CHARACTERIZING COMPUTATIONAL THINKING THROUGH THE USE OF MODELING AND SIMULATION ACTIVITIES WITHIN THE ENGINEERING CLASSROOM

by

Joseph A. Lyon

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Engineering Education

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Soli Deo Gloria.
To my wife Alivia, I love you and will always love you.

To my daughter, Hazel, thank you for letting papa work AND take care of you during the day.

To my parents, who have been my greatest supporters my whole life.

ACKNOWLEDGMENTS

First and foremost, I'd like to thank God for giving me the support system and blessings that have enabled me to get to this point. With Him, all things are possible, and I am indeed living proof of that. As I sit here writing this, I think of all the support and sacrifices that my family and others have made for me to get to this point.

To my wife, Alivia, you are a constant when things feel unstable. You help me stand when I can't anymore. I love you more than you know. Your constant encouragement and faith in me through this process made it all possible.

To my daughters Hazel and Ella, I love you both and thank you for the sacrifices that you didn't even know you were making. There were many days where I had to write and care for you at the same time, and probably was imperfect at both. Just know that my hope and dream for you is that you will never fear to follow the path you are called towards. And I hope that I can give you every opportunity to succeed as you navigate that path and help you grow to be the women of valor I know you can become.

To my parents, thank you for the support that you have given me my whole life. You never pressured me to do anything other than follow the path to which I was called. There are little pieces of you in every bit of work I do, and I hope reaching this point makes you proud. I love you both.

To Ale, thank you for believing that I could do this and supporting me through the entire process. Your mentorship throughout this process was invaluable, and I look forward to many more years of learning from you. Thank you for pushing and stretching me throughout graduate school in ways that you knew would benefit me for the rest of my career. I cannot thank you enough for your guidance.

To Ruth, thank you for giving me the confidence and belief that I could succeed. That a first-generation college student, let alone a Ph.D. student, could thrive in the academic world. Thank you for listening to me when I needed to talk and being there when I was extremely stressed. I owe a lot of my success and progress through this program to your mentorship and guidance.

To Michael Loui and Muhsin Menekse, thank you for your feedback and all of the advice and support. Thanks for all the little things that a committee member does that certainly add to big things. The letters of recommendation, the one-on-one meetings, and the feedback on my writing were critical to me writing and finishing this dissertation and succeeding as a graduate student.

To Richard, thank you for being my graduate school mentor and a close friend. You were always a guidepost for how to navigate the decisions and situations that comprise being a Ph.D. student. The long talks and little pieces of encouragement over the years have been vital in me being able to finish what sometimes felt unfinishable.

To my church family and small group, thank you for being a family to my family. Thank you for showing me constant grace and support, even though sometimes you had no idea what I was doing. Thank you for being the hands and feet of Jesus and welcoming my family to join you in the mission. You were always there to listen, pray for me, and love my family.

Finally, I would like to thank the National Science Foundation for supporting my graduate research fellowship (DGE-1842166). It enabled me to do work I was passionate about and chart my research path without fear of how I would support my graduate school education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The concept of computational thinking (CT) has become more prevalent across the engineering education research and teaching landscape. Yet much of the research to date has been more definitional and has not offered many ways to convert CT theory to practice. One prominent set of tools used across engineering disciplines is modeling and simulation, which allows students to create a representation of the outside world as they understand it.

This three-paper dissertation connects modeling and simulation skills with eliciting CT by leveraging model-based reasoning as a theoretical framework. A learning design was created and delivered here via design-based research that includes educational frameworks such as productive failure and model-eliciting activities (MEAs) to structure the modeling activity within a classroom setting. The designed learning intervention used a four-part sequence to scaffold the modeling activity in the classroom: (1) planning the model, (2) building the model, (3) evaluating the model, and (4) reflecting on the model. A case study of a final-year capstone course in biological engineering implemented the four-week designed learning intervention as part of the course.

The guiding research question for the study was *how do modeling and simulation activities elicit computational thinking practices in the context of undergraduate engineering education?* To approach this question, data were collected in audio transcripts and student-generated artifacts to identify areas where the modeling activity elicited different forms of CT in the student work. The first study examined how CT was elicited within the model-building phase and developed an initial codebook for CT practices and outcomes using thematic analysis. The second and third studies built upon that codebook and further the outcomes by analyzing the modeling activity's planning and evaluating/reflecting phases. The results indicate that CT is used throughout the entire modeling and simulation process as students engage in model-based reasoning. The identified CT practices of abstraction, algorithmic thinking, evaluation, generalization, and decomposition emerged from a thematic analysis, and each practice was further characterized and refined into a set of outcomes. Furthermore, each phase of the modeling activity emphasized unique CT outcomes suggesting that students would benefit from enacting the entire modeling and simulation process to acquire and practice a diverse range of CT outcomes.

