Practicing and Assessing Formal Systems Competencies in ECE Senior Design

Dr. Mario Simoni, Rose-Hulman Institute of Technology

Dr. Simoni is an Associate Professor of Electrical and Computer Engineering at Rose-Hulman Institute of Technology.

Mr. William D. Schindel, ICTT System Sciences

William D. Schindel is president of ICTT System Sciences, a systems engineering company, and developer of the Systematica Methodology for model and pattern-based systems engineering. His 40-year engineering career began in mil/aero systems with IBM Federal Systems, Owego, NY, included service as a faculty member of Rose-Hulman Institute of Technology, and founding of three commercial systems-based enterprises. He has consulted on improvement of engineering processes within automotive, medical/health care, manufacturing, telecommunications, aerospace, and consumer products businesses. Schindel earned the BS and MS in Mathematics.

Dr. Xiaoyan Mu, Rose-Hulman Institute of Technology
Dr. Dan Moore, Rose-Hulman Institute of Technology

Dan Moore is a professor in the electrical and computer engineering Department at Rose-Hulman Institute of Technology. He received his PhD in Electrical Engineering from N. C. State University in 1989 in the area of compound semiconductor growth and device fabrication. He joined the faculty at Rose-Hulman in 1995 as an associate professor of electrical and computer engineering. Prior to joining the faculty at Rose-Hulman he was an assistant professor at Virginia Tech and an instructor at N. C. State University. After completing his M.S. in electrical engineering in 1976, he joined the DuPont Corporation where he worked in various technical, design, and supervisory positions before returning to obtain his PhD. Dr. Moore directed the electrical and computer department's senior design program for several years and is currently involved in externally sponsored multidisciplinary graduate and undergraduate projects as well as international project teams and collaborations. He recently spent a sabbatical year at the University of Applied Sciences, HS-Ulm in Ulm Germany working with their design programs and finalizing a dual degree graduate program between UAS and Rose-Hulman. His current research interests include engineering design methodologies, student learning styles, active/cooperative education and the integration of entrepreneurial concepts and practices throughout the curriculum. He was the 2001 – 2003 chair of the Educational Research Methods (ERM) division of ASEE, is a senior member of IEEE, and an ABET program evaluator. He was FIE program co-chair for FIE 98, 01, and 04 and served two terms on the FIE steering committee. He is an associate editor of the on-line Journal of Advances in Engineering Education (AEE), an ASEE publication.

Dr. Wayne T. Padgett, Rose-Hulman Institute of Technology

Wayne T. Padgett is a professor of Electrical and Computer Engineering at Rose-Hulman Institute of Technology. He has been teaching signal processing and related courses there for 19 years. He received his B.E.E. from Auburn University in 1989 and his Ph.D. from Georgia Tech in 1994. He has specialized in fixed-point algorithm design and education, and has a special interest in hands-on learning. He has a variety of industrial experiences from consulting, summer, and sabbatical positions.

Practicing and Assessing Formal Systems Competencies In ECE Senior Design

Abstract

Systems engineering concepts were included for the first time this year in the Electrical and Computer Engineering sequence of senior design courses at Rose-Hulman Institute of Technology. In this year-long program, teams of three to four students complete an externally sponsored project. In this year's class, a subset of the Model-Based Systems Engineering (MBSE) Competencies was introduced at the beginning of the course, and assigned as model artifacts to appear in project deliverables. This paper presents an early report of a work in progress, and as such, it primarily describes the process of transforming the sequence of courses and the assignments that were used to provide practice with the MBSE Competencies. Qualitative assessments are included in order to provide some indication of the students' abilities to use the MBSE Competencies. The goal is to use these qualitative results to help produce meaningful rubrics that can be used next year to provide a more quantitative analysis of student performance.

1.0 Introduction

Systems engineering concepts were included for the first time this year in the Electrical and Computer Engineering (ECE) senior capstone design sequence of courses at Rose-Hulman Institute of Technology in order to address several challenges. In the senior year, students take a year-long sequence of courses, ECE460: Engineering Design I, ECE461: Engineering Design II, and ECE462: Engineering Design III. As a 3 credit-hour course in the Fall Quarter, ECE460 is focused on formulating and structuring the problem and then beginning to work on the solution. As a 4-credit-hour course in the Winter Quarter, ECE461 is focused on developing and evaluating the solution. As a 2 credit-hour course in the Spring Quarter, ECE462 is focused on documenting the solution. There is typically one faculty member who manages the entire sequence of courses, and 2–3 additional faculty members who help to supervise teams. Each faculty member involved supervises 3–5 teams for the entire year, and each student team consists of 3–4 students. The department has self-imposed a constraint that all projects must be externally driven or supported. Faculty research projects can be used as long as there is a well-defined goal to which the faculty member will hold the students accountable. As such the variety and types of projects available is dependent upon availability.

The senior capstone design course is a challenging experience for both the faculty and the students. For the students, this course is often the first experience in which they have the primary responsibility to formulate and solve a complex open-ended problem over an extended period of time. For the faculty, the challenge is to develop a course structure that

- teaches students how to break down the open-ended problem into manageable pieces and then formulate a plan for solving those pieces,

- is equally applicable and useful to a wide variety of projects,
- provides assessment tools that are an integral part of the process,
- provides opportunities for students to reflect on the usefulness of the process,
- is easily learned by faculty who haven't previously taught the course and don't have a lot of design experience,
- minimizes the overhead to faculty in terms of working with the teams and assessing their progress,
- and minimizes the additional workload on the students.

In the past, the course structure depended on the experiences of the faculty supervisors and varied from project to project. This variability and lack of formal structure made it difficult to assess one project relative to another and to maintain consistency from year to year. Throughout the years, various documentation assignments were given in various order and combinations. Other than using these documents to evaluate the student's solutions to their design problems, they were generally used in an open-loop fashion and there was no formal method of relating the contents of one document to another. Therefore, the act of producing the documentation did not force the students to continually reflect on their efforts in order to improve their results. The lack of a formal design process also meant that these documents often varied in content from project to project, making it difficult for faculty to apply rubrics consistently.

