ASEE'S VIRTUAL CONFERENCE
At Home with Engineering Education JUNE 22 - 26, 2020 #ASEEVC
Paper ID #29427

Comparing Students' Solutions to an Open-ended Problem in an Introductory Programming Course with and without Explicit Modeling Interventions

Dr. Kelsey Joy Rodgers, Embry-Riddle Aeronautical University - Daytona Beach

Kelsey Rodgers is an Assistant Professor in the Engineering Fundamentals Department at Embry-Riddle Aeronautical University. She teaches a MATLAB programming course to mostly first-year engineering students. She primarily investigates how students develop mathematical models and simulations and effective feedback. She graduated from the School of Engineering Education at Purdue University with a doctorate in engineering education. She previously conducted research in Purdue University's First-Year Engineering Program with the Network for Nanotechnology (NCN) Educational Research team, the Model-Eliciting Activities (MEAs) Educational Research team, and a few fellow STEM education graduates for an obtained Discovery, Engagement, and Learning (DEAL) grant. Prior to attending Purdue University, she graduated from Arizona State University with her B.S.E. in Engineering from the College of Technology and Innovation, where she worked on a team conducting research on how students learn LabVIEW through Disassemble, Analyze, Assemble (DAA) activities.

Dr. Matthew A. Verleger, Embry-Riddle Aeronautical University - Daytona Beach

Matthew Verleger is an Associate Professor of Engineering Fundamentals at Embry-Riddle Aeronautical University in Daytona Beach, Florida. His research interests are focused on using action research methodologies to develop immediate, measurable improvements in classroom instruction and on the development of software tools to enhance engineering education. Dr. Verleger is an active member of ASEE, having served as the founding chair of the Student Division, a Program Chair and a Director for the Educational Research and Methods Division, and the General Chair of the First-Year Division's First-Year Engineering Experience Conference.

Dr. Farshid Marbouti, San Jose State University

Farshid Marbouti is an Assistant Professor of General (interdisciplinary) Engineering at San Jose State University (SJSU). He is currently the chair of SJSU Senate Student Success Committee. Farshid completed his Ph.D. in Engineering Education at Purdue University. His research interests center on First-Year Engineering student success and engineering design.

Comparing Students' Solutions to an Open-ended Problem in an Introductory Programming Course with and without Explicit Modeling Interventions

Abstract

Engineers must understand how to build, apply, and adapt various types of models in order to be successful. Throughout undergraduate engineering education, modeling is fundamental for many core concepts, though it is rarely explicitly taught. There are many benefits to explicitly teaching modeling, particularly in the first years of an engineering program. The research questions that drove this study are: (1) How do students' solutions to a complex, open-ended problem (both written and coded solutions) develop over the course of multiple submissions? and (2) How do these developments compare across groups of students that did and did not participate in a course centered around modeling?. Students' solutions to an open-ended problem across multiple sections of an introductory programming course were explored. These sections were all divided across two groups: (1) experimental group - these sections discussed and utilized mathematical and computational models explicitly throughout the course, and (2) comparison group - these sections focused on developing algorithms and writing code with a more traditional approach. All sections required students to complete a common open-ended problem that consisted of two versions of the problem (the first version with smaller data set and the other a larger data set). Each version had two submissions – (1) a mathematical model or algorithm (i.e. students' written solution potentially with tables and figures) and (2) a computational model or program (i.e. students' MATLAB code). The students' solutions were graded by student graders after completing two required training sessions that consisted of assessing multiple sample student solutions using the rubrics to ensure consistency across grading. The resulting assessments of students' works based on the rubrics were analyzed to identify patterns students' submissions and comparisons across sections. The results identified differences existing in the mathematical and computational model development between students from the experimental and comparison groups. The students in the experimental group were able to better address the complexity of the problem. Most groups demonstrated similar levels and types of change across the submissions for the other dimensions related to the purpose of model components, addressing the users' anticipated needs, and communicating their solutions. These findings help inform other researchers and instructors how to help students develop mathematical and computational modeling skills, especially in a programming course. This work is part of a larger NSF study about the impact of varying levels of modeling interventions related to different types of models on students' awareness of different types of models and their applications, as well as their ability to apply and develop different types of models.

Introduction

Engineers must understand how to build, apply, and adapt various types of models, including mathematical and computational models, to be successful. Modeling is fundamental for many core undergraduate engineering courses, though it is rarely explicitly taught [1]. There are many benefits to teaching modeling or using modeling projects, particularly in the first years of the engineering curriculum [1-3]. There are some well-developed pedagogies that demonstrate the successes of doing this. Model-eliciting activities (MEAs) are an impactful example of a pedagogical approach used in first-year engineering to teach mathematical modeling skills [3]. Even though there are some established approaches, there is still a need for more meaningful ways to teach modeling throughout the engineering curricula and especially in first-year engineering courses [1].

Developing computational thinking skills is something that has been emphasized in engineering education more recently and aligns with this call for curriculum that explicitly teaches mathematical and computational modeling skills [4]. Computational thinking skills involve abstraction, analysis, automation, and modeling [5]. Again both the objectives of developing students' computational thinking skills and explicitly teaching modeling present similar underlying goals.

Many modeling interventions have been proven effective based on research conducted within the Computational Adaptive Expertise (CADEX) [2] and Models and Modeling Perspective (M&MP) [6] frameworks. Carberry, McKenna, Linsenmeier, and Cole [7] conducted research within the CADEX framework and found that a modeling intervention caused a significant shift in senior engineering students' concepts of models, including discussing more predictive models. Carberry and McKenna [1] expanded their research within the CADEX framework to gain a greater understanding of modeling conceptions and found more students discussed mathematical and predictive models when they were taught an explicit mathematical modeling module. Research efforts within the M&MP have focused around a mathematical modeling building intervention called MEAs [3]. Some of this research has focused on how students develop mathematical model solutions to MEAs (e.g., [8, 9]) and how MEAs are incorporated in engineering courses (e.g., [10, 11]). Also this research has been conducted in various settings including research in large first-year engineering courses (e.g., [12]) and upper division engineering courses (e.g., [13, 14]). There is also some research about a modeling application intervention called model-adaptation activities (MAAs) within the M&MP framework [15].

Models are fundamentally systems based on real-world systems [16]. There are various types of models used in engineering. Physical models (e.g., prototypes, drawings) are the most common type of models that engineering students identify as a type of model and discuss as a model used in the design process [1, 17]. Although physical models are one type of model that engineers use, there are other critical types of modeling, including mathematical models [3, 6, 18] and computational models [19, 20]. Within engineering curricula, various types of models need to be explicitly introduced to enhance students' definitional knowledge and implemented to develop students' modeling skills. Interventions need to focus on types of models, model application, and model development in engineering curricula, especially on mathematical and computational models [1, 3, 15, 19].

Mathematical models and computational models were the two types of models focused on in this study. Mathematical models utilize mathematics to quantify and explain a real-world system [16]. As aforementioned, MEAs are a pedagogical approach to help students develop mathematical modeling skills [18]. MEAs are a mathematical modeling problem that require students to create a model to meet particular criteria and constraints for a stakeholder based on provided, relevant data, where there are multiple good possible solutions [3]. Computational models are based on mathematical models and utilize computer programs to enable users to interact with the underlying model [15-16]. A specific type of computational model are simulations, which require a visualization of outputs in addition to the manipulable user inputs (e.g., [15, 21-23]). Simulations are more commonly being implemented in courses as a tool to enable more exploration of concepts (e.g., [19]). When tools are implemented in courses, it is not as common to see an emphasis on computational models nor the development of these tools. Rodgers [15] discussed the use of MAAs to challenge students to apply models in computer programs (specifically MATLAB). The structure presented by Rodgers [15] of having students develop a simulation based on a mathematical model built through a MEA motivated the design of the problems utilized in this study.

This study focused on embedding a modeling problem in a programming course. Programming courses typically focus primarily on syntax, but there is a need to teach students how to effectively develop an algorithmic solution to create good programs [24, 25]. Most engineering education studies on computer programming focus on other pedagogical approaches, such as paired programming (e.g., [26]) or extreme programming (XP) (e.g., [27]). This study focused specifically on modeling development in a first-year programming course and built off a similar previous study [28].

Rodgers et al. [28] found that students participating in a programming course explicitly teaching mathematical and computational modeling consistently improved their demonstrated modeling abilities across four submissions of a modeling problem. In the same study, students in the same programming course that did not explicitly teach modeling did not demonstrate consistent improvement [28]. Based on these findings, this study investigated similar research questions in a later semester with similar conditions.

Research Purpose and Questions

The research questions that drove this study are: (1) How do students' solutions to a complex, open-ended problem (both written and coded solutions) develop over the course of multiple submissions? and (2) How do these developments compare across groups of students that did and did not participate in a course centered around modeling?.