1. INTRODUCTION

1.1 Background

Computation and computational thinking are of critical interest to the research and policy-making communities as all industry sectors continually move towards computational practices (National Research Council, 2011; President's Information Technology Advisory Committee, 2005). And while these skills are essential in many areas, they are of particular concern within engineering, especially to skills such as modeling and simulation (Magana & Silva Coutinho, 2017). Furthermore, modeling and simulation skills are critical to professional practice within the engineering discipline (Gainsburg, 2006). Because of this, educational researchers must find new ways to implement computation and computational thinking practices within undergraduate coursework, and modeling and simulation seem to be prime areas in which to do so. The integration of modeling and simulation activities with the elicitation and learning of computational thinking will allow for computational thinking to be built as a core skill within an already crowded engineering curriculum (Magana & Silva Coutinho, 2017).

Computational thinking, as a construct, is newer to the broader research literature and the world of educational research. Seymour Papert was a pioneer in what would eventually become computational thinking by discussing the symbiotic relationship between children programming and learning about how they think in the process (Papert, 1980). However, Jeanette Wing has largely been credited with popularizing the term in 2006 (Wing, 2006). Additionally, there has been growing research since the term's origination on what computational thinking is, why it is necessary, what its encompassing definition entails, and how to best use it (Ilic et al., 2018). When introducing the concept, Wing (2006) described computational thinking as "a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability (p. 33)." Papert, in his seminal work on computers and thinking, *Mindstorms: Children, Computers, and Powerful Ideas*, wrote that "in teaching the computer how to think, children embark on an exploration about how they themselves think" (Papert, 1980, p. 19). This dissertation addresses the learning need for looking at how this form of thinking, computational thinking, can be integrated into the engineering classroom within the context of higher education. By doing so, this dissertation aims to advance the understanding of how instructors can elicit computational thinking behaviors within students and the diverse ways that students use computational thinking throughout the modeling process. Yet, to better understand the need for computational thinking, we should first understand the environment and discipline from which it emerged, computing.

1.2 Computing: The third pillar of science

Traditionally, science is said to have two pillars, theory and experiment, underpinning the scientific method and how we come to learn about the world around us. However, in recent years, the idea of computing and computation as the third leg of science has become a common and popular idea (Skuse, 2019). Even national calls for computation within educational settings have acknowledged this third pillar as a unique skill that fundamentally contributes to the scientific method (President's Information Technology Advisory Committee, 2005). This third pillar has primarily emerged from the advancement of technology and its ability to work with large datasets and perform complex calculations. In an editorial regarding computation as the third pillar of science, Skuse (2019) wrote that "computation is not just an extra tool. It is a new way of doing science, irrevocably changing how scientists learn, experiment and theorize" (p. 40).

Thus, computing is changing how we learn, experiment with the world around us, and theorize about how the world around us works. And because of this importance, we must understand the thinking patterns that emerge when engaged with this third pillar of science. Science and the scientific method have fundamentally changed:

> The volume and rate at which scientists and engineers are now collecting and producing data--through instruments, experiments and simulations--are demanding advances in data analytics, data storage and retrieval, as well as data visualization. The complexity of the multi-dimensional systems that scientists and engineers want to model and analyze requires new computational abstractions (Wing, 2011, p. 3).

Thus, producing scientists and engineers ready to engage in this practice, where computing and computational science are paramount, needs research that investigates how to elicit and build these skills in our students from the time they walk into a classroom until they leave.

1.3 Why is computational thinking important?

Although computational thinking encompasses more than just computation and computer science-related concepts, it finds its most natural fit in these same spaces. This form of thinking, while not called computational thinking at the time yet, was the idea that children could interact with computers and begin to learn about their thinking processes in terms of their interaction with the computer (Papert, 1980). When popularized by Jeannette Wing, she further defined computational thinking as a set of skills derived primarily from computer science (Wing, 2006). Thus, the importance of computational thinking centers on computation and the types of thinking that go along with computer programming. Therefore, as the need for computation in the workforce increases, so does the need for computational thinking-enabled professionals. While computational thinking need not be tied directly to computer programming, the structure of computational thinking is born out of computer programming.

Computation and programming are becoming a skill that is increasingly desired within the engineering workforce. Specifically, the United States government considers education in computation a matter of global economic competitiveness and national security (President's Information Technology Advisory Committee, 2005). As computers and the need for computation become ubiquitous across all industries, failure to properly teach these skills within educational systems may further the gap between the demand for computationally enabled professionals and the currently produced supply. Additionally, government agencies such as the National Research Council have held workshops intending to look at how best to teach computation and computational thinking (National Research Council, 2011). Yet, much of the research delivered by academia continues to be definitional or theoretical as opposed to concrete ways to teach computational thinking within all levels of education, but especially within higher education (Ilic et al., 2018; Kalelioğlu et al., 2016).