During the 2013-2014 Academic Year, model-based systems-engineering (MBSE) competencies were introduced into this electrical engineering senior design sequence in order to address these problems. The generality of systems engineering concepts makes them applicable to a wide variety of projects. The formal process of generating the models gives the students a method for breaking down the open-ended nature of the projects into manageable pieces. This formal process also means that a single design method can be learned by all faculty members and therefore applied consistently to the various projects from year to year. Having a formal process also makes it easier for faculty to work with and assess the variety of projects because there is a common language regardless of the domain-specific knowledge required for the project. The model-based approach allows the instructors to define a minimum set of models that can adequately represent the project, which helps to reduce faculty and student workload. Well-defined dependencies between the models force the students to continually reflect on their understanding of the project as they improve the consistency between the models. Because the models are tangible representations of the students' work, they are natural assessment instruments that are also integral parts of the design process. As such, the models are less likely to be viewed by the students as "extra unrelated work" than assignments whose only purpose is to provide a grade for the course. This paper describes the process that was used to transform the senior capstone design sequence, the model-based assignments that were introduced, some preliminary qualitative assessment of those assignments, and planned future improvements for each.

2.0 An Overview of MBSE and Systems Competencies

Explicit models have a long history in science and engineering, originally focused on mathematical descriptions of physical phenomena [1]. As human-engineered products became more complex, innovation and adoption cycles shorter, risks more significant, and demands for flexibility greater, systems engineering has emerged (over about 60 years) in domains such as aerospace, automotive, energy and power, telecommunications, medical and health care, advance manufacturing, and otherwise [2-5]. The emergence of explicit model-based methods for systems engineering has been even more recent (over about 20 years), improving on traditional systems engineering approaches in response to these challenges [6-18].

Model-Based Systems Engineering (MBSE) expresses system-level requirements, designs, couplings, risks, and other aspects in explicit data structures, expressed in emerging system-level modeling languages [19-20]. MBSE has special implications for education of engineers, and not only systems engineering specialists. The System Competencies, as a part of the Innovation Competencies have been identified [21-22] in connection with the education of future innovators.

Of potential interest to all engineers and innovators, the model-based approach of the Systems Competencies allows undergraduates and less experienced practitioners to more directly address competencies that otherwise had historically demanded accumulation of long and deep experience to develop. Said another way, the model-based Systems Competencies shift the emphasis from intuitive craft to explicit, observable, assessable skill. While a smaller number of systems engineering specialists pursue deeper education and practice in systems engineering, the larger engineering community, across all engineering domains, is the target of the Systems Competencies.

The System Competencies are summarized in the related literature [21] as:

- S1. Describing the target of innovation from a systems perspective;
- S2. Applying a system stakeholder view of value, trade-offs, and optimization;
- S3. Understanding system's interactions and states (modes);
- S4. Specifying system technical requirements;
- S5. Creating and analyzing high level design;
- S6. Assessing solution feasibility, consistency, and completeness;
- S7. Performing system failure mode and risk analysis;
- S8. Planning system families, platforms, and product lines;
- S9. Understanding roles & interdependencies across the innovation process.

Each of the System Competencies may be practiced and demonstrated through the use of certain types of system-level model artifacts, lending a tangible flavor. These are further described,

detailed and illustrated by Appendix B of the related literature[21]. Model-based rubrics for each of the System Competencies are described in Appendix A of the current paper.

3.0 Process for Including Concepts into Senior Design

In order to integrate these systems competencies into the senior design sequence, the course content needed to be updated and the four faculty members who would be supervising the course needed to understand them. Much of this work was begun during the summer of 2013 before the competencies were used by these faculty members for the first time. Revision of the course content began by reviewing the assignments from previous years to see how systems competencies could be integrated and assignments reorganized. These assignments are presented in much more detail in the following section.

In addition to revising the assignments, the lectures that introduced those assignments also had to be updated. Two two-hour lectures, with interactive exercises for the students were developed during the summer and presented in the first two weeks of ECE460 in the Fall Quarter. The students were asked to come to the lectures with background knowledge of their project, so that they could apply the systems competencies to their projects as active learning activities during the lecture. The first lecture presented the System Competencies S1–S3, and gave the students an introduction to high-level system modeling. The most common problem noticed during this lecture was that students wanted to immediately begin with low-level physical components or code, but through questions, were taught to refocus their thoughts on a much broader perspective. The students were asked to return the following week with a first version of system models.

The second lecture presented the System Competencies S4 and S5, which describe respectively the synthesis of system requirements and decomposition and synthesis of lower-level system models down to the physical level. While assignments related to these competencies would not be collected until the end of the Fall Quarter, the material was presented early in the quarter so that students could have an understanding of the whole design process and what would be expected of them. These competencies were reviewed again in a 1-hour lecture two weeks before that assignment was due. The in-class activities were based on applying these competencies to the system models that they brought to class. The greatest difficulty for the students during these activities was to understand requirements as input-output relationships.

The lectures and activities helped the students to get started on their projects, but were significantly inadequate to give them a full understanding of the competencies or how to apply them. The true learning process was the application of the competencies throughout the project. In order to help the students learn this process, the faculty supervisors meet weekly with each team. These meetings were crucial to the students' ability to use the systems competencies because they allowed the supervisors to provide regular feedback. Most teams were able to produce a reasonable first draft of the system models after 5–6 weeks of effort.

The weekly meetings with the student teams revealed that the four faculty supervisors were sometimes unprepared to deal with questions because of a lack of their own experience with MBSE techniques. This became evident when students were trying to create a high-level model of the system modes. Many of the student teams were producing state machines for a small subset of their system rather than thinking about the entire system lifecycle. The supervisors were not able to identify this and make the corrections until they met as a group and discussed the problem. As a result, the supervisors began meeting weekly in order to practice and reflect on the MBSE design process. The supervisors divided into two groups, each group going through the process for a simple design and each supervisor in that group working independently. The designs chosen were a thermostat and a traffic light system. As a result, the supervisors were able to compare two different designs for the same system and then again to see two different systems. These experiences helped the faculty to identify and solve some of the problems that students were facing such as

1. how to effectively define the system boundary,
2. how to derive the models for a system when there is a lack of domain-specific knowledge for that system,
3. understanding the definitions for the various modeling components such as actors, IOs, and events,
4. how to reconcile differences between two different designs of the same system,
5. the benefit of having a database of model templates for commonly-used systems, actors, and input-outputs such as a DC motor.

The actual models that were produced by the faculty during these weekly meetings will be used to improve the students' preparation for the use of MBSE in senior design as will be discussed in a later section.