Methods

Setting and Participants

In Spring 2019, 348 students enrolled in an introductory computer programming course for engineers across 16 sections at a medium-sized, private, STEM+Business university. MATLAB is the programming language for the course. All mechanical, civil, and aerospace engineering students are required to take this course. The electrical and computer engineering students take a similar course that uses Java as the programming language. This course is open to other students at the university and required by some non-engineering degree programs, such as the astronomy and astrophysics program and the unmanned aircraft systems (UAS) science degree. Even with these additional requirements, sections are made up of mostly engineering students and more specifically aerospace engineering students. The focus of the course is to teach engineering students how to develop effective computer programs for solving engineering problems. The learning objectives of the course are to:

1. Demonstrate understanding of the role of software design when solving problems using the computer.
2. Apply knowledge of mathematics and computer programming to communicate ideas when solving computational problems.
3. Design and implement algorithmic solutions to problems requiring user input/output (I/O), data processing, control structures, arrays, and file input/output (I/O).
4. Solve problems of intermediate complexity requiring the use of non-numerical data such as characters and strings.
5. Apply a top-down design methodology to problems of intermediate complexity using functions.

Two of the 16 sections were taught by one instructor that explicitly incorporated modeling throughout the course materials by developing various levels of modeling problems for all the homework assignments and in-class activities. The revised course was designed to scaffold students from solving more close-ended modeling problems to more open-ended modeling problems across the semester. Students were presented different approaches to solve problems throughout the course. They were challenged to see how there can be multiple good approaches and solutions depending on the problem. The language of the class focused on mathematical and computational models, in addition to the more common computing language of algorithms and computer programs. There was a total of 47 students enrolled in these two sections. The other 14 sections taught by four other instructors were not significantly modified from previous iterations of the course. There was a total of 301 students enrolled in these 14 sections. These sections focused more on syntax and used close-ended problems with one correct answer for most of their assignments. Most sections use a seven step problem solving process in the class (demonstrated in the modeling problem shown in Appendix A). One step of this process prompts the students to identify assumptions they are making that simplify the problem in developing their solution.

The students from all sections of this course were required to complete one common modeling problem with two different versions individually. Each version consisted of two submissions – the first submissions were students' written solutions potentially with tables, figures, and flowcharts (i.e. their mathematical model) and the second submission were students' MATLAB code with comments, as specified (i.e. their computational model). The four submissions are summarized in Table 1. One important change in this implementation in comparison to the previous semester (Fall 2018 – [28]) is students were required to incorporate comments throughout their code, whereas before they could attach their previously submitted written model. The problem explicitly stated, "You are required to ***comment*** your assumptions (for Step 4) and all your justifications for each step throughout your code" (see Appendix A).

Table 1: Description of the four submissions for the modeling problem

Submission	**Modeling Problem Assigned**	**Submission Type**
Submission 1	Version 1 (table of data)	Written solution
Submission 2	Version 1 (table of data)	Coded solution (.m file)
Submission 3	Version 2 (Excel file with a larger data set)	Written solution
Submission 4	Version 2 (Excel file with a larger data set)	Coded solution (.m file)

The modeling problem was developed using the six design principles of the M&MP theoretical framework [6]. The second version of the problem contained more data than the first version to prompt the students to reevaluate their solutions based on successful implementations of MEAs [3]. The intention of the second version was also to ensure more iterations in their model development process and capture how their demonstrated modeling skills changed across the course. The written submission for the problem was similar to a MEA [3] in that the students had to interpret a problem and data to develop a model. The coded submission followed the concept of a MAA [6, 15] in that the students had to apply the model they developed through the first submission.

The modeling problem challenged students to develop a model to rank possible wind farm locations from best to worst based on historical wind data and a relevant graph about power output vs. wind speed for an alternative engineering company. The first version of the modeling problem is provided in Appendix A. In the first version, the students were provided a table summarizing a year of wind data for five different locations. In the second version of the problem, the students were provided an Excel file of data that contained the minimum, average, and maximum windspeed for every day of the year of 2018 for seven different locations. The larger data set for the second version contained the raw data for the original five locations and two additional locations.

Data Collection

The 46 students in the two sections taught by the one professor with the revised modeling curriculum make up the experimental group for this study. Out of the 14 sections taught by the four instructors (i.e. the comparison group), only two instructors both implemented the assignment consistent with the way it was developed and ensured the assessment of the assignment was consistent. One instructor taught four sections with 74 students (Comparison Group – Instructor 1) and the other instructor taught four sections with 85 students (Comparison Group – Instructor 2). The data collected for this study were the students submitted solutions for all four of the modeling problem submissions, summarized in Table 1. Out of the 205 students across the 10 sections, 192 students completed at least one submission, summarized in Table 2. All of these students submissions were included in this study.

Table 2: Summary of the number of student submissions per group

Group and instructor	Total number of students in the group	Students submitted at least one submission
Experimental Group	46	45 (97.8%)
Comparison Group – Instructor 1	74	69 (93.2%)
Comparison Group – Instructor 2	85	78 (91.8%)

Data Analysis

To assess the quality of students' models in each of the submissions, a rubric was developed. The rubric consisted mostly of items based on the four dimensions established for assessing students' solutions to MEAs [8, 29]. The other additional rubric items were included to assess the quality of students' code for the coding submissions. The already established dimensions are mathematical model complexity, modifiability, reusability, and shareability [8, 29]. The purpose of the mathematical model complexity dimension is to assess students' ability to address the complexity of the problem in an elegant solution [8, 29]. Items related to this category consisted of meaningful utilization of the provided data and relevant facts, as well as proper conversion of units. There was a total of five items developed within this category. The purpose of the modifiability dimension is to ensure a student's model is generalizable by evaluating the student's rationale for each step in their model [8, 29]. There was a total of five items created that focused on students' rationale for five major components of their models. The purpose of the reusability dimension is to assess students' understandings of the given problem and their client or user (i.e., problem scoping) [8, 29]. There was one item created that focused on students' ability to identify assumptions they are making in solving the problem. The purpose of the shareability dimension is to evaluate students' ability to communicate their solution [8, 29]. There were two items that assessed this dimensions; these items related to how the students communicated the written model and the provided sample results. These 13 rubric items were used to assess students' models in all four submissions.

The entire rubric used for the fourth submission is provided in Appendix B. All the scores for the rubric items shown in the appendix are based on the fourth submission scoring used for assigning students' grades. For the data analysis all the rubric items for were assigned a possible score based on how many levels there were for each item. Every rubric item consisted of either two, three, or four options for each grade with points and options as summarized in Table 3.

Table 3: Description of the four submissions for the modeling problem

Number of Options for Rubric Items	**Submission Scores/Levels**
2 options (levels)	demonstrated – 1 point; or not – 0 points
3 options (levels)	fully – 2 points; partially – 1 point; or not demonstrated – 0 points
4 options (levels)	fully – 3 points; some – 2 points; less – 1 point; or not demonstrated – 0 points

In the development of this rubric, reflection on the previous implementation of a similar problem were considered – findings discussed by Rodgers et al. [28]. The two biggest changes were: (1) rubric items related to the shareability dimension were incorporated in and (2) some rubric items had more levels rather than having as many dichotomous rubric items. The first change was to add another dimension of analysis in the study. The second change was primarily based on the different context of the problem aligned better with more levels for some rubric items.

The students' solutions to the modeling problem included in this study were graded by two student graders. The graders completed a required training that consisted of assessing multiple sample student solutions using the rubrics to ensure consistency across grading. There were two different trainings – one for each version of the modeling problem. In this training, the graders practiced applying the rubric for both the model and code submissions after a detailed explanation of each item.

The resulting assessments of students' works based on the 13 modeling items of the rubric was analyzed using descriptive statistics and t-tests, when fitting [30]. The t-tests were used to determine statistically significant differences between the first and last submission within each group, where further investigation was of interest.

Findings

Mathematical Model Complexity – Comparing Groups/Instructors

The average scores for all the mathematical model complexity rubric items across all four submissions for all three instructors (the one teaching students in the experimental group and the other two teaching students in the comparison group) are presented in Figure 1. Along the mathematical model complexity dimension, the students across all sections improved from the first written model (submission 1) to second written model (submission 3) and from the first coded solution (submission 2) to second coded solution (submission 4). Only the average scores for the students in the experimental group improved from the first submission (first written model) to the last submission (second coded solution).