1.4 What are the current issues that need addressing relating to computational thinking?

There are many issues facing the research literature regarding computational thinking and its use within educational settings. The biggest problem is that computational thinking continues to be heavily definitional due to the high amount of confusion associated with the term (D. Barr et al., 2011; Grover & Pea, 2013). The definitional nature of the literature has left educators implementing computational thinking into their classrooms based on highly varied points of perspective (Kalelioğlu et al., 2016). This variety in implementation leaves practitioners with little ability to understand if students are using and building computational thinking skills within classroom interventions.

Ultimately, there is a need in the literature for design-based research studies that aim to create and implement computational thinking interventions across all levels of education (Ilic et al., 2018). These studies should address the content of the course and issues of pedagogy and assessment as well (Ilic et al., 2018). As mentioned earlier, another issue that needs addressing is the implementation of computational thinking into higher education environments. Studies looking at implementation within higher education populations are less common than studies focusing on K12 students (Kalelioğlu et al., 2016). Thus, there is a need for research within computational thinking that implements content, assessment, and pedagogy into higher education settings.

1.5 Purpose of the research

The purpose of this dissertation study is threefold. First and foremost, the purpose of this dissertation is to create and implement educational units that effectively answer national calls for key computation and computational thinking skills (National Research Council, 2011; President's Information Technology Advisory Committee, 2005). Additionally, this dissertation addresses gaps from the literature for the more practical implementation of computational thinking, more research regarding undergraduate populations, and operationalizing computational thinking definitions for practical use (Ilic et al., 2018; Kalelioğlu et al., 2016; Weintrop et al., 2016). And finally, this dissertation aims to understand how often and in what ways students use computational thinking within a modeling and simulation intervention.

The selected field of study is engineering education, addressing a need for the disciplinary implementation of computational thinking (Ilic et al., 2018). A critical aspect of the engineer's job is modeling and simulation practices. Such practices often incorporate computation skills (Gainsburg, 2006; President's Information Technology Advisory Committee, 2005). This incorporation means that modeling and simulation, when implemented into engineering effectively, can also help address the growing disparity between the need and supply of computational skills within the modern workforce.

1.6 Guiding research question and scope of the study

This research aims to understand how computational modeling activities are effectively designed to incorporate computational thinking into engineering classrooms. Based on this purpose, one research question guides the study:

(1) How do modeling and simulation activities elicit computational thinking practices in the context of undergraduate engineering education?

This research question defines the scope of the study in multiple ways. First, the study only focuses on undergraduate populations instead of other educational levels. Because of this, this dissertation makes no claims about generalizability beyond the engineering context within which it was used but instead develops guiding concepts to be used as a starting point to eliciting computational thinking and abstraction behaviors within a variety of disciplines.

Additionally, this research only focuses on eliciting and using computational thinking within modeling and simulation practices. While there are skills useful in other engineering discipline areas, modeling and simulation are the boundaries that scope this dissertation. This scoping does not say that modeling and simulation activities are the only way to elicit these needed computational thinking outcomes. This dissertation aims to prove that modeling and simulation effectively elicit these outcomes.

Finally, this dissertation does not measure how much computational thinking is improved upon within modeling and simulation activities. Instead, it focuses on the different ways in which computational thinking is used and performed during these classroom interventions. Therefore, this study involves qualitatively understanding these behaviors and demonstrating how students use them within the context of the study.

1.7 Summary

In summary, computation and computational thinking are practices in science and engineering that are of paramount interest to research, industry, and governmental entities. Computing has even been suggested as the third pillar of the scientific enterprise. As all industry sectors move towards more computational methods, teaching these skills to engineering students is vital to ensure their competitiveness and success in solving the complex problems of tomorrow. Modeling and simulation activities are ideal for teaching these skills, and this dissertation will characterize the different ways that modeling activities elicit computational thinking practices throughout their completion.

2. LITERATURE REVIEW

2.1 Introduction

This chapter overviews the relevant literature to provide a proper background to the concepts and ideas discussed in this study. Figure 1 shows the organization of the topics in this literature review, starting from the broadest (modeling and simulation) and funneling into the most specific topic of computational thinking.

