral analysis. As a result, the teaching approach for these concepts has been overhauled and there is now an increased emphasis on mapping the sending/receiving to specific ports in the logical architecture. This seems to create a better appreciation for the transmission of information, material, or energy from one system element to another.

One of the greatest challenges is to balance modeling sophistication and expectations. On multiple occasions, students have grasped advanced modeling concepts and spent significant amounts of time executing them. Unfortunately, simpler methods existed and were omitted by the instructor to avoid confusing students. There is an ongoing, dynamic balance between the students' grasp of the language, the tool user interface, and the tool's capabilities. Anyone teaching system modeling must be sensitive to the rate at which individual students are

progressing; the very best will quickly master the language, tool, and concepts and should be supported in their attempts to grow in capability.

Other Best Practices

While not directly related to system modeling content, the instructor does front-load the term with all the reading assignments and the bulk of the lecture content. This provides the students the background information to spend the second half of each term actively modeling. This approach has been applied to both Systems Architecture and Systems Engineering.

The instructor reserved the *problem* element type for his comments. These may be linked to any other element. A simple table of problems could then be studied by students and model errors/corrections could be easily resolved.

Care must be taken to provide prompt feedback, especially early in a model's development. System models can grow very quickly as teams add detail and if care is not taken to provide guidance significant rework may be needed to harmonize the model (particularly at the interfaces between responsible teams if a unified project is given to the entire class).

On occasion, video grading was used. Screen capturing of model navigation and review, coupled with an audio narrative of model strengths and weaknesses, may be posted privately. This provides students with a rich narrative of content and is more helpful for showing students subtleties in modeling and user interface navigation than static text feedback.

Although the instructor records class sessions for later student review, the complexity of the modeling tool and language present challenges to students new to system modeling. Therefore, the instructor began to make short videos highlighting model principles or techniques. Usually less than five minutes long, they provide targeted examples of how to create and modify system models. They may include intentional errors to show how to fix them, multiple user interface paths to the same information/process, and general commentary during the model construction.

The favorable student response to these videos and a genuine desire to improve the average skill level of system modelers led the author to create the Systems Architecture Guild website and companion YouTube channel.[14]

Finally, the instructor provides multiple help sessions (both scheduled and on-demand) to ensure the students are not inhibited by the modeling tool or SysML misunderstandings. This requires a significant personal time commitment (often six or more hours per week); however, tasks and principles that are clear to a skilled modeler may confound a novice. Rather than dilute the course content and expectations, the instructor sets high expectations and works to help the students meet them. These sessions also provide immediate feedback on the efficacy of the pedagogical approach so it may be improved for future sessions.

[14] (Systems Architecture Guild, n.d.) (Systems Architecture Guild, n.d.)

Lessons Learned / Recommendations

- Ensure students have access to basic SysML reference materials at the start of any modeling course (Delligatti's *SysML Distilled* is highly recommended).
- Standardize on a single modeling tool/environment and ensure that a collaborative server or comparable solution is available to allow student collaboration and prompt instructor intervention.
- Include non-modeling content to shape how students assess and deconstruct problems.
- Develop a clearly defined set of model elements to be used the class and provide relevant examples of their use.
- Create short videos (1-10 minutes in length) demonstrating relevant modeling techniques (this facilitates student review of difficult concepts).
- Provide a significant segment of the course (30-50%) for dedicated modeling with few, if any, other assignments due.
- Provide routine and on-demand help sessions for students.
- Remember that students may not wish to become full-time system modelers; set reasonable expectations for tool and modeling mastery.
- Include individual assignments to assess student skill and detect "free riding" in group projects.

Future Work

The existing term projects may be analyzed more rigorously to detect patterns in student behavior and the impact of the shift to two-course modeling projects. There are only two projects (the PRZ-1 and this year's Mars orbiter) similar enough and large enough to warrant this comparison.

Recent improvements in the collaborative modeling environment (notably TeamWork Cloud 18.5 and the MagicDraw 18.5 client) support improve model difference analysis (highlighting model changes and growth). These will also facilitate richer and more rigorous analysis of student models.

In the Fall of 2017, the University of Detroit Mercy is launching a Systems Engineering certificate program.[15] It will consist of the core systems-related content from the MPD program. A notable change is the creation of a dedicated SysML course that will be taught in parallel with Systems Architecture and be immediately followed by Systems Engineering during the next term. This will allow students to focus more heavily on SysML and tool language in the dedicated modeling course and allow an increased focus on core architecture concepts in Systems Architecture.

[15] https://eng-sci.udmercy.edu/academics/engineering/systems-engineering.php

Conclusion

In two terms in the MPD program (Systems Architecture and Systems Engineering), students learn how to architect, grasp fundamental systems engineering principles, learn a new language (SysML), and become proficient enough to execute personal and group projects.

The personal projects allow the instructor to provide direct feedback to the students about their personal execution of modeling and knowledge of the language; the group projects allow them to experience first-hand the advantages of system modeling.

Despite the steep learning curve and the challenges associated with learning an entirely new language as well as the class content, student response has been generally positive. Several thesis teams have chosen to complete SysML-centric theses; all have been successful. Their topics have included transmission analysis and requirements rationalization, robust description of product line variants, and the use of system models in support of new product development.

The instructor has solicited feedback from every cohort and attempted to integrate as many improvements into the course material. As a result, each MPD cohort experiences a richer, improved introduction to system modeling in support of systems architecture and engineering. For example, in the 2016/2017 class sessions, students were tasked with modeling NASA's Next Generation Mars Orbiter (NeMO) using publicly-available information. This model was created in Systems Architecture and will be further matured in Systems Engineering. As the first two-term model, it should represent greater fidelity and rigor than previous assignments and will give students insight into how much information can be rigorously captured in a relatively short span of time.