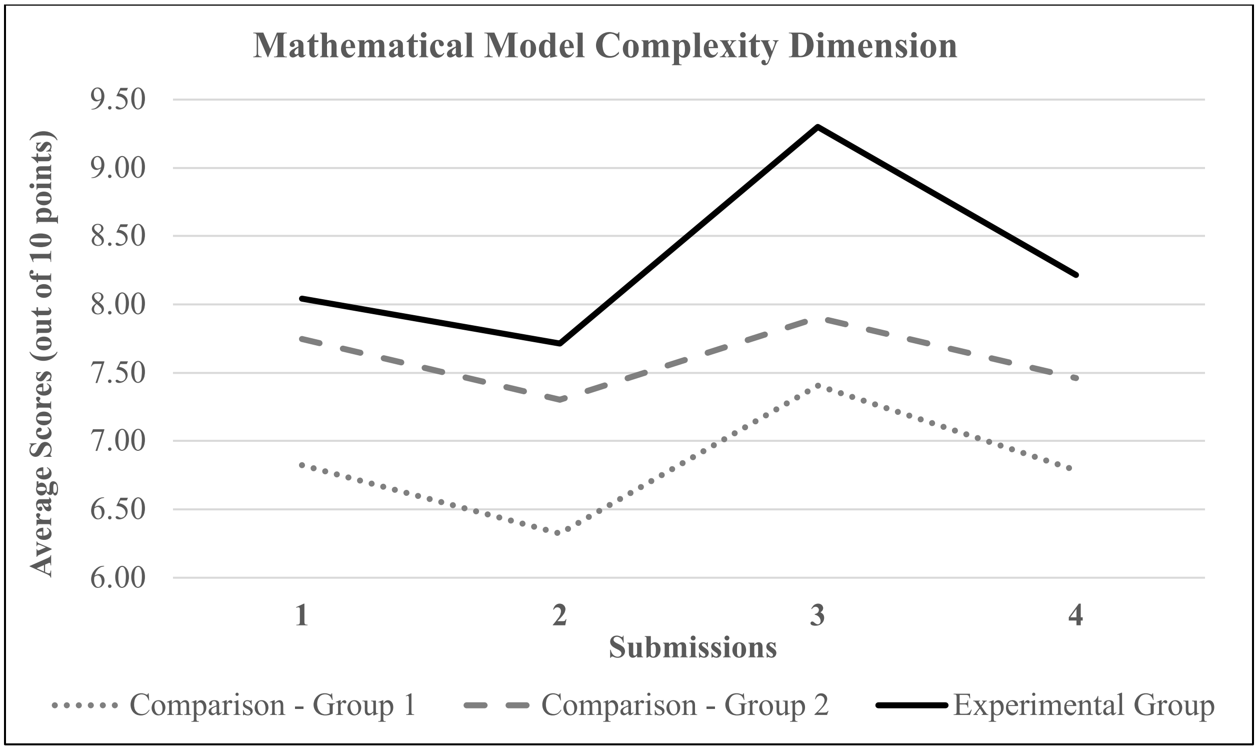

Figure 1: Average of students' scores on Mathematical Model Complexity items

As demonstrated in Table 4, the only statistically significant change in the average scores was from submission 1 (first written model) to submission 3 (second written model) for the students from the experimental group. There were no other statistically significant changes at a 0.05 level of significance across the submissions. The next closest to statistically significant change was the change in the average scores from the first to the last coding submission (submissions 2 to 4) for the students from the experimental group.

Table 4: Mathematical model complexity – All instructors – Difference between submissions

Section	Submissions	p-value
Comparison Group - Instructor 1	1 (model v.1) to 2 (code v.1)	0.32
	1 (model v.1) to 3 (model v.2)	0.21
	1 (model v.1) to 4 (code v.2)	0.94
	2 (code v.1) to 4 (code v.2)	0.37
Comparison Group - Instructor 2	1 (model v.1) to 2 (code v.1)	0.39
	1 (model v.1) to 3 (model v.2)	0.70
	1 (model v.1) to 4 (code v.2)	0.51
	2 (code v.1) to 4 (code v.2)	0.84
Experimental Group	1 (model v.1) to 2 (code v.1)	0.35
	1 (model v.1) to 3 (model v.2)	**0.00***
	1 (model v.1) to 4 (code v.2)	0.31
	2 (code v.1) to 4 (code v.2)	0.10
* statistically significant difference between first and last submissions		

Figure 1 presents the change in students' average scores summed up across all the various rubric items. The following figures show the students' average scores for each of the five individual rubric items for the mathematical model complexity dimension.

The students' average scores for the utilizing data set rubric item are shown in Figure 2. The students from the experimental group started with a much higher average than the students from the two comparison groups. By the third submission, all the groups had more similar scores. The two comparison groups showed the most improvement across the submissions.

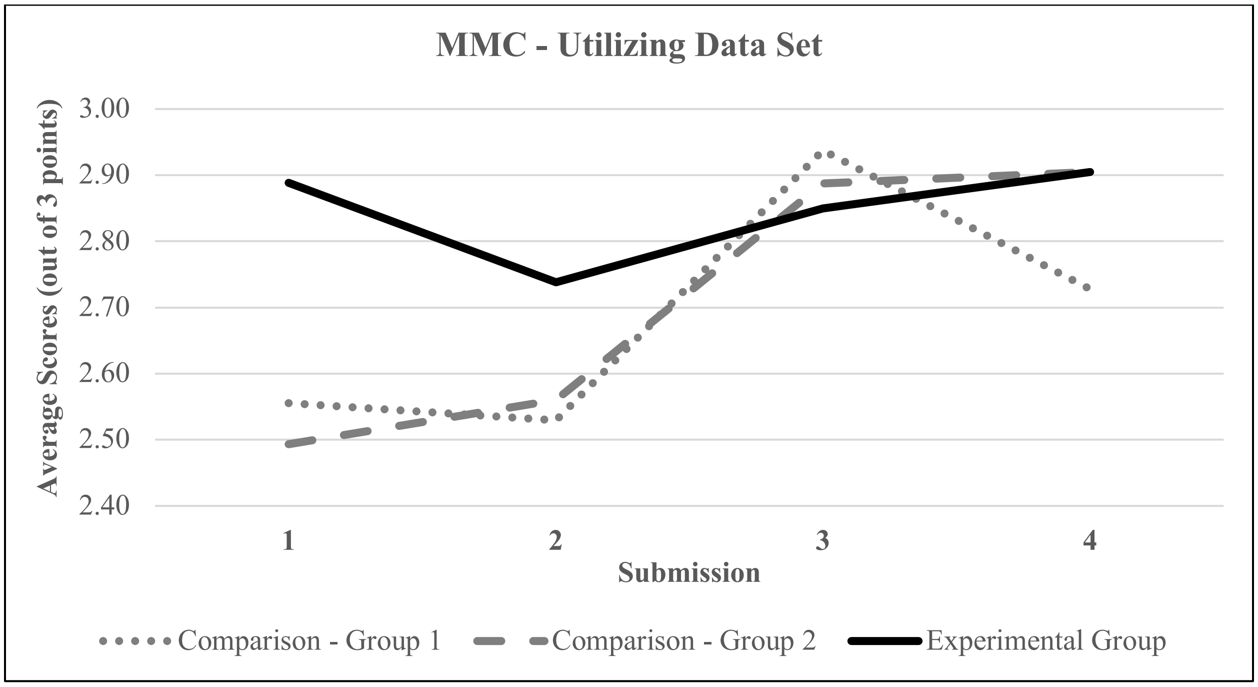

Figure 2: Average of students' score on a Mathematical Model Complexity item

The students' average scores for the two rubric items regarding students ability to ensure their solution works for a varying number of locations and implementation of appropriate unit conversions are presented in Figures 3 and 4, respectively.