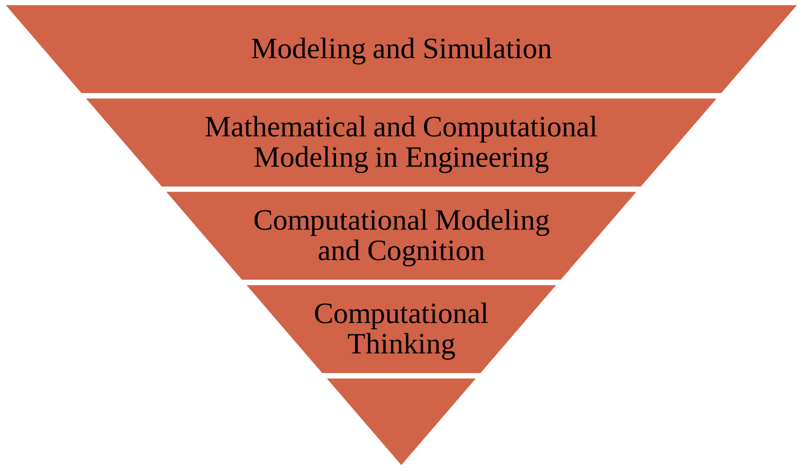
Modeling and Simulation
Mathematical and Computational Modeling in Engineering
Computational Modeling and Cognition
Computational Thinking

Figure 1. Structure of the topics within the literature review.

First, the chapter discusses modeling and simulation in the classroom in a general sense and then discusses mathematical and computational modeling within engineering education. The review links these topics into computational thinking and abstraction, looking at how computational thinking has been defined in the literature and used in the classroom.

2.2 Modeling and simulation in the classroom

Modeling and simulation have long been used across various degrees and disciplines in the classroom. However, how they are used can differ significantly based upon the goals of the educational intervention in which they are embedded. Two concepts that are greatly useful in demonstrating this difference would be the idea of black-box versus glass-box approaches to modeling and programming in general. Glass-box approaches allow students to see and interact with the underlying model and programming (Jonassen, 2009). Glass-box approaches differ from black-box approaches that only allow the students to interact with the model's interface and not change or view the underlying programming. A black-box approach would prove fruitful if the educational goal is merely to run simulation or experiments. If the educational goal is to learn programming or aspects of designing a model itself, a glass-box approach is needed, although both can be useful within learning contexts (Vieira et al., 2017).

However, more broadly, modeling has found use across STEM disciplines, such as the physical sciences (Campbell et al., 2013; Weintrop et al., 2016), mathematics (R. Lesh et al., 2000; R. A. Lesh & Zawojewski, 2007; R. Lesh & Harel, 2003), and engineering (Diefes-Dux et al., 2004, 2006; Hjalmarson et al., 2006; Moore et al., 2013). Previous literature reviews have found multiple themes that address how students approach modeling within educational contexts: (1) student strategies are either mathematical, contextual, or both, (2) student strategies to mathematical modeling are diverse and nonlinear, and (3) student strategies to modeling involve simplifications (Lyon & Magana, 2020a)

Because students take nonlinear approaches to construct models, one of the most challenging areas for implementing modeling and simulation into any classroom is assessing the work. While teaching and learning are often covered in the literature, assessment is often more difficult to find research into effective methodologies (Lyon & Magana, 2020a). One reason for this is the open-ended nature of the responses, especially when the class sizes are large (Diefes-Dux et al., 2004). It often becomes a battle for more resources due to the limited amount of grading time and the difficulty in assessing qualitative grading rubrics. However, the work is well worth it. These open-ended modeling activities can help shift instructors from focusing on summative assessment to formative assessment and change attitudes about how to assess students on other tasks (Moore et al., 2015).

2.3 Mathematical and computational modeling in engineering

Computational modeling is such a skill that combines programming skills and mathematical applications while filling the CS&E gap that plagues engineering departments. When the modeled activities are presented in a discipline-specific context, computational modeling activities have been shown to increase the perceived significance of computational modeling by engineering students (Magana et al., 2013). In addition, when implemented in conjunction with typical lecture-type classes, computational modeling and simulation have been shown to dramatically increase conceptual understanding by allowing students to visualize and predict the behavior of the modeled system (Brophy et al., 2013). Research into why engineering departments have not widely adopted these CS&E activities, specifically computational modeling, has delivered four main hurdles: limited time, limited curriculum space, faculty knowledge, and student skillset (Magana & Silva Coutinho, 2017). Thus, computational modeling in a discipline-specific context may provide a solution that both fills the CS&E gap and an opportunity to incorporate computational thinking into the upper-division engineering classroom. In addition, computational modeling pedagogy offers an environment to observe both mathematical and computational thinking develop in an engineering education context.