MBSE is the only viable way to manage the increasing complexity of engineered systems and SysML is a broadly supported language with the rigor and flexibility to support effective MBSE. It is the author's hope that a federation of educators, drawing from practical experience and industrial best practices, can successfully communicate its advantages to students so they may effectively apply system modeling and accelerate the transformation of the systems engineering discipline.

Bibliography

(n.d.). Retrieved from Systems Architecture Guild: www.systemsarchitectureguild.org

Anderson, W. R. (1959). *Nautilus 90 North.*

Anderson, W. R., & Keith, D. (2008). *The Ice Diaries: The Untold Story of the Cold War's Most Daring Mission.*

No Magic. (2016, March 24). *30 Meter Telescope*. Retrieved from Modeling Community Blog: https://blog.nomagic.com/thirty-meter-telescope-sysml-model/

Pearce, P., & Friedenthal, S. (2013). A Practical Approach for Modelling Submarine Subsystem Architecture in SysML. *Proceedings from the 2nd Submarine Institute of Australia (SIA) Submarine Science, Technology and Engineering Conference.*

Systems Architecture Guild. (n.d.). Retrieved from YouTube: http://tinyurl.com/showmethewow

University of Detroit Mercy. (2017, February 12). *Master of Science in Product Development*. Retrieved from Graduate Catalog: http://www.udmercy.edu/catalog/graduate2016-2017/programs/eng-sci/product-development/index.htm

University of Detroit Mercy. (2017, February 12). *MPD Program Background*. Retrieved from UDM College of Engineering and Science: http://eng-sci.udmercy.edu/programs/eng/product-development/description/index.htm

Vinarcik, M. J. (2013, August 22). The Ultra Survey Mission: Crafting A Systems Architecture Design Project. *2013 NDIA Ground Vehicle Systems Engineering and Technology Symposium*. Troy, MI: NDIA.

Vinarcik, M. J. (2016, October 8). *Requirements Churn: The Hidden Drain on Systems Engineering*. Retrieved from YouTube: https://www.youtube.com/watch?v=T84WZ4WLqw8

Weilkiens, T. (2012, March 26). *The SYSMOD Zigzag Pattern*. Retrieved from Model Based Systems Engineering Blog: http://model-based-systems-engineering.com/2012/03/26/the-sysmod-zigzag-pattern/

List of Figures:

Appendix 1: PRZ-1 Assignment Text

This project will provide you with an end-to-end systems engineering experience, from a fuzzy concept to a well-defined system architecture (including system behavior, structure, requirements, and parametrics).

This project will be structured differently; each of your teams has been assigned two related subsystems in the logical architecture. You are responsible for use cases, activity diagrams, functional architecture, logical architecture, requirements, and variants related to those subsystems.

The instructor will serve as chief architect, maintaining the integrity of the system model and assisting your teams. You are expected to use the model to communicate with each other (for example, placing a proxy port on the boundary of an adjacent subsystem to request that the other subsystem provide or accept something from you).

Rules of the road:

1. A testbed project has been provided on the server for you to practice techniques and obtain instructor feedback. Do not place the collaborative model at risk if you are unsure of what you are doing.
2. **Do not leave model elements locked if you are not editing them**. Release the locks and update the server.
3. Commit changes regularly (and update your model regularly) so others can see what you are doing and act on current information.
4. Use the type library for interface blocks, signals, value types, and units so they may be readily shared by teams.
5. Make liberal use of the "show related elements" functions (right-click on most diagrams/elements) to display missing elements (for example, a newly created port is not visible on a diagram). Do not create wayward model elements by creating "fresh" elements; reuse, reuse, reuse.
6. Use hyperlinks or <<trace>> relationships to establish the sources for assumptions, values, and other information.
7. Ask the instructor for help; if you have an idea for a table or matrix or some other way to show relationships (or help getting content into the model), ask.

Team Captain:

Each team shall designate a team captain who will be responsible for coordinating the efforts of the team and will serve as the liaison with the instructor. This individual will provide a cell phone number or other reliable means of communication to the instructor and will be on call, within reason, to resolve issues (for example, a team has left large sections of the model locked…the instructor will contact the team captain to resolve it).

Team captains may also be called to participate in meetings with the instructor to coordinate the modeling effort.

Grading:

The instructor will take snapshots of the model weekly (typically Wednesday morning) to assess your progress. You will receive a subjective score based on demonstrated effort in the following areas:

- Is the model under your control maturing?
- Is the model complete (all elements documented, connected, typed, etc.)?
- Is the subsystem "playing well with others"?
- Are there any major gaps in thinking/approach?
- Are assumptions documented/traced?
- Is the team asking reasonable clarifications of the chief architect?

These assessments will be worth 25 points each. Note that it is OK to create an element and not type it immediately if you are researching it (documentation should be entered immediately, however).

Class Format:

Classes for the remainder of the term will consist of 45-60 minute lectures (case studies and discussions of systems engineering concepts), with the balance of the class period spent as a lab session with the instructor available to help resolve issues. Class time will also be spent reviewing the model live, with commentary and discussion as appropriate.

Recommendation:

Treat this project as if it were a "real" engineering job; in many ways, it is what happens in companies around the world every day. Apply the lessons learned in systems architecture and from the books, case studies, and videos to guide your thinking.

Bonus Resource:

A video about the U.S.S. *Nautilus* has been placed in the References section. She was the first nuclear submarine and successfully transited the polar ice cap to reach the North Pole in the 1950s. There is a significant portion of the video related to under-ice navigation and operations that will be invaluable to you.

Final Report:

The model will be the final report for the class; however, I will make periodic writing assignments to you as groups or individuals to collect feedback and have you document your progress. There will also be a final presentation made by each team.