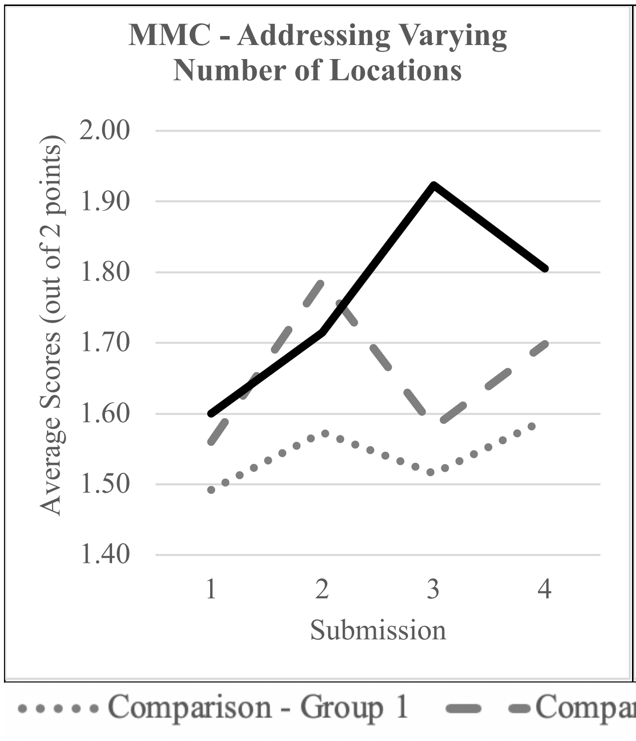

Figure 3: Average of students' scores on a Mathematical Model Complexity item

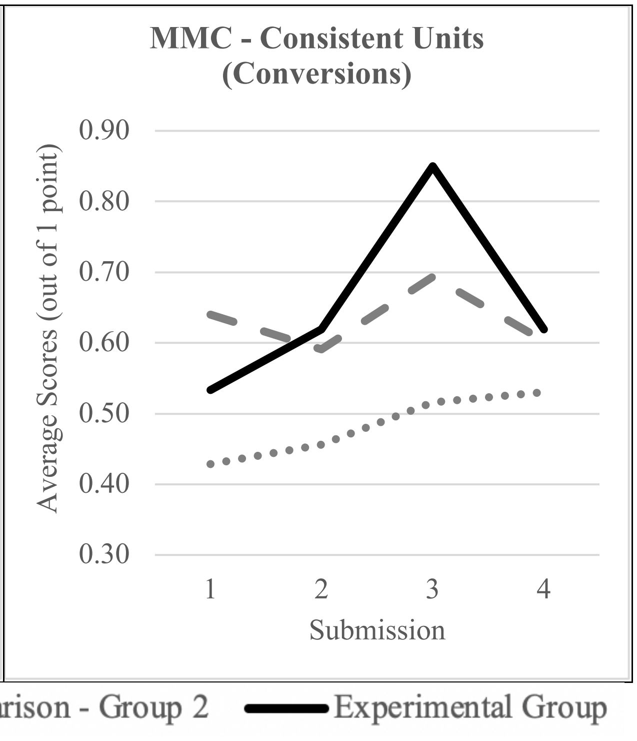

Figure 4: Average of students' scores on a Mathematical Model Complexity item

The students' average scores for the rubric items related to students implementing provided relevant theory about wind turbines are shown in Figures 3 and 4. The average scores for students from all three groups did much better implementing the relevant theory in their written models than their coded solutions.

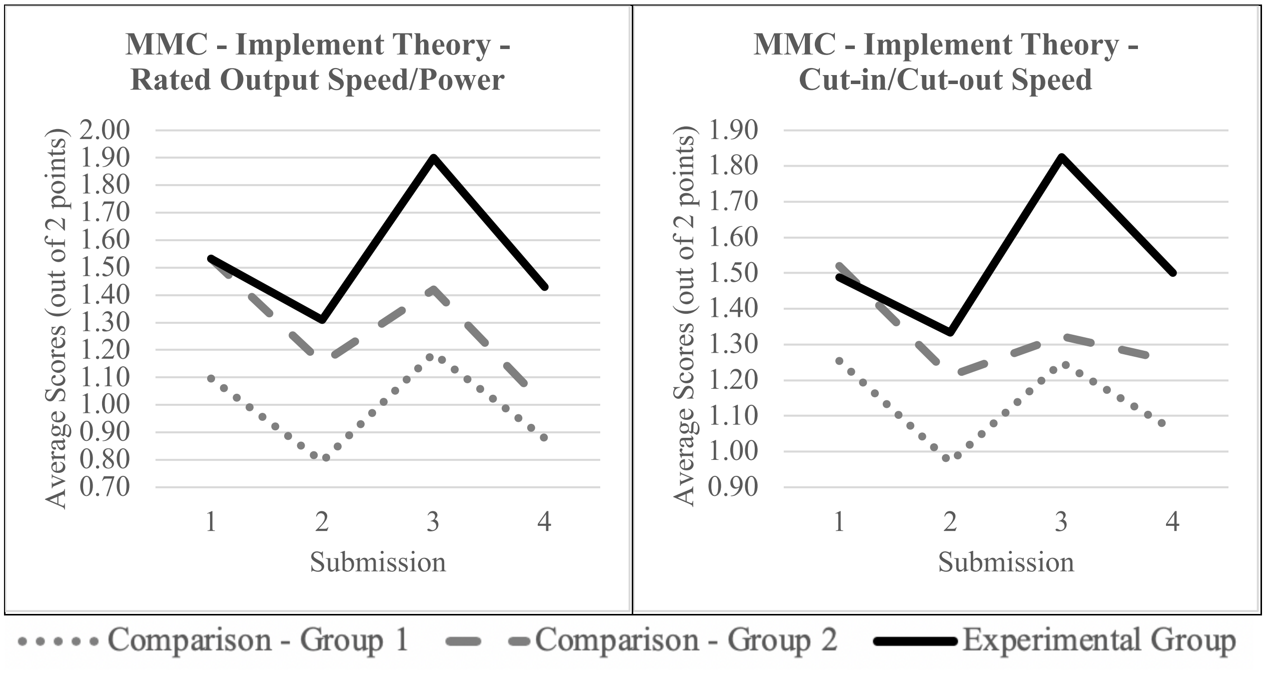

Figure 5: Average of students' scores on a Mathematical Model Complexity item

Figure 6: Average of students' scores on a Mathematical Model Complexity item

Modifiability – Comparing Groups/Instructors

The average scores for all the modifiability rubric items across all four submissions for the experimental group and two comparison groups are presented in Figure 7. All groups scored higher on the written submissions than the coding submissions. One of the comparison groups (Group 1) had the most consistent average scores between the written and coded solutions. The experimental group was the only group that had a later submission (submission 3) with a higher average score than the first submission.

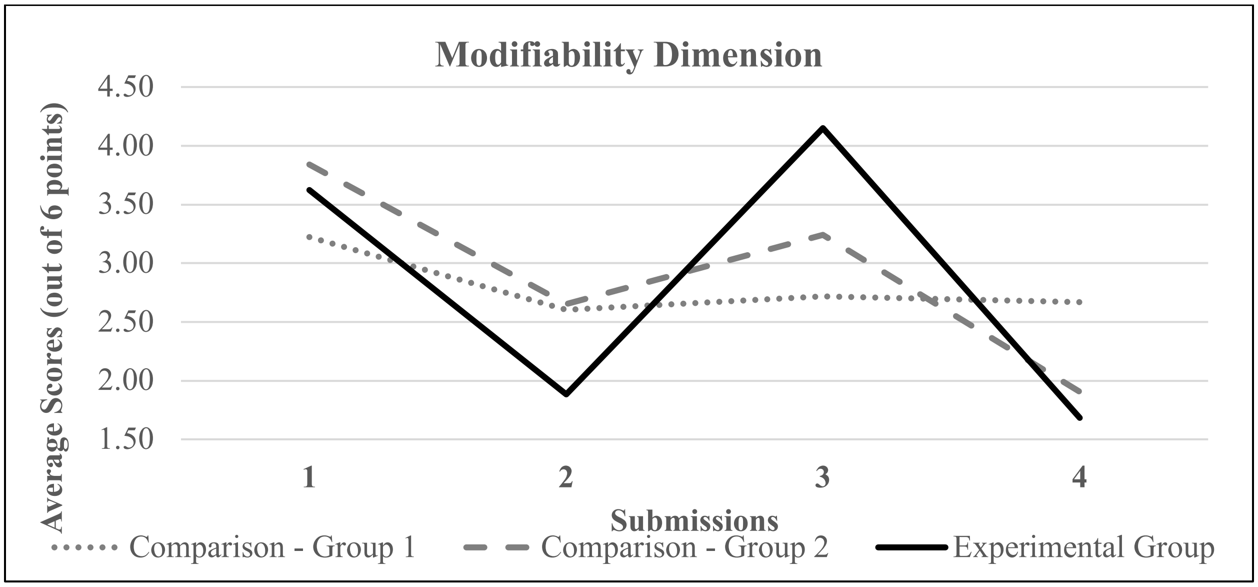

Figure 7: Average of students' scores on Modifiability items

As presented in Table 5, there was a statistically significant negative change from the first to the last submission across all three groups (both comparison groups and the experimental group). There was also a statistically significant negative change across the two written submissions for one of the comparison groups (Group 2). There were no other statistically significant changes at a 0.05 level of significance across the analyzed submissions.

Table 5: Modifiability – All instructors – Difference between submissions

Section	Submissions	p-value
Comparison Group - Instructor 1	1 (model v.1) to 3 (model v.2)	0.09
	1 (model v.1) to 4 (code v.2)	0.05*
Comparison Group - Instructor 2	1 (model v.1) to 3 (model v.2)	0.03*
	1 (model v.1) to 4 (code v.2)	0.00*
Experimental Group	1 (model v.1) to 3 (model v.2)	0.16
	1 (model v.1) to 4 (code v.2)	0.00*
* statistically significant difference between first and last submissions		

Reusability – Comparing Groups/Instructors

The average scores for the reusability rubric items across the submissions for the experimental group and two comparison groups are presented in Figure 8. The students had similar average scores across all the groups for both the written solutions. The experimental group had the highest scores across all the submissions, although it was extremely close to one of the comparison group's scores (Group 2) on the written submissions.