The literature has much to say about how engineering students learn, are taught, and should be assessed regarding mathematical modeling. Lyon and Magana (2020a) found multiple themes throughout the literature regarding how mathematical modeling should be taught in the engineering classroom: (1) mathematical modeling activities should be situated in a real-world context, (2) implementation can be both extension and creation, (3) student background knowledge must be adequate, (4) modeling activities should be implemented in modules, (5) modeling activities should include adequately difficult concepts, and (6) modeling activities should be implemented as team exercises. Together these six themes can help guide educators in making sure to design effective learning environments in engineering through the use of mathematical modeling.

2.4 What is computational thinking?

The idea of computational thinking is based on skills commonly associated with computer science (Wing, 2006). The most widely found skills associated with computational thinking in the literature are abstraction, algorithmic thinking, problem-solving, pattern recognition, among others (Kalelioğlu et al., 2016). However, others have looked at computational thinking as something more tangible. For instance, the Royal Society argued that computational thinking is “the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both the natural and artificial systems and processes” (Royal Society, 2012, p. 29). Thus, computational thinking might not require a computer but rather a set of skills to understand computational processes and how computers operate. This includes leveraging tools, many of which are from the field of Computer

Science, to interact with this computation. This recognition of implicit computation in our environment and work requires abstracting information from the physical realm.

Abstraction is a commonly associated skill with computational thinking (Kalelioğlu et al., 2016). For example, Wing (2008) noted that abstraction was “the essence of computational thinking” (p. 3717). Another example in the literature is the one provided by Grover and Pea (Grover & Pea, 2013), who wrote that abstraction is the keystone of computational thinking. In terms of the scope and the various elements within computational thinking, abstraction is the most reported term associated with computational thinking in the literature (Kalelioğlu et al., 2016). Thus, there is a large consensus on the importance of this construct within the framework of computational thinking.

2.5 What are the current definitions of computational thinking?

One of the biggest challenges when engaging with computational thinking literature is the vast array of definitions for computational thinking (Kalelioğlu et al., 2016). Not only have research communities produced definitions (Curzon et al., 2014; Grover & Pea, 2013; Wing, 2008), so have public entities such as Google (Google, 2015), as well as government agencies such as the College Board (CollegeBoard, 2013).

Many of the definitions have significant overlap within them. For example, the concept of abstraction is contained in every definition listed above. Other concepts such as decomposition and algorithm/algorithmic thinking appear in most definitions. Some definitions have more computer science-related processes, such as parallelization, conditional logic, and debugging. These arrays of definition vary from precise definitions such as Grover and Pea (2013), while others are broad, such as Curzon et al. (2014). This diversity in definition leaves the concept difficult to operationalize as there is little consensus on its exact nature and practices.

2.6 How has computational thinking been used in the classroom?

Because of this difficulty in operationalizing computational thinking, the literature surrounding computational thinking is relatively small. Many papers that have been written around computational thinking are mainly position papers and literature reviews (Kalelioğlu et al., 2016). Not only that, but K-12 populations are often the main focus of computational thinking research, while higher education is less studied (Kalelioğlu et al., 2016). Even so, studies have looked to operationalize and use these computational thinking definitions within higher education spaces.

Some researchers and educators have used computational thinking as a structuring tool for their various course designs or the design of activities and assignments (Evia et al., 2015; Jeon & Kim, 2017; Romero et al., 2017). In contrast, some other researchers have looked at how students develop computational thinking skills either qualitatively or by using quantitative computational thinking assessment tools (Korkmaz et al., 2017; Peteranetz et al., 2017; Yuen & Robbins, 2014). These differences mean that computational thinking has been treated as an overarching framework by which to structure other content or the content itself. Wing (2006), in her original proposal, treated computational thinking not as a framework but as a fundamental skill that students should be learning, much like other skills throughout their academic curriculum.

2.7 Summary

To summarize, modeling and simulation have been studied extensively, broadly amongst disciplines and specifically within engineering education. Frameworks such as the MEA and literature reviews have indicated strategies for incorporating modeling activities into the engineering classroom. Studying modeling activities allows researchers to uniquely be able to see aspects of the way a student is thinking about the phenomena of interest.

It is here, within the ways students are thinking about phenomena, that computational thinking becomes a helpful construct. Yet, computational thinking continues to be vaguely defined within the literature. Few methods of actually measuring these constructs have been developed within the literature. More research is still needed to understand developing and assessing computational thinking. To better learn how computational thinking emerges from modeling activities, we must visit how learners think about and understand the phenomena of interest. For this study, we will use the lens of model-based reasoning.