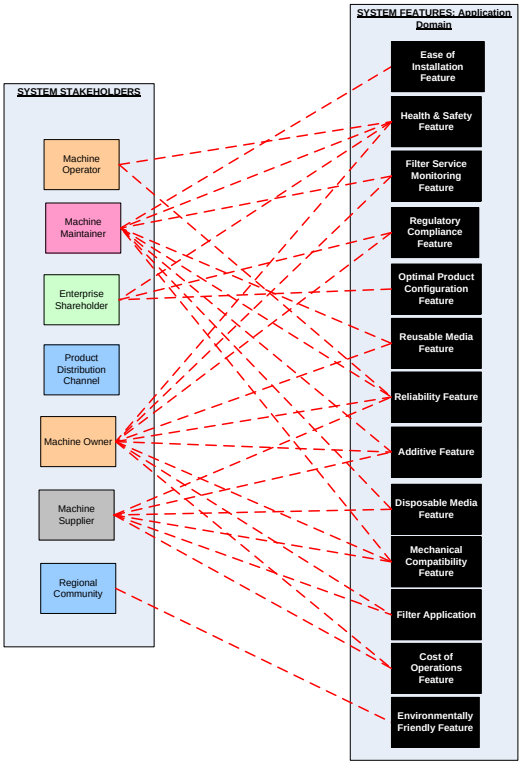
4.0 A Description of the Model-based Assignments for ECE Senior Design

The typical extent to which students are able to work on their projects in the three quarter sequence of courses (ECE460, ECE461, and ECE462) fits best with systems competencies S1–S7 as described previously. This subset of competencies was integrated into four primary assignments that were distributed among the courses: 1) System Proposal, 2) System Decomposition and Requirements, 3) Test Plan and Requirements Evaluation, and 4) Failure Modes Analysis. Because this effort has not yet been sent to the Institutional Review Board, examples of student work could not be included. To give some sense of what the students were able to accomplish, the templates that were given to the students are shown and the common strengths and weaknesses of the student submissions are provided. In some cases the students have not yet completed the assignment for this year, so no data can be given. The experiences of working with the students are being used to develop and refine a set of rubrics that will applied to the 2014-2015 design teams.

4.1 The System Proposal Document

The System Proposal document is completed in the first 3–5 weeks of the first capstone course, ECE460. This document contains a set of models that describe the high-level system design for the project (competencies S1, S3, and S5). These models are all traceable to the stakeholder features, which are specified by the client (competency S2). Because of this perspective and input from all constituencies, the first draft of this document is used as a contract between the team, the client, and the faculty supervisor. The System Proposal consists of the following content

1. a brief introduction to the project,
2. the Stakeholder model,
3. the Domain and Logical Architecture models,
4. the System Modes (states) model, and
5. a complete set of Interactions.



Stakeholder	Desired fetures and their attributes/descriptions
User	**Signal Processing Feature** 1. sample the input signals 2. filtering the input signals with 1st and 6th order filters 3. multiply and add the input signals to supplementary inputs **Signal Input Feature** 1. ECG Input with isolated ECG circuits and RLD output 2. Microphone input 3. Generic high-impedance voltage input 4. Instrumentation amplifier input for differential sensors **Signal Output Feature** 1. Be able to drive 8Ohm speaker load at full rails with low distortion 2. Be able to drive high-impedance load **Configurability Feature** 1. quickly and easily configure the board for a wide variety of experiments **Affordability Feature** 1. purchase the board for less than the cost of a standard textbook **Power Supply Feature** 1. provide sufficient power to drive 8 Ohm load to full rail 2. Low noise 3. Accept input from standard AC/DC wall power supply 4. Power indicator lights 5. Separate power supply for isolated ECG circuits **Usability Feature** 1. easily interface with a wide variety of equipment 2. use the board at home for homework assignments and distance learning **Durability Feature** 1. Will last for several years in spite of student misuse and mishandling.

Figure 1: These two stakeholder models were given to the students as templates.

Table 1: Strengths and weaknesses of student submissions of the stakeholder models

Some common strengths were…	Some common weaknesses were…
• Identify the primary stakeholders	• Missing the feature for the primary purpose • Didn't define the features very well • Didn't explicitly indicate which features were most important

The Stakeholder Model maps the stakeholders to a set of features they desire from the project. A *stakeholder* is defined as an individual or group who has a vested interest in the outcome of the project. *Features* are value based properties that the stakeholders would like the outcome to have and essentially define the problem to be solved. Ultimately, the students will know if they have successfully completed the project if they can show that their solution adequately satisfies the set of features described in this model. This feature space also gives the students a framework in which to evaluate different design decisions. If there is conflict between features or if it becomes difficult to meet all of the features, this model helps the students to prioritize their choices. In essence, this model is the primary tie back to the client, and as such, requires extensive discussions between the client, the supervisor, and the students. Any significant changes to this model after the initial client approval must be accompanied by the client's signature. Two different templates for the stakeholder model that were given to the students are shown in Figure 1 and the strengths and weaknesses observed are shown in Table 1.