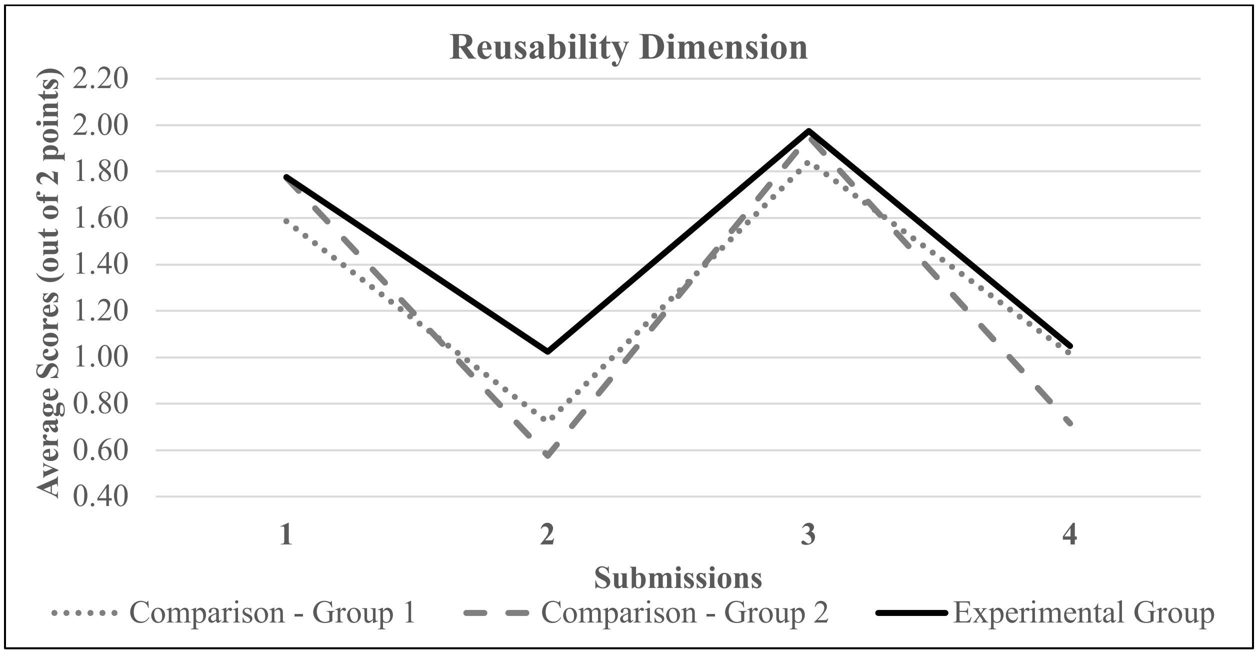

Figure 8: Average of students' scores on Reusability items

As shown in Table 6, there was a statistically significant negative change from the first to the last submission across all three groups (both comparison groups and the experimental group). On the other hand, there was also a statistically significant positive change across the two written submissions for all three groups.

Table 6: Reusability – All instructors – Difference between submissions

Section	Submissions	p-value
Comparison Group - Instructor 1	1 (model v.1) to 3 (model v.2)	0.02*
	1 (model v.1) to 4 (code v.2)	0.00*
Comparison Group - Instructor 2	1 (model v.1) to 3 (model v.2)	0.01*
	1 (model v.1) to 4 (code v.2)	0.00*
Experimental Group	1 (model v.1) to 3 (model v.2)	0.01*
	1 (model v.1) to 4 (code v.2)	0.00*
* statistically significant difference between first and last submissions		

Shareability – Comparing Groups/Instructors

The average scores for the shareability rubric items across the submissions for the experimental group and two comparison groups are presented in Figure 9. The experimental group had the highest average scores across all four submissions, but also had the largest negative change from the third to the fourth submission. One of the comparison groups (Group 1) received higher average scores on their coded solutions (submissions 2 and 4) than the written solutions (submissions 1 and 3).

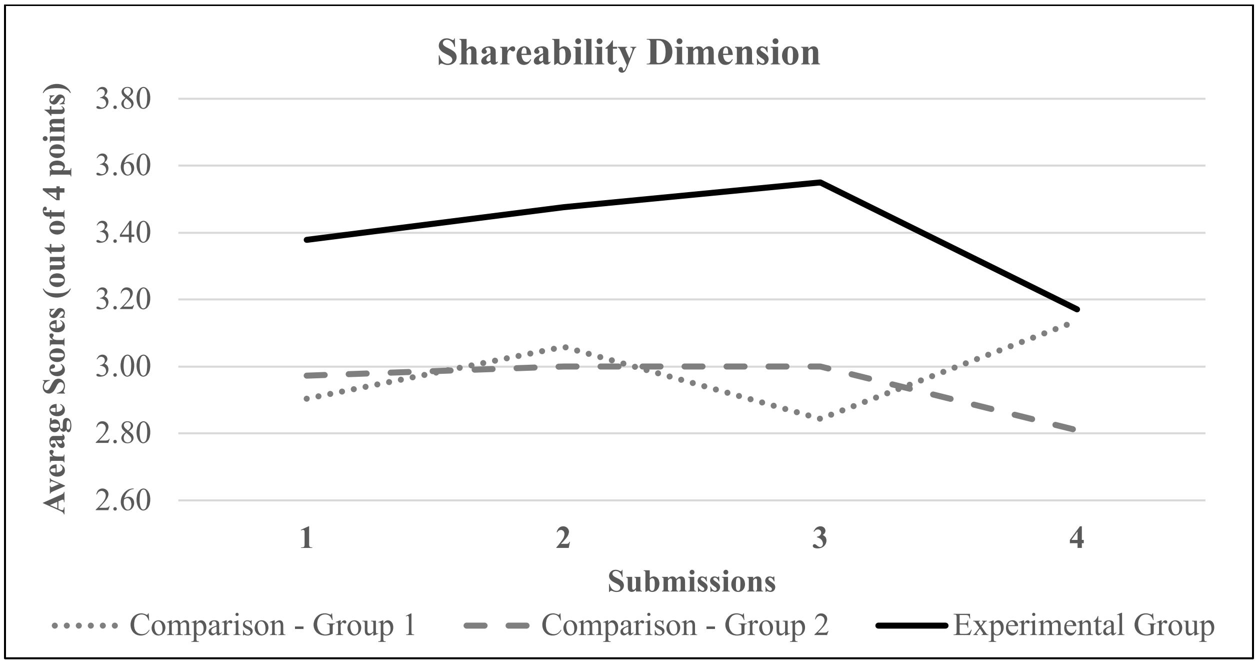

Figure 9: Average of students' scores on Shareability items

As demonstrated in Table 7, there were no statistically significant changes related to the shareability dimension at a 0.05 level of significance across the analyzed submissions.

Table 7: Shareability – All instructors – Difference between submissions

Section	**Submissions**	**p-value**
Comparison Group - Instructor 1	1 (model v.1) to 3 (model v.2)	0.78
	1 (model v.1) to 4 (code v.2)	0.26
Comparison Group - Instructor 2	1 (model v.1) to 3 (model v.2)	0.88
	1 (model v.1) to 4 (code v.2)	0.32
Experimental Group	1 (model v.1) to 3 (model v.2)	0.31
	1 (model v.1) to 4 (code v.2)	0.33
* statistically significant difference between first and last submissions		

Summary

As demonstrated in the figures above, the students from the experimental group were consistently receiving higher scores on the mathematical model complexity, reusability, and shareability dimensions than the comparison groups. The modifiability dimension was the only area the comparison groups typically outperformed the experimental group. Although the students across all three instructors had similar scores on the reusability dimension, especially on their written submissions. The comparison groups had fairly similar scores across most of the dimensions; the greatest consistent difference across the comparison groups was on the mathematical model complexity dimension.

Also shown in the figures, the experimental group scores consistently improved across the two written submissions for all four dimensions. For the experimental group, there was a statistically significant difference between students' scores on the first and third submissions for the mathematical model dimension. All groups had statistically significant improvements across the written submissions for the reusability dimension. As demonstrated in the figures above, the students' scores in the two comparison groups were not consistently improving nor declining; their scores typically fluctuated across the four submissions.

Conclusions

The results identified differences that exist in the mathematical and computational model development between students who were exposed to the revised modeling language and examples (experimental group) compared to those students who were not (comparison groups). There were four dimensions in the modeling rubric that were analyzed for this study: mathematical model complexity, modifiability, reusability, and reusability [8, 29]. The dimensions of the rubric were analyzed separately to capture the nature of the differences between the experimental and comparison groups. Since this study complemented the work of a previous study [28], these results are compared where relevant.

Rodgers et al. [28] found that students in the experimental group demonstrated statistically significant improvements across all three analyzed dimensions (i.e. mathematical model complexity, modifiability, and reusability), while the comparison group only demonstrated statistically significant improvement across the reusability dimension. In this study, the experimental group demonstrated statistically significant improvements from across their written submissions on the mathematical model complexity and reusability dimensions, while the comparison groups only demonstrated statistically significant improvement on the reusability dimension (across the two written submissions). There were no other positive statistically significant changes. The change in submission requirements that students had to add comments in their code seemed to have a huge impact on students' lower scores on coded submissions for the reusability and modifiability dimensions.