3. THEORETICAL FRAMEWORK: MODEL-BASED REASONING

3.1 Introduction

Model-based reasoning is a construct resulting from "investigations in different domains that have led many cognitive scientists to conclude that much of human reasoning is by means of mental modeling" (Nersessian, 1999, p. 10). This term has been used in many contexts throughout the literature. Model-based reasoning has been primarily studied in physical modeling activities or around programming reasoning into artificially intelligent systems for diagnostic purposes. While both are relevant, the focus of this literature review is broader into the body of research that speaks explicitly to model-based reasoning as a cognitive framework by which individuals use models (both internal mental models and external representations of those mental models) to interpret the world around them.

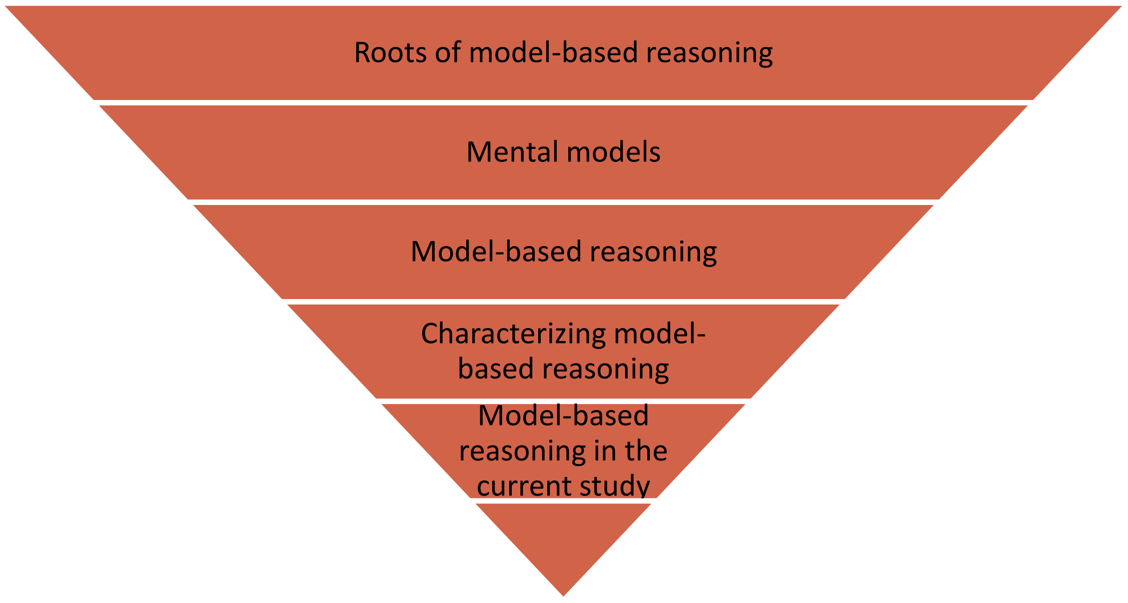
Roots of model-based reasoning
Mental models
Model-based reasoning
Characterizing model-based reasoning
Model-based reasoning in the current study

Figure 2. Overview of topics within the theoretical framework.

Figure 2 above shows the structure of this theoretical framework from broadest to the most specific literature. I will first start by looking briefly into the roots of model-based reasoning. From there, I will dive into the larger body of literature around mental models that inform model-based reasoning as a theory. I will then draw connections between this body of literature and reasoning processes that commonly occur within modeling activities (mental, physical, or otherwise). Following this, I will look at how studies have characterized or measured reasoning processes tailing this discussion with the consensus and disagreement that exists within the literature around model-based reasoning. Finally, all of the discussion will be brought full circle to discuss the methodological and theoretical implications that model-based reasoning has in informing the current study.

3.2 Roots of model-based reasoning

Model-based reasoning has its roots within the broader theory of constructivist learning. Model-based reasoning assumes that learners construct conceptual or mental models that they believe are in alignment with reality (Jonassen et al., 2005). These mental models are built in meaningful contexts and experiences of the learner (Nersessian, 2002). Because of this connection to both constructivism and the role of prior experiences in learning, it is essential to situate model-based reasoning within both bodies of theory.

Constructivism posits that everything we know of the world around us and reality "stems from our own interpretations of our experiences" (Ertmer & Newby, 1993, p. 62). This means that the structures of knowledge that any particular learner has are highly dependent on the experiences of that learner. Essentially the mind operates as a filter, built from life experiences and unique to each learner, by which new experiences and information are passed through (Jonassen, 1991). Thus, within constructivism, the role of personal experiences cannot be overemphasized.

The importance of experience within education is not a new idea, being heavily promoted by educational theorists such as John Dewey (1938). Dewey (1938) wrote that "all genuine education comes about through experience" [p. 25]. This emphasis on experience aligns with the constructivist notion that experiences accumulate into our understanding of the world to create mental filters for new information and experiences. These mental filters are a good introduction to the larger body of literature around mental models and their role in model-based reasoning.