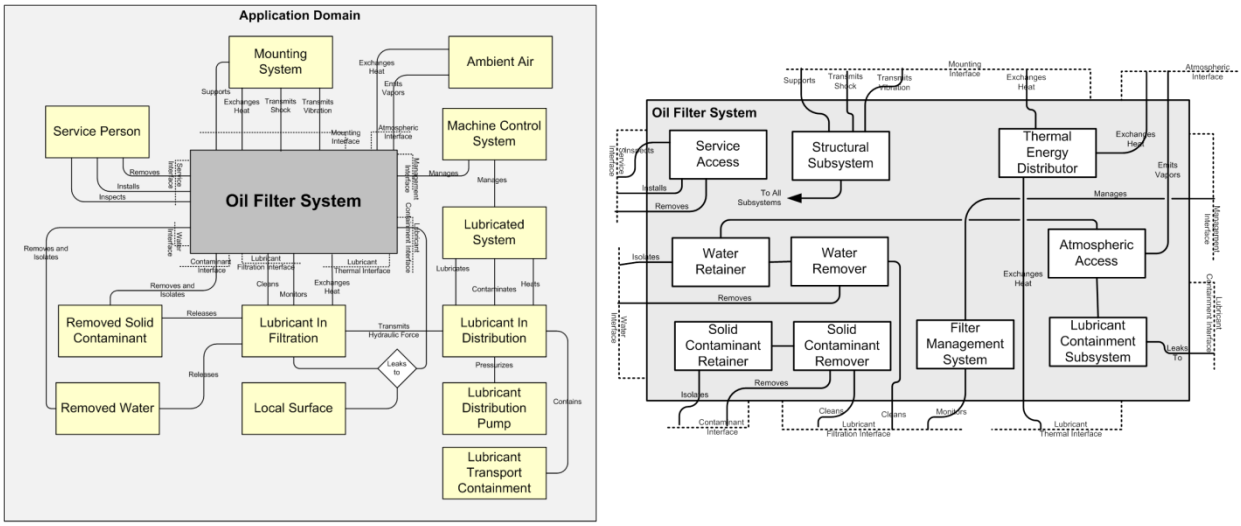


Figure 2: Domain (left) and Logical Architecture (right) models that were provided to the students as examples.

The Domain and Logical Architecture models are the first physical description of the students' solution to satisfy the feature set. These models define the system boundary and the system itself from each side of that boundary. The Domain model describes the system with respect to its external environment, while the Logical Architecture decomposes the system into high-level behavioral blocks. In the Domain model, students are introduced to *actors*, which are external elements that physically interact with the system through *input-outputs* (*IO*s), which describe what is physically transferred between the actors and the system. *Interfaces* at the system boundary group the IOs into common media or transport mechanisms of interaction. The students learn to synthesize the Domain model by asking what *interactions* must occur between the actors and the system and what IOs are necessary to describe those interactions. They learn to synthesize the Logical Architecture by asking what the system must do to each input in order to produce the corresponding outputs. Repeated evaluation of the models by the faculty supervisor

helps to validate that the described solution will completely satisfy the feature set. The templates for the domain model that were given to the students are shown in Figure 2 and the strengths and weaknesses observed are shown in Table 2.

Table 2: Strengths and weaknesses of the Domain and Logical Architecture models that were produced by the students.

Some common strengths were…	Some common weaknesses were…
• Identify the primary actors • Identify the primary relationships between actors and the system • Demonstrate an understanding of their system as part of a larger system • Demonstrate a high-level understanding of what the system does in order to convert the system inputs into system outputs	• Didn't label or define the IOs either inside or outside the system boundary • Connecting two different IOs together • Missing some actors • Incorrect system boundary • Actors were actually behavior blocks within the system • Lack of or missing interfaces • Confusion between IOs and actors • Not accounting for every IO in the logical architecture

The System Modes (states) model is a representation of the interactions that occur while the system is in a specific mode of existence and covers the entire system lifecycle. The students are asked to synthesize this model by

1. defining the entire *lifecycle* for their system,
2. dividing that lifecycle into general *modes* of operation,
3. defining *events* that cause the system to transition from one mode to another, and
4. determining which *interactions* occur in each of the modes.

The students are told that there is likely to be many different ways to organize these situations and interactions into different modes. The important property of this model is the inclusion of all of the different behaviors throughout the lifecycle. Again, evaluation of the model by the faculty supervisor is necessary to validate that this model covers the full scope of the project. Figure 3 is the example shown to students that indicates what was expected.

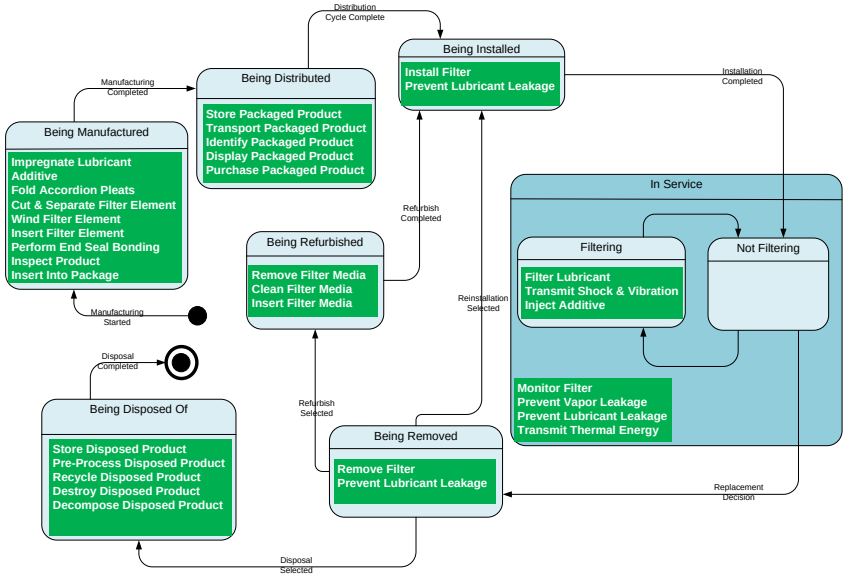


Figure 3: The system modes model that was used as an example for the students.

Table 3: Common strengths and weakness of student submissions of the system modes models.

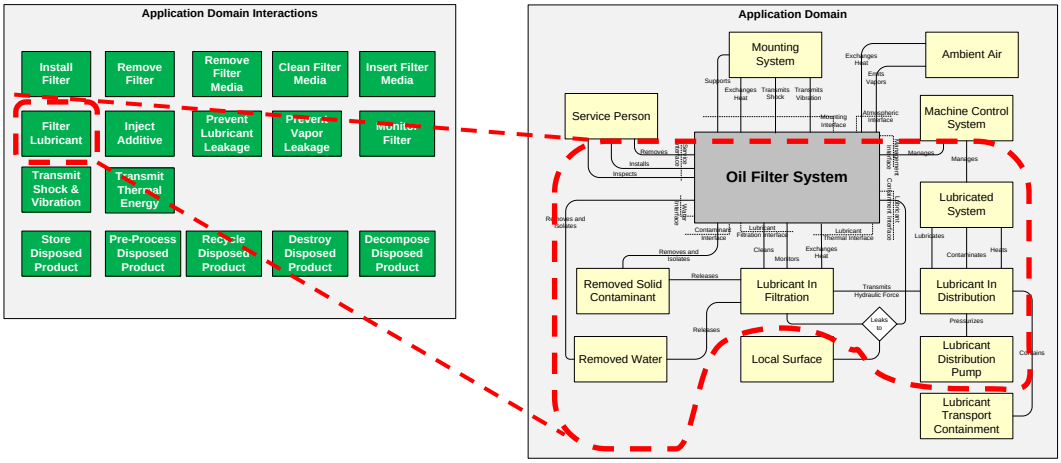
Some common strengths were…	Some common weaknesses were…
• Demonstrated consideration of modes that were not identified in previous years such as shipping, installation, and repair • Were able to capture the primary system modes	• Didn't label the events that caused state transitions • Created a finite state machine for computer code or hardware instead for the whole system modes • Didn't include the list of interactions in the definition of the mode • Not connecting events to sub-states • Not including definitions of the states