Although there was not as consistent improvement seen in students' coded solutions, the students in the experimental group presented higher scoring ideas in their written mathematical models than the implemented in their computational models (i.e. coded solutions). There is a need for further investigation into how to help students improve their computational models. Future research will integrate in revisions to enhance computational thinking skills into the course to hopefully help enable students to better improve their computational models (coded solutions).

The students who were exposed to the revised modeling language and materials had the only statistically significant gains across the course of the semester in their ability to address the complexity of the problems (mathematical model complexity). The experimental group had the greatest improvement on their written solutions in their ability understand the application of the model and potential for adaptation (modifiability). Again for the coded submissions, the students had to comment their rationale in their code. This may have caused the much lower scores on the coded submissions. One of the comparison groups had the most consistent grades between the written and coded solutions. The instructor for the this group has the most specific requirements for the format and comments in students' code submissions. These requirements may have positively contributed these students' higher average scores on this dimension and the shareability dimension.

All the groups scored similarly with high scores in their ability to address the users' anticipated needs by identifying assumptions they made in solving the problem (reusability). This improvement in both groups could be attributed to the fact that emphasizing writing code for a given user and understanding your assumptions in solving a problem to know when the solution is applicable is embedded in the course. Identifying assumptions is something that all three instructors require their students to do in various homework assignments. This may have contributed to the consistently high students' scores on this dimension.

The students who were not exposed to the modeling language nor given the opportunity to engage in modeling activities beyond the assigned modeling problem did not demonstrate as much improvement on the mathematical model complexity dimension. Their completion of the two versions of the one modeling problem could have had some impact on their demonstrated modeling abilities. Although, the additional course materials seemed to be much more fundamental in leading to the success seen by the experimental group. These findings help inform other researchers and instructors how to help students develop modeling skills, specifically modeling skills related to mathematical model complexity, modifiability, reusability, and shareability dimensions. This research will build upon this idea by further analyzing impact of the revised modeling language in more courses and covering more types of modeling, including physical and business models.

Acknowledgements

This work was made possible by a collaborative research grant from the National Science Foundation (DUE 1827392; DUE 1827600; DUE 1827406). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. R. Carberry and A. F. McKenna, "Exploring student conceptions of modeling and modeling uses in engineering design," *Journal of Engineering Education,* vol. 103, no. 1, pp. 77-91, 2014.

[2] A. McKenna, R. Linsenmeier, and M. Glucksberg, "Characterizing computational adaptive expertise," in *2008 ASEE Annual Conference and Exposition*, 2008.

[3] J. S. Zawojewski, H. A. Diefes-Dux, and K. J. Bowman, *Models and modeling in engineering education: Designing experiences for all students*. Sense Publishers, 2008.

[4] J. M. Wing, "Computationalthinking," in *Communications of the ACM,* vol. 49, no. 3, p. 33-35. 2006.

[5] U. Ilic, H. I. Haseski, and U. Tugtekin, "Publications trends over 10 years of computational thinking research," in *Contemporary Education Technology*, vol. 9, no. 2, p. 131-153, 2018.

[6] R. Lesh and H. M. Doerr (Eds.). *Beyond constructivism: Models and modeling perspectives on mathematics problem solving, learning, and teaching*. Lawrence Erlbaum Associates Publishers, 2003.

[7] A. R. Carberry, A. F. McKenna, R. A. Linsenmeier, and J. Cole, "Exploring senior engineering students' conceptions of modeling," in *118th ASEE Annual Conference and Exposition*, 2011.

[8] H. A. Diefes-Dux, M. A. Hjalmarson, and J. S. Zawojewski, "Student Team Solutions to an Open-Ended Mathematical Modeling Problem: Gaining Insights for Educational Improvement," *Journal of Engineering Education,* vol. 102, no. 1, pp. 179-216, 2013.

[9] H. A. Diefes-Dux, K. Bowman, J. S. Zawojewski, and M. Hjalmarson, "Quantifying aluminum crystal size part 1: The model-eliciting activity," *Journal of STEM Education: Innovations and Research,* vol. 7, no. 1/2, p. 51, 2006.

[10] H. A. Diefes-Dux, M. A. Hjalmarson, T. K. Miller, and R. Lesh, "Chapter 2: Model-eliciting activities for engineering education," *Models and modeling in engineering education: Designing experiences for all students,* pp. 17-35, 2008.

[11] E. Hamilton, R. Lesh, F. Lester, and M. Brilleslyper, "Model-Eliciting Activities (MEAs) as a Bridge between Engineering Education Research and Mathematics Education Research," *Advances in Engineering Education,* vol. 1, no. 2, p. n2, 2008.

[12] H. A. Diefes-Dux and P. Imbrie, "Chapter 4: Modeling activities in a first-year engineering course," *Models and modeling in engineering education: Designing experiences for all students,* pp. 37-92, 2008.

[13] R. M. Clark, L. J. Shuman, and M. Besterfield-Sacre, "In-Depth Use of Modeling in Engineering Coursework to Enhance Problem Solving," in *Modeling Students' Mathematical Modeling Competencies*: Springer, 2010, pp. 173-188.
[14] T. P. Yildirim, L. Shuman, M. Besterfield-Sacre, and T. Yildirim, "Model eliciting activities: assessing engineering student problem solving and skill integration processes," *International Journal of Engineering Education,* vol. 26, no. 4, pp. 831-845, 2010.
[15] K. J. Rodgers, "Development of First-Year Engineering Teams' Mathematical Models through Linked Modeling and Simulation Projects" (2016). *Open Access Dissertations*. 838. https://docs.lib.purdue.edu/open_access_dissertations/838
[16] R. Lesh and G. Harel, "Problem solving, modeling, and local conceptual development," *Mathematical thinking and learning,* vol. 5, no. 2-3, pp. 157-189, 2003.
[17] A. R. Carberry and A. F. McKenna, "Engineering student's conceptions of model uses in design," in *2011 Research in Engineering Education Symposium, REES 2011*, 2011.
[18] R. Lesh, M. Hoover, and A. Kelly, "Equity, assessment, and thinking mathematically: Principles for the design of model-eliciting activities," *Developments in school mathematics education around the world,* vol. 3, pp. 104-130, 1993.
[19] O. B. J. Daniel K. Howe, "Developing an Interactive Computer Program to Enhance Student Learning of Dynamical Systems," in *2016 ASEE Annual Conference & Exposition*, 2017.
[20] A. J. Magana, "Modeling and Simulation in Engineering Education: A Learning Progression," *Journal of Professional Issues in Engineering Education and Practice,* vol. 143, no. 4, p. 04017008, 2017.
[21] A. J. Magana, S. P. Brophy, and G. M. Bodner, "Instructors' intended learning outcomes for using computational simulations as learning tools," *Journal of Engineering Education,* vol. 101, no. 2, pp. 220-243, 2012.
[22] A. Stefan, "A Computer Model of Cell Dynamics Using Agents," in *American Society for Engineering Education*, 2010: American Society for Engineering Education.
[23] J. P. A. Omer Farook, Athula Kulatunga, Ashfaq Ahmed P.E., Wangling Yu, Yoonill Lee, Hassan Abdullah Alibrahim, "Freshman Experience Course in Electrical and Computer Engineering Technology Emphasizing Computation, Simulation, Mathematical Modeling, and Measurements," in *2017 ASEE Annual Conference & Exposition*, 2017.
[24] A. El-ZEin, T. Langrish, and N. Balaam, "Blended Teaching and Learning of Computer Programming Skills in Engineering Curricula," *Advances in Engineering Education,* vol. 1, no. 3, p. n3, 2009.
[25] H. Fangohr, "A comparison of C, MATLAB, and Python as teaching languages in engineering," in *International Conference on Computational Science*, 2004: Springer, pp. 1210-1217.
[26] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald, "The impact of pair programming on student performance, perception and persistence," in *Software Engineering, 2003. Proceedings. 25th International Conference on*, 2003: IEEE, pp. 602-607.
[27] L. Williams and R. Upchurch, "Extreme programming for software engineering education?," in *Frontiers in Education Conference, 2001. 31st Annual*, 2001, vol. 1: IEEE, pp. T2D-12.
[28] K. J. Rodgers, J. C. McNeil, M. A. Verleger, and F. Marbouti, "Impact of a modeling intervention in an introductory programming course," presented at the 2019 ASEE

Annual Conference & Exposition Tampa, Florida, 2019. [Online]. Available: https://peer.asee.org/32918.

[29] H. A. Diefes-Dux, J. S. Zawojewski, and M. A. Hjalmarson, "Using educational research in the design of evaluation tools for open-ended problems," *International Journal of Engineering Education,* vol. 26, no. 4, p. 807, 2010.

[30] J. M. Stonehouse and G. J. Forrester, "Robustness of the t and U tests under combined assumption violations," *Journal of Applied Statistics,* vol. 25, no. 1, pp. 63-74, 1998.