3.3 Mental model theory as a framework for model-based reasoning

Foundational to the theory of model-based reasoning is mental model theory and how internal mental models operate as a way in which humans reason in everyday life. The idea that humans reason through mental models is usually credited as first being proposed by Kenneth Craik (1943) in his book titled *The Nature of Explanation*. He wrote about humans' reasoning through thought experiments using internal cognitive models. This theory stands in contrast to other popular ideas around the same time, such as propositional logic (Inhelder & Piaget, 1958). Propositional logic is a theory that posits that reasoning is carried out by using mental logic to question a proposed way of how the world works. Another theory would be case-based reasoning, which refers to the use of prior experiences to make 'scripts' that are much more situation-specific than the proposed mental model structures, which people use to inform future events (Kolodner, 1992). While similar to model-based reasoning, the mental model framework offers that humans reason by using mental models they test against the world around them.

Mental models are built from collective premises and understandings, allowing people to make leaps within their mental knowledge to understand how new contexts may work and operate (Johnson-Laird, 1983). This process is advantageous for humans to be able to do because this ability allows us to predict the environment around us and simulate how to operate within physical reality (Nersessian, 2002). The mental model theory would posit that everyone uses mental models to navigate the world around them. Evidence suggests that children use mental models to understand the world around them from a relatively young age (Lehrer et al., 1994).

Additionally, with the dawn of the scientific age, mental models have been more and more adapted to being used in scientific and creative ways, rather than just environmental navigation (Nersessian, 1999). Thus, mental models may be inherent to how we learn and reason with the world around us and would be advantageous to us in navigating the world. Pirnay-Dummer et al. (2012) may have said it best when writing of mental models that "in short, they provide the mind with a suitable basis (world-representation) for reasoning and decision-making" [p. 68]. These decision-making and reasoning processes are based on mental models constructed from prior experiences. Exploring this connection between the mental model framework and our reasoning and decision-making becomes the basis for model-based reasoning.

3.4 Overview of model-based reasoning (MBR)

Model-based reasoning is the use of mental models to better understand the world around us. The use of mental models to reason about the world around us was put rather well by Lehrer et al. (1994), who argued that "models are not mental games conducted without reference to the world; conversely, there is no purely perceived world to which models can be applied, because no one has direct access to reality" [p. 219]. This lack of direct access to reality means that to make inferences about reality and reason through new situations, we must use mental models that are intricately tied to reality through experience but will always differ from reality.

What differentiates many other learning theories from model-based reasoning is "the philosophical focus on choice between competing systems rather than construction of the alternatives" (Nersessian, 2002, p. 137). Often, learning is posited as a movement from a wrong way of thinking by choosing a more correct way of thinking (such as propositional logic). In contrast, model-based reasoning as theory would suggest that alternatives to the current way of thinking are constructed and used to make inferences. The results of the mental model simulation are then used as feedback into the mental model and future model creation (Buckley, 2012; Nersessian, 2002). This feedback further develops the mental model, and learning subsequently occurs.

However, rarely are these mental models used in isolation. Instead, they are often co-constructed from multiple more concrete schema or existing mental models referred to as the base and target domains (Ifenthaler & Seel, 2011, 2013). One example that we could use would be the study of mental models of electricity by Gentner and Gentner (1983). The study looked at how individuals reasoned about the nature of electricity using different source domains. For example, if the individual were drawing upon an understanding of electricity as flowing water, they would create a mental model of "flowing electricity" based upon the experience they had of both the source domain (flowing water) and the target domain (electricity). Thus, the mental model created of the situation inherits characteristics of both the water and the electricity. The learner's mental model or the developed external model may exhibit properties of smooth flowing water through pipes. Alternatively, if subjects understood the particulate nature of electricity much like a moving crowd of people, the nature of the source domain changes, and thus the created mental model does likewise (Gentner & Gentner, 1983). This relationship between the source domain and the target domain can be seen in Figure 3.

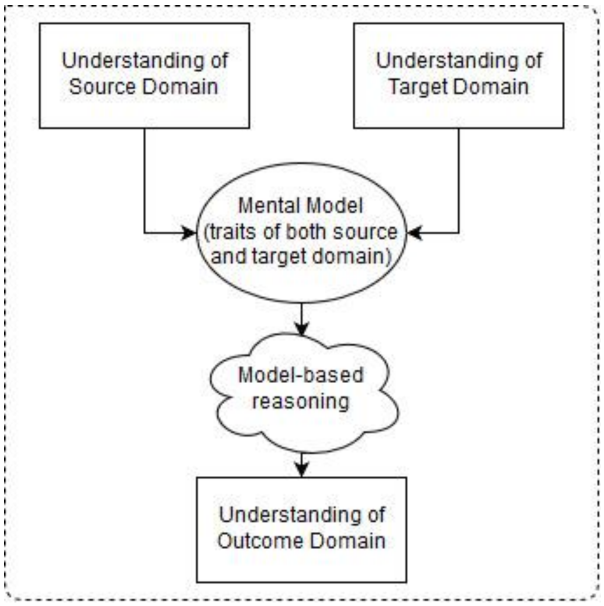
Understanding of Source Domain
Understanding of Target Domain
Mental Model (traits of both source and target domain)
Model-based reasoning
Understanding of Outcome Domain

Figure 3. The combination of source and target domains using model-based reasoning.