The Stakeholder, Domain, Logical Architecture, and System Modes models should be understood to be "views" or cross-sections of a single underlying holistic consistent system model. A key value in using these different views to formulate and structure an open-ended design problem is in how they relate to each other. If only a single view is used to describe the project, there is no way to ensure the validity of that model, and it also describes only one aspect. By successive refinement to ensure consistency among the different models, there is a greater probability that the whole problem space is understood, so that the proposed solution will successfully cover the feature set. A key part of that consistency lies in iteratively comparing the explicitly defined ***interactions*** that are visible in the different model views. In other words,

1. The same consistent set of physical Interactions should be found in or traceable to each of the different high level "views";
2. Those views and the list of Interactions should be refined until this consistency is achieved.

This kind of explicit model-based consistency also appears in the rubrics of Appendix A. After several iterations of these consistency checks, each of the model views should eventually arrive to a steady state form. This steady-state form at the end of the first 3–5 weeks of ECE460

becomes the official System Proposal Document. An example of what was expected from the students in terms of interactions is shown in Figure 4.



Interaction	Description	Actors involved	Occurs During State	Features Satisfied
Filter Lubricant	The routing of contaminated lubricant to the filter system, removal of contaminants from the lubricant, returning the clean lubricant to the system, and dealing with the removed contaminant	Lubricated System, Lubricant in Distribution, Lubricant Distribution Pump, Lubricant in Filtration, Removed Contaminant, Removed Water	In Service, Filtering	Filter Application Market-Application Coverage Reliability Feature

Figure 4: This template was used to illustrate how actors and IOs (as identified by the dashed outline in the diagram) can be associated with a particular interaction, which is also defined more specifically by the corresponding spreadsheet entry.

Table 4: Common strengths and weaknesses of the interactions that students generated.

Some common strengths were…	Some common weaknesses were…
• Demonstrate some understanding of the relationship between the different models • Demonstrate the need to tie every design step back to the stakeholder model	• Lack of consistency between the model views • Incomplete list of interactions • The scope of interactions were either too coarse or too fine

4.2 The System Decomposition and Requirements Document

The System Decomposition and Requirements Document addresses Systems Competencies S4–S6, and is completed in the last 2 weeks of ECE460. By the end of the first quarter, the students have a much better understanding of their project and have begun to decompose their logical architecture and synthesize a physical system design. The purpose of this document is to describe that lower-level design, show how it is derived from the higher-level models, and provide requirements that can be used to verify the solution. The first part of this document contains a revision of the Domain and Logical Architecture models. Revision of these models is necessary to ensure that the lower-level design is consistent with the current understanding of the project.

This revision also provides a tangible focus for the students to reflect on the current status of their project with respect to their initial understanding.

One of the primary strengths of MBSE is the explicit hierarchical and relational nature of the models that allow any physical part or code in a system to be explicitly traced all the way back to the feature set that was specified by the client. With such tracing, the inherent value impact of every physical part of the system can be verified. Unfortunately, this process of progressive hierarchical decomposition can require more documentation and bookkeeping than is practical for the given time constraints of a senior design course. In order to keep the documentation at a manageable level, the students are asked to produce only three levels of hierarchy, the Domain (black-box) model, the Logical Architecture (white-box) model, and then a synthesized Physical model.

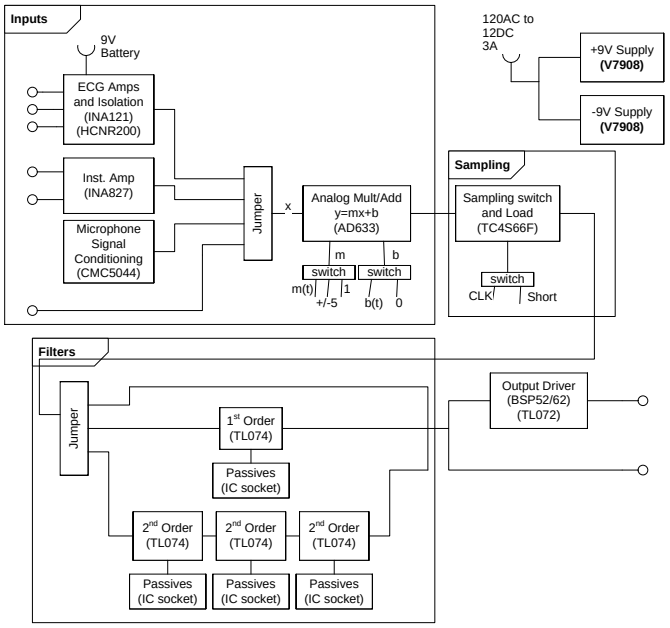


Figure 5: This diagram was presented to the students as an example of a synthesized physical architecture.

They Physical model is described with another block diagram, the exact nature of which depends on the specific project. The diagram can be composed of schematics, mechanical diagrams, simplified blocks that represent small sub-circuits or other hardware, and/or flow diagrams or UML models or source code for software. Whatever the exact form, the diagram must contain some reference to physical part numbers or code functional modules where appropriate so that there can be some validation with respect to the Logical Architecture and the requirements. Figure 5 is the example of such a diagram that was provided to the students. The circles and arcs at the ends of lines represent IOs at the system boundary. The boxes labeled Filters, Sampling, and Inputs represent behavioral blocks from the logical architecture model and help to validate the design. Important part numbers are indicated in parentheses within the block.

Table 5: Common strengths and weaknesses of physical architectures that students generated.