Appendix A. Modeling Problem Assignment (Version 1 – Submission 1 and 2)

Modeling Problem v.1 – Wind Farm Development

Problem:

Zee's Alternative Energies was founded in 2000 to increase the use of alternative energies across the United States. The company has been extremely successful with their development of solar energy farms. They are expanding and looking to develop a wind farm of horizontal-axis wind turbines.

You have been hired by Zee's Alternative Energies to develop a reusable computational model that will determine the best location to build a wind turbine farm based on historical wind data. The company has provided you with some descriptive statistics for a few locations that they are considering. They will send more data for potentially more locations later for further analysis. You must first provide a written solution (description of your model) and then later provide a coded solution (implementation of your model). Keep in mind your solution must be developed in a way that it will work for various locations *(meaning your solution should make sense for locations with different values and it should be easy to change the data in your code)*.

Submissions:

Please note that the submission for this assignment is different than any of your homework assignments. (It is worth 5% of final grade, hence requires more work than usual.). A written model will be due for this problem in about one week and then your code for your solution will be due about one week later *(refer to Canvas for exact dates)*.

- (Part1 due now): Your **model** – for this submission you must complete the Engineering Process (steps 1 to 7a only) either typed or handwritten as described in the document. You must also follow these additional guidelines: Keep in mind ***assumptions*** will be very important for your user to understand the constraints and limitations for using your solution. For your solution steps (Step 5), you must explain why you went about solving the problem the way you did – meaning you must explain your ***rationale*** for designing the steps the way that you did *(refer to the example provided in the bullet point below)*. In addition to your written model, you must provide the outputs of your model based on the provided data.
    - Example of some steps and *justifications in italics* (keep in mind some parts of these examples may be wrong – we can't give you all the answers! ☺): (EXAMPLE 1) The locations for which "the maximum of the daily maximum wind speeds are between the rated output speed and cut-out speed" are ranked towards the top. *I want the wind turbine to run within the rated output power most of the time. I have decided that having the maximum at within this range will help ensure the wind speeds of the selected location are more likely within the rated power output.* - . (EXAMPLE 2) The locations that have a mean average wind speed closest to the cut-out speed without going over the cut-out speed are ranked towards the top. *I want the wind turbine to run within the rated output power as much as possible. I have decided the closer that the mean is to being within that range the better, but I have eliminated above the cut-out speed because I want to ensure the data is not above the specified range for the rated output power.*
- (Part2 due later): Your **code** (or computational model) – your code will be based on the model that you submitted above. It is natural to modify your model (and algorithm) throughout the coding process. You are required to ***comment*** your assumptions (for Step 4) and all your justifications for each step throughout your code. In addition to your code, you must provide ***outputs*** for your code (as specified in Steps 5&6).

Solution:

Below are the 7 steps of the engineering process. They are also shown in Figure 1. These steps can be used to guide you through solving a problem. Sometimes you will cycle through these steps, as shown by the arrows in Figure 1. ***You are required to show your work for all of these steps, as specified under each step.***

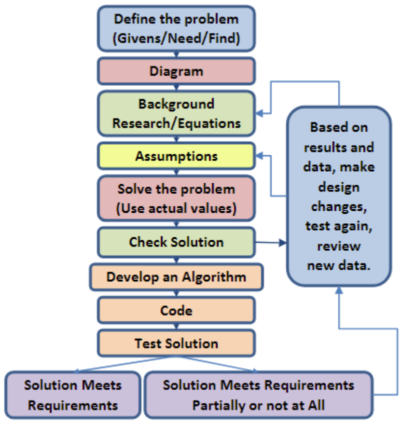
Define the problem (Givens/Need/Find)
Diagram
Background Research/Equations
Assumptions
Solve the problem (Use actual values)
Check Solution
Develop an Algorithm
Code
Test Solution
Solution Meets Requirements
Solution Meets Requirements Partially or not at All
Based on results and data, make design changes, test again, review new data.

Figure 10. Engineering Process

1. Decipher Problem Statement

The first step is to decipher the problem and identify the information (or variables) given and what information you need to find (or display to the user at the end of your program).

Givens *(inputs)*:

The user has told you that they have provided you with **historical wind data** to use for your sample output (shown in Table 1). The data consists of some descriptive statistics about the daily maximum, daily average, and daily minimum wind speed for the entire year of 2018 for five locations. *An example of what the raw data (before the descriptive statistics were found) looked like for one day is shown in Table 2.* The relevant data that you decided to use will be **hardcoded in your code** (computational model), but be sure to code it in a way that it is easy to change the data later. *Be sure to clearly state what data you plan on using and why. If you do not use any data, be sure to clearly state what data you are not using and why you are not using that data.*

Table 1. Wind Speed Data for Five Locations in 2018 (365 days: Jan. 1st – Dec. 31st)

Locations		**mean**	**mode**	**standard deviation**	**minimum**	**maximum**
Buffalo, NY	Daily Maximums	20.4 mph	16.0 mph	6.2 mph	9.0 mph	47.0 mph
	Daily Averages	7.8 mph	8.0 mph	3.9 mph	2.0 mph	25.0 mph
	Daily Minimums	3.5 mph	0.0 mph	3.2 mph	0.0 mph	13.0 mph
Abilene, TX	Daily Maximums	21.9 mph	23.0 mph	6.6 mph	7.0 mph	43.0 mph
	Daily Averages	10.9 mph	0.0 mph	8.1 mph	0.0 mph	31.0 mph
	Daily Minimums	3.8 mph	0.0 mph	3.7 mph	0.0 mph	17.0 mph
Oklahoma City, OK	Daily Maximums	18.2 mph	15.0 mph	7.0 mph	0.0 mph	46.0 mph
	Daily Averages	7.5 mph	4.0 mph	4.3 mph	0.0 mph	22.0 mph
	Daily Minimums	3.9 mph	0.0 mph	4.1 mph	0.0 mph	18.0 mph
Wichita, KS	Daily Maximums	22.9 mph	21.0 mph	7.2 mph	8.0 mph	47.0 mph
	Daily Averages	8.6 mph	10.0 mph	4.9 mph	1.0 mph	24.0 mph
	Daily Minimums	3.9 mph	0.0 mph	4.0 mph	0.0 mph	21.0 mph
Rochester, MN	Daily Maximums	18.9 mph	9.0 mph	8.7 mph	4.0 mph	44.0 mph
	Daily Averages	8.8 mph	7.0 mph	4.2 mph	2.0 mph	20.0 mph
	Daily Minimums	1.8 mph	0.0 mph	2.6 mph	0.0 mph	12.0 mph

Table 2. Sample of Data Collected for January 1st, 2018 in Abilene, TX

	Date		**Wind Speed (mph)**		
Location	**Month**	**Day**	**Daily Maximum**	**Daily Average**	**Daily Minimum**
Abilene, TX	January	1	15	4	4

Finds *(outputs)*:

The output of your model must be a ranking of all the provided locations from best location for the company to purchase land in for a wind farm to worst location. All locations must be ranked and there cannot be any ties for any locations. *You must ensure that a tie will not happen so make sure that you consider many points in your data.*

2. Draw a Diagram

Sometimes the problem will include a diagram; be sure to still draw your own diagram/s. This step will help you better decipher the problem by visualizing it. ***This is the only step you are not required to do. If you draw anything out to help you visualize this problem be sure to include your sketches.***

3. Identify Relevant Theory

After determining the information that you know and need to find and drawing a diagram, you should start to have an idea what theory and/or background information you need to solve the problem. In this step you will identify the information (e.g., formulas, data) that you will need to create a solution to the given problem. ***Some information has been provided for you.***

The horizontal-axis wind turbines used for this project will operate according the specifications as described and presented in Table 3 and Figure 2. *Please note you are not given any formulas for power because you will not need to calculate the power produced to solve this problem.*

Table 3. Wind Turbine – Wind Speed vs. Power Output Explanation

Wind Speed Range (m/s)	Explanation
Below 3.5 m/s (the cut-in speed)	This wind speed will be insufficient to provide the necessary torque to rotate the turbine blades (meaning no power will be generated).
Between 3.5 m/s (the cut-in speed) and 14 m/s (the rated output speed)	The wind turbine will produce more power with higher wind speeds, as shown in Figure 2.
Between 14 m/s (the rated output speed) and 25 m/s (the cut-out speed)	The wind turbine will produce the same amount of power for all these wind speeds because the limit of the electrical generator is reached.
Above 25 m/s (the cut-out speed)	The wind turbine's braking system will be initiated to stop the system form moving to ensure the rotor is not damaged from excessive wind speeds (meaning no power will be generated).