Although a simple example, it presents the idea that model-based reasoning supposes that mental models that demonstrate the world around us are subject to the constraints of the information in which they are operating, both the source and target domains (Nersessian, 2002). These source and target domains are often constrained based upon the context by which a person has experienced them, and mental models are bound inseparably within the learner's personal experiences (Ifenthaler & Seel, 2013; Johnson-Laird, 1983; Nersessian, 2002). Thus, personal experiences feed directly into the mental models that are created.

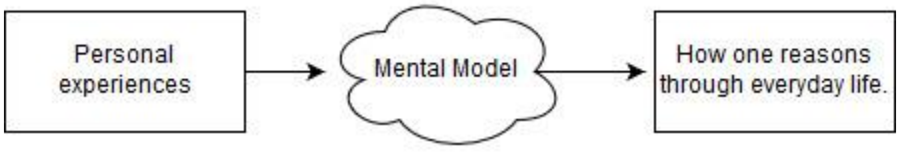

Figure 4. Constructivist model-based reasoning.

Because mental models are based on personal experiences, as demonstrated in Figure 4, they are, in essence, probabilistic models (Ifenthaler & Seel, 2013). The idea that humans would think in terms of probabilistic models, meaning based on the likelihood of the outcome, makes sense. It allows us to anticipate our environment as we experience life (Nersessian, 2002). Thus, mental models create the ability to deductively and inductively reason about new domains and experiences we encounter. When these inferences based on our mental models conflict with the world around us (the experiences we have in life), that creates a feedback loop in which we must adjust our understanding of the world by adjusting our mental models or rejecting reality (Pirnay-Dummer et al., 2012). However, model-based reasoning consists of multiple facets that must be explored as numerous types of reasoning are inherent to reasoning with mental models.

3.5 Model-based reasoning processes within modeling activities

Much of the literature and research on model-based reasoning supports the idea that multiple distinct reasoning processes occur during the model-based reasoning process. Nersessian (2007) writes that three criteria must be met for something to be considered model-based reasoning: (1) it must involve either the construction or retrieval of a model, (2) manipulation of a model to make inferences and predictions about the phenomenon of interest and (3) the inferences must be applied to either a single model or an entire class of models. Based on these characteristics, three main reasoning processes occur within model-based reasoning and are found throughout the literature: analogical modeling, visual modeling, and simulative modeling (Nersessian, 1999).

Analogical modeling is commonly associated with reasoning processes within mental models and model-based reasoning (Hesse, 2000; Ifenthaler & Seel, 2011; Lehrer & Schauble, 2003; Nersessian, 1999, 2002, 2007; Seel, 2003). In the literature, analogical modeling is ultimately a general form of abstraction where an analogy is drawn from a target domain and a source domain to understand one or both (Hesse, 2000; Nersessian, 1999, 2002). An analogy is a critical piece of model-based reasoning. It allows us to better understand a target domain by constructing a model based on a source domain with which we are much more experienced (Seel, 2003). A relatively simple example of this type of model-based reasoning might be using the solar system as an analogy of how an atom is structured (Lehrer & Schauble, 2003). We may very well be familiar with the solar system while unfamiliar with the atom (or we may have premises about each individually). We then create a mental model of how an atom operates through analogy, with our mental model taking on aspects of the solar system (as well as any notions we have of the atom). Another example would be Newton deriving his fundamental ideas about gravity when drawing analogous relationships between “what a projectile and a planet have in common in the context of determining motion” (Nersessian, 2002, p. 144). Thus, analogy is a critical piece of model-based reasoning because it allows us to make inferences about new domains and new problems based on previous understandings of a source domain.

Another form of reasoning used within model-based reasoning found in the literature is visual modeling, such as using pictures, diagrams, or other visual/structural representations of the information either internally or externally (Johnson-Laird, 1983; Lehrer et al., 1994; Leutner et al., 2009; Nersessian, 1999, 2002; Quillin & Thomas, 2015). The most apparent aspect of visual modeling would be imagery to