Some common strengths were…	Some common weaknesses were…
• Showed how the physical architecture is related to the high-level system design	• The inputs-outputs of the physical blocks were not labeled • The system IOs were not explicitly labeled in the physical architecture

Synthesis of System Requirements is necessary in order to provide objective technical descriptions of system behaviors delivering the relatively more subjective Feature set. Requirements are defined as any quantifiable and/or measurable input-output relationship in the system model. If the complete MBSE approach is used, requirements are first defined at the Domain Model level (black-box requirements), which describe all interactions between the system and its environment at the system boundary. These black-box requirements would then be decomposed along with each hierarchical level of the system architecture down to each physical component. The extent of this description may be judged impractical for an initial experience in senior design. Instead, the students are asked to fully decompose only one or a small subset of the black-box requirements into white-box requirements for the Logical Architecture and finally for the corresponding physical components in the Physical Architecture.

Table 6: Example of how students were asked to report their system requirements.

Interaction	Block	ID	Requirement	Stakeholder Features
Measure the ECG	Input Signals	IS-1	The **ECG electrodes** must be isolated from the **main power supply** by at least [5000V].	Safety
Measure the ECG	Input Signals	IS-2	The ECG circuits must amplify the **electrode input** such that the **ECG output signal** voltage is greater than [500mV peak-peak]	Usability

Table 6 is an example that was presented to the students and shows how they were expected to present the requirements for their system. The requirements shown here are black-box requirements for an ECG system. In this table, the requirements are related to the high-level design by specifying the interaction, any blocks in the Logical Architecture that would be involved in satisfying these requirements, and the stakeholder features to which it is related. The requirements are ***validated*** by the faculty supervisor against the features that they support. The ID is used to help ***verify*** the physical architecture against the high-level design. In the requirement definition, the bold words represent IOs or actors in the Domain model and the square brackets represent the measurable input-output requirement. The information in this table provides a rudimentary mechanism to help ***validate*** that each interaction, behavior, and feature is addressed by the physical architecture design.

Table 7: Common strengths and weaknesses of the requirements that students generated.

Some common strengths were…	Some common weaknesses were…
• Those requirements that were expressed as input-output relationships were quantitative and measurable • Requirements were closely tied to the physical design	• Not all requirements were expressed as an input-output relationship

While Table 6 is used to help validate the physical architecture ***design***, another mechanism is necessary to verify that the actual ***components and/or code*** can satisfy the black-box requirements. The matrix shown in Figure 6 is used for this purpose. Each row in this matrix corresponds to a physical component or computer code block in the physical architecture design, and each column corresponds to a black-box requirement. The center of the matrix where the rows and columns intersect is used to verify the design by indicating with a mark which physical components satisfy which black-box requirements. By fully completing this mapping, there is at least some tangible evidence that each of the requirements has been addressed by a physical component.

Requirement	IS-25	PS-5	PS-2	OD-3	BB-3	ST-1
AD633	X					
Molex connector	X					
LM351		X				
BP51 and BP52					X	
Rubber footings						X

Figure 6: An example of the type of verification matrix that the students are asked to complete.

Table 8: Common strengths and weaknesses of the validation matrices that students generated.

Some common strengths were…	Some common weaknesses were…
• Demonstrated that the physical components could be traced back to stakeholder features and also implement the design	• The logical architecture blocks were used instead of the physical components

In order to synthesize the physical design and requirements, the students must make many design decisions. The students were already familiar with the mechanics of creating a decision matrix, but not with how to generate meaningful criteria. The MBSE approach provides those criteria in the form of the stakeholder feature set. A decision matrix was included as another part of this document in order to provide some limited evidence of evaluating design choices against the feature set. Because the students created decision matrices in lower-level courses, they were already familiar with the concept so that no example was provided.

Table 9: Common strengths and weaknesses of the decision matrices that students produced.

Some common strengths were…	Some common weaknesses were…
• Decisions were tied back to the stakeholder features	• Some of the stakeholder features were missing • Criteria that were not stakeholder features were used to make decisions

4.3 The Test Plan and Requirements Evaluation Documents

The Test Plan and Requirements Evaluation Documents focus on systems competency S6, and occur in the middle of ECE461. At this point in the sequence of courses, the students should have completed some portion of their physical design and are able to verify it against the corresponding requirements. This assignment consists of producing a unit and system-level test plan and then carrying out the unit-level plan and documenting the results to verify that the associated requirements are met. By the end of the project, the students must verify that they meet all of their system requirements, but in order to save time, they are asked to formally document and carry out test plans for only a small subset of them. The unit-level plan is focused on the requirements pertaining to a certain physical component or behavioral block from the logical architecture. The system test plan is focused on a requirement that is related to an IO at the system boundary with respect to an external actor and also involves more than one unit of the system. Both the test plan and evaluation of the requirements were documents that were required in previous years. However, the use of MBSE concepts provides a formal vocabulary and process for creating the test plans and shows explicitly how verified requirements trace back to the stakeholder features. The test plans have not yet been completed by the students and the results are usually several pages long so it is impractical to include them in this paper.

4.4 Failure Modes Analysis

The final course in the sequence is ECE462, in which the teams produce the final documentation that is given to the client. Because this course occurs in the Spring Quarter, the relevant assignments have not yet been given to the students. In previous years, the documentation assignments were primarily based on a final report that summarized the work from the year. It was impractical in that previous approach to ask the students to conduct a failure modes analysis because each project was very different and there was no common language with which to introduce the concept. With the inclusion of MBSE concept, the formal structure of the models provides that common language. The plan is to ask the students to conduct a failure modes and effects analysis on one physical component of the design that has a higher probability of failure. This failure mode's effects can be related back to the stakeholder features.

5.0 Observations and Conclusions

This first year of applying the MBSE Competencies in ECE Senior Design has been focused on developing meaningful assignments and faculty expertise. Qualitative evaluations of the student work were used to provide meaningful feedback that is being used to refine the assignments and produce more definitive rubrics for assessment. The use of MBSE Competencies provides a direct means for students to address a key aspect of Senior Design's intended outcome—the

structuring of problem space and design from an (intentionally) unstructured starting point. This structure forced the students to interact with their client more at the beginning of the project than they have in previous years. The formal definitions and structure within the models gave them a focus for their discussions with the clients that they didn't have in previous years. The thought required to create the system models forced the students to have a much better understanding of their project before they began producing the physical solutions. As such they were less quick to hurry into a physical design that would lead to an insufficient solution to the problem.