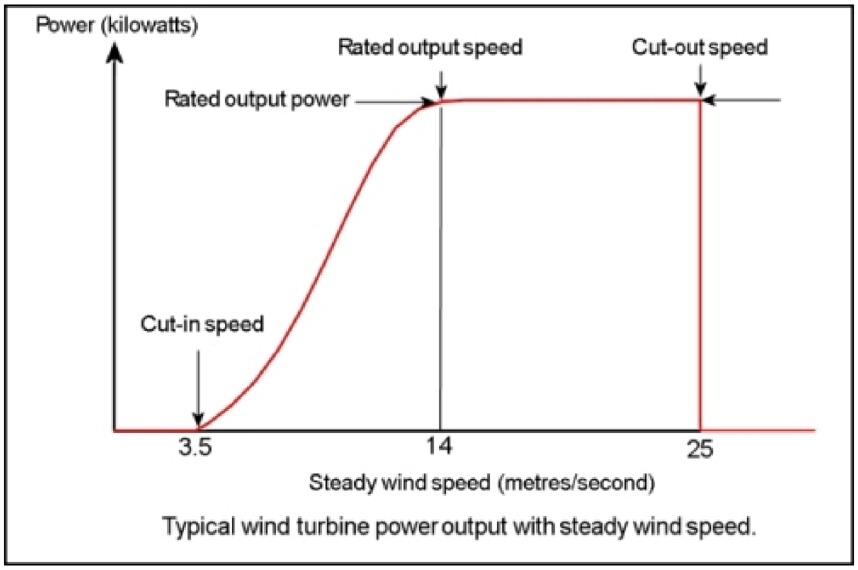

Figure 2. Wind Turbine Wind Speed vs. Power Output
Image retrieved from: http://www.wind-power-program.com/turbine_characteristics.htm

4. Assumptions

In this step you will communicate some ideas that you have assumed to simplify the problem. These are things that you may try to address later to make your solution address a more realistic scenario. It is okay if you cannot envision how you would solve the problem without the assumption, but try to think about this for each assumption that you write.

You must write at least 3 assumptions you will make in coming up with a solution for this problem. (*EXAMPLE: The elevation of the location where the wind data was collected will be the same as the elevation of the potential land the company will purchase. – you cannot use this as one of your assumptions, but maybe this example will help you think of others.)*

5. Solution Steps

This step requires solving for the finds using the givens and theory. For this step you will create equations to solve the problem, but you will not plug in any numbers yet. Be sure to base your equations on variables and NOT numeric values (unless they are a constant, such as pi). Be sure to also refer to your diagram and assumptions to help you through this step. Throughout this step you may find it useful to go back and draw another diagram or necessary to use another formula.

Before figuring out a ranking of locations for the provided data sets, determine your model. Define any steps and/or calculations that you will do to determine the rankings. Once you have developed your model, try it with the provided data set in Step 6. Evaluate your output and determine if you want to return to this step to modify your model. Development of a good model involves iteration like this. Be sure to show all your work.

6. Identify Results and Verify Accuracy

Now that you have solve the problem without plugging in values, you will plug in values for this step to verify if your problem is accurate or not.

You must include the ranking that your model will output in your first submission. You must show every step of your solution for the provided data to show how you get to this final ranking. You must also include the ranking that your code outputs as a comment in your code for your second submission.

7. Algorithm and Code

Your solution steps will lay out the process that you will need to code. In complex problems that require conditionals and/or repetition, it may be beneficial to draw out a flowchart, concept map, etc. or write out bullet point or numbered steps. Doing this step can ensure you understand the flow of your code before you start writing code in MATLAB.

Based on your work in Step 5, you should have a well-formed model that can be implemented as a program. To ensure that you are prepared to start coding, develop an algorithm that clearly summarizes the steps you will take in your program. You can do this with a flowchart, numbered steps, or other methods you may have discussed in class. If your class has a designated algorithm format, you must use that for this step. Keep in mind that your model in Step 5 will look very similar to what you complete for the algorithm in this step.

Only for the second submission will you code your solution. When you code your solution, keep these steps in mind and revisit any that may help you in further developing your solution. (Refer back to steps 5 & 6, you must also provide outputs commented in your code.)

Appendices B. Wind Farm Modeling Problem Rubric (Submission 4)

Criteria	Ratings				Pts
Reusability - Assumptions Does the solution have at least 3 clearly communicated assumptions? There should be significantly more, but 3 is the minimum. (Reminder: Assumptions are not facts. Assumptions are things assumed to solve the problem - information that needs to be addressed to move forward, but was not given.) (EXAMPLE: The elevation of the location where the wind data was collected will be the same as the elevation of the potential land the company will purchase. – you cannot use this as one of your assumptions, but maybe this example will help you think of others.)	1.0 pts Full Marks	0.5 pts partial include 3 "assumptions", but are facts not assumptions. OR include less than 3 assumptions, but have at least 1 valid assumption	0.0 pts No Marks		1.0 pts
Mathematical Model Complexity - Data Set Student took all of the available data into account. A significant amount of the data was used to help them evaluate the ranked locations for a wind farm.	8.0 pts Full Marks Used at least 67% of the data for their analysis AND some dist. approach OR if used statistical measurements, used at least 3.	6.5 pts partial Used at least 33% of the data for their analysis AND some dist. approach OR if used statistical measurements (e.g., mean, mode, stdev), used at least 2 with at least 2 data sets (daily mins, maxs, avgs) or at least 1 with all 3 data sets (daily mins, maxs, avgs).	4.0 pts attempted did not use much of the data	0.0 pts No Marks	8.0 pts
Modifiability - Data Set (used data) Student communicated what data were using and why they were using this data.	1.0 pts Full Marks	0.0 pts No Marks			1.0 pts
Modifiability - Data Set (unused data) Student communicated what data were not using and why they were not using this data.	1.0 pts Full Marks	0.0 pts No Marks			1.0 pts

Criteria	Ratings			Pts
Mathematical Model Complexity - Number and Types of Locations The model/code is not designed only for the given data set. It would work for different data. There are enough factors considered that ties would not easily happen. Also the code would work with a different data file that contained different locations and a different number of locations (assume same file name).	3.0 pts Full Marks some limitations are to be expected	1.5 pts Many Limitations/Errors Student clearly attempted to ensure their model/code would work for other data, but there are many limitations/errors.	0.0 pts No Marks	3.0 pts
Modifiability - Number and Types of Locations Student clearly communicates the need to ensure their model will work for other scenarios. (e.g., adding additional steps that may not matter for the provided data, but adding them to ensure a potential tie wouldn't happen given another scenario, when considering different data this added new steps to their model)	2.0 pts Full Marks	0.0 pts No Marks		2.0 pts
(CODE only) Data Correctly Implemented in Code Data set is uploaded into code correctly (used xlsread function with correct return variables). The data types are handled in code appropriately (e.g.,: working with cell arrays)	3.0 pts Full Marks	1.5 pts partial some errors throughout, but at least some correct aspects demonstrated	0.0 pts No Marks	3.0 pts
Mathematical Model Complexity - Data Type (Conversions/Units) Any necessary conversion calculations are done. Units are consistent. (For code, any conversions from provided data should be done in code and not in the file.)	2.0 pts Full Marks	0.0 pts No Marks		2.0 pts
Mathematical Model Complexity - Rated Output Speed/Power (from the Power Output Chart) Student shows a significant amount of consideration of the rated output speed/power on power output chart.	4.0 pts Full Marks	2.0 pts partial did not use this much, but at least acknowledged in their solution.	0.0 pts No Marks	4.0 pts
Modifiability - Rated Output Speed/Power Student communicated where and how the rated output speed/power was being used within (a) rationale/s.	2.0 pts Full Marks	0.0 pts No Marks		2.0 pts

Criteria	Ratings	Pts
Mathematical Model Complexity - Cut-in and/or Cut-out Speed (from the Power Output Chart) Student shows a significant amount of consideration of either the cut-in or cut-out speed from the power output chart.	4.0 pts Full Marks | 2.0 pts partial did not use this much, but at least acknowledged in their solution. | 0.0 pts No Marks	4.0 pts
Modifiability - Cut-in/Cut-out Speed Student communicated where and how the cut-in speed and/or cut-out speed were being used within (a) rationale/s.	2.0 pts Full Marks | 0.0 pts No Marks	2.0 pts
Shareability - Format/Communication Student formats their solution in a way that enables someone else to apply their solution and replicate the same results. The process is clearly communicated.	9.0 pts Full Marks | 6.0 pts partial There is 1 error in the code. | 3.0 pts attempted There are 2+ errors in the code. | 0.0 pts No Marks	9.0 pts
(CODE only) **Shareability** - Outputs The program clearly communicate to the user what the results are (ranking of locations for wind farm).	2.5 pts Full Marks | 0.0 pts No Marks	2.5 pts
(CODE only) Sample Data Student generated sample data to test their solution. Excel file sample data provided.	3.0 pts Full Marks | 0.0 pts No Marks	3.0 pts
Shareability - Sample Calculations/Output The sample outputs for the provided data and generated data in student's code in comments (comment block - identify original and generated data sets).	2.5 pts Full Marks | 1.5 pts sample output only for provided cities - not with additional sample data | 0.0 pts No Marks	2.5 pts
		Total Points: 50.0