Because the required submissions (which are about different technical projects and for different faculty members) are based upon a common set of models, the students and faculty have an improved objective basis (a common underlying language and conceptual space) for comparison of their work—both to each other and to model-based rubrics. As a result of this common structure, student teams supervised by different faculty with widely differing experience nevertheless experience a more common expectation and supervision from those faculty.

Faculty members have discovered a high value in independently constructing their own personal practice models within this common framework. This has not only given insight into the students' experience, but also provides evidence to students of the different solutions that can be produced when modeling common systems. This experience suggests the value of training any faculty who are new to supervising student teams. The training should require the faculty to produce a set of system models and requirements under the supervision of other experienced faculty and evaluation of previous student work in order to better understand how to help the students. The training exercise does not need to be extensive, and could be done in just a couple of days before classes begin. A possible future outcome may be to develop online resources and materials for faculty at other institutions and in other disciplines.

One of the most important conclusions from this year is that introducing systems competencies with only two lectures at the beginning of the year is insufficient to familiarize the students with MBSE Competencies. The students really need an extended course that covers these concepts with an emphasis on practical application. Fortunately, the curriculum already contains a Junior-level design course ECE362, for which the department has been in much debate about what to present in this course. After the experience gained in this year, ECE362 will be redesigned to present the MBSE Competencies of stakeholder, domain/logical architecture, system modes, and interactions models. Students will practice these competencies by generating models for several different simples systems for which they are very familiar. By going through more familiar constrained systems, the students will be able to focus their attention on learning the model-based implementations of systems competencies rather than the domain-specific knowledge required to understand the project. The models that were generated by the faculty supervisors during their weekly meetings as discussed in Section 3 will be used as examples during the lectures. It is expected that students will be able to carry these experiences through senior design

and be much more adept at generating system models. The senior design sequence can then focus on teaching students how to work with more unconstrained problems for which they are less familiar. These aptitudes are addressed by the Discovery Competencies as part of the innovation process[21].

Another important result of this effort is the development of meaningful quantitative rubrics that are based on those shown in Appendix A. The structure of the models means that certain lists can be checked and verified against each other so that some rubrics can be binary in nature. For example, either all of the actors are accounted for in the interactions or they are not. The binary nature of such rubrics makes it easier for students to apply the rubrics to their own work and their simplicity makes it possible to generate rubrics to cover all mechanical details of creating the models. These binary rubrics would serve as a guide for reflection and improve the quality of the system models. As a result, faculty should be able to focus on those aspects of models that require greater experience to assess such as completeness and feasibility. The foundational rubrics shown in Appendix A are being used to develop a core rubrics set for the course. The authors plan to apply these rubrics at the end of the academic year and include the data in the presentation. Next year, the assessment of student assignments from this year will be compared to the students who will have been exposed to MBSE Competencies in ECE362 in order to observe the impact of that additional training and the benefit of having students apply rubrics to their own work during the design process.

6.0 Bibliography

1. Van Fraassen, B. C., Scientific Representation: Paradoxes of Perspective, Oxford U Press, 2008.15288
2. ISO/IEC 15288: 2002. Systems engineering – System life cycle processes. Geneva: International Organization for Standardization. ISO/IEC 15288: Systems Engineering—System Life Cycle Processes. International Standards Organization (2008).
3. Haskins, Cecilia, ed., INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Version 3.2, International Council on Systems Engineering (2010). 15288
4. NASA Systems Engineering Handbook, NASA/SP-2007-6105, Rev 1, U.S. National Aeronautics and Space Administration (2007).
5. International Council on Systems Engineering (INCOSE) web site: www.incose.org
6. Estafan, J. 2008. Survey of model-based systems engineering (MBSE) methodologies, INCOSE MBSE Initiative.
7. Model-Based Systems Engineering (MBSE) wiki of INCOSE and OMG: http://www.omgwiki.org/MBSE/doku.php
8. Schindel, William D., "System Engineering: An Overview of Complexity's Impact", SAE International, Technical Paper 962177, October, 1996.
9. Schindel, W. 1997. The tower of Babel: Language and meaning in system engineering. Technical Report No. 973217 SAE International.

10. Schindel, W., and Smith, V., "Results of Applying a Families-of-Systems Approach to Systems Engineering of Product Line Families", SAE International, Technical Report 2002-01-3086, November, 2002.

11. Schindel, W. "Requirements statements are transfer functions: An insight from model-based systems engineering", Proceedings of INCOSE 2005 Symposium, July, 2005.

12. Schindel, W. "Pattern-Based Systems Engineering: An Extension of Model-Based SE", INCOSE IS2005 Tutorial TIES 4.

13. Schindel, W. "Feelings and physics: Emotional, psychological, and other soft human requirements, by model-based systems engineering", Proceedings of INCOSE 2006 Symposium, July, 2006.

14. W. Schindel, "Failure Analysis: Insights from Model-Based Systems Engineering", Proceedings of INCOSE 2010 Symposium, July 2010.

15. Schindel, W., and Gunyon, R., "Engineering Global Pharmaceutical Manufacturing Systems In The New Environment", Proceedings of the INCOSE 2010 Symposium, July, 2010.

16. Bradley, J., Hughes, M. and Schindel, W. "Optimizing Delivery of Global Pharmaceutical Packaging Solutions, Using Systems Engineering Patterns" Proceedings of the INCOSE 2010 Symposium, July, 2010.

17. Schindel, W. 2011. Systems engineering for advanced manufacturing: Unit op insights from model-based methods. To appear in Proceedings of the INCOSE 2011 International Symposium.

18. W. Schindel, "What Is the Smallest Model of a System?", Proc. of the INCOSE 2011 International Symposium, International Council on Systems Engineering (2011).

19. Buede, Dennis M., The Engineering Design of Systems: Models and Methods. Second edition. John Wiley & Sons, Inc., New York, 2000.

20. S. Friedenthal, A Practical Guide to SysML: The Systems Modeling Language, Morgan Kaufmann; 2 edition (October 31, 2011)

21. Schindel, William D., W., Peffers, Samuel N., Hanson, James H., Ahmed, Jameel, Kline, William A., "All Innovation is Innovation of Systems: An Integrated 3-D Model of Innovation Competencies", Proc. of 2011 Conference of the ASEE, June, 20

22. W. Schindel, "Systems of Innovation II: The Emergence of Purpose", Proceedings of INCOSE 2013 International Symposium (2013).

Appendix A: Model-Based Rubrics for the System Competencies

A strength of the System Competencies is that their practice may be directly demonstrated using explicit system "models" — technical artifacts that are explicit data structures of a graphical or tabular nature. Although these may be expressed using various specific system modeling languages such as SysML™ and IDEF, the System Competencies and rubrics below are independent of the use or choice of modeling language. The following table expresses typical rubrics that may be applied to the System Competencies:

S1: Describing the target of innovation from a systems perspective	
S1R1	Domain diagram and definitions are available, showing that the subject system is understood to be itself part of a larger system.
S1R2	What percent of external domain actors are identified by the domain model?
S1R3	Logical architecture diagram is available, showing that the subject system' external behavior is understood to emerge from the interactions of a set of decomposed subsystems.
S1R4	What percent of the subject system's external behavior is covered by the logical subsystems / logical architecture model?
S2: Applying a system stakeholder view of value, trade-offs, and optimization	
S2R1	A stakeholder model is available, identifying and defining the classes of stakeholder with a stake in the subject system.
S2R2	What percent of the total set of classes of stakeholders in this system are represented by the stakeholder model?
S2R3	A system feature model is available, identifying and defining, in stakeholder terminology, the aspects of system behavior that carry stakeholder impact, positive or negative.
S2R4	A stakeholder-feature association trace is available, showing which features are of interest to each stakeholder class.
S2R5	To the extent that the interests of stakeholders are quantified or further identified by parameters, the features have defined feature attributes identifying and defining those variables.
S2R6	What percent of the total set of stakeholder interests are covered by the features and their attributes?
S2R7	Stakeholder features are explicitly traced to the system external interactions that deliver on or have stakeholder-impacting aspects.
S2R8	What percent of the features are fully covered by the interactions associated with them?
S2R9	Does the design solution selection rationale demonstrate optimization with respect to the (possibly weighted) stakeholder feature value space?
S3: Understanding system's interactions and states (modes)	
S3R1	An interaction model is available identifying and defining the different physical interactions the system has with its environment over its life cycle.
S3R2	A state model is available, identifying and defining the different modes of externally visible system behavior in interacting with its environment over its life cycle, including state definitions and association with external interactions.
S3R3	What percent of the total set of interactions the system has with its environment are included in the system interactions model?
S3R4	What percent of the total set of system states or modes are included in the system state model?
S4: Specifying system technical requirements	

S4R1	The externally visible behavior of the system, interacting with its environment, is fully specified by system requirements statements associated with each modeled interaction.
S4R2	System external interactions are individually modeled by interaction diagrams showing the related system input-output exchanges with external actors.
S4R3	For each modeled interaction, a set of associated system requirement statements is provided that are objective, testable, atomic, descriptions of the required system input-output behavioral relationships.
S4R4	Key attributes (parameters) further characterizing the requirements are included with the system requirements.
S4R5	Attribute value dependency couplings of requirements attributes and feature attributes are identified and characterized, showing how stakeholder feature satisfaction varies with respect to change in technical requirement attribute values.
S5: Creating and analyzing high level design	
S5R1	A physical architecture model is provided, identifying and defining physical subsystems or components and their arrangement into physical relationships with each other.
S5R2	System black box requirements are traceably decomposed to white box requirements that are objective, testable, atomic descriptions of internal functional roles.
S5R3	The decomposed white box requirements are explicitly allocated to the components of the physical architecture which are responsible for meeting those requirements.
S5R4	Key attributes (parameters) of the physical architecture are identified and defined.
S5R5	Attribute value dependency couplings of physical architecture attributes and requirements attributes are identified and characterized, showing how system behavior varies with respect to change in physical component attribute values.
S6: Assessing solution feasibility, consistency, and completeness	
S6R1	Based upon a review of the modeled design, would the decomposed white box requirements, if met, satisfy the parent black box requirements?
S6R2	Based on a review of the modeled design, are the physical subsystems or components capable of meeting the white box requirements that have been allocated to them?
S6R3	Based on a review of the modeled design, have design margins or gaps for each of the requirements attributes been identified?
S6R4	Have any additional parasitic behaviors of the selected physical components or subsystems been identified and included in the model?
S6R5	If fabricated, assembled, integrated, or otherwise constructed, is the implemented system solution consistent with the modeled system design?
S6R6	If fabricated, assembled, integrated, or otherwise constructed, does the implemented system solution meet the modeled system requirements?
S7: Performing system failure mode and risk analysis	
S7R1	Have the impact effects of not delivering each of the stakeholder features been identified, including the severity of those impacts?

S7R2	Have the counter-requirements associated with each of the modeled system black box requirements been identified for use in risk analysis?
S7R3	Have the failure modes of the design components or subsystems been identified?
S7R4	Have the failure modes associated with external human actors been identified?
S7R5	Have the failure modes associated with external processes been identified?
S7R6	Have the failure modes been associated with related counter requirements?
S7R7	Have the failure modes been associated with probabilities of their occurrence?
S7R8	Have the counter requirements been associated with the related impact effects?
S7R9	Have the relative risks, based on probability and severity, been estimated?
S7R10	Have detection and mitigation strategies for the failure modes and effects been described?
S8: Planning system families, platforms, and product lines	
S8R1	Has the range of configured stakeholder configurations to be satisfied been modeled?
S8R2	Have product line configuration rules for system features been modeled?
S8R3	Has configuration of external system environments across different configurations been modeled?
S8R4	Has configuration of the system state model for different configured features been modeled?
S8R5	Has variation in configured interactions with respect to configured features been modeled?
S8R6	Has variation in system requirements and their attributes been modeled across the range of configurations?
S8R7	Have interfaces been modeled to minimize impact across varying configurations?
S8R8	Have variant product lines, archetypes, and sub-families been identified to globally optimize Return on Variation across the system family, platform, or product line?
S9: Understanding roles & interdependencies across the innovation process	
S9R1	Are the roles and interdependencies of the team members responsible for different aspects of the innovation process identified, described, understood, and agreed upon?
S9R2	Is the innovation process, and its allocation to different organizations, partners, team members, and information systems described as a modeled system?
S9R3	Are the goals of the innovation process identified, used to configure instances of the process, and known to the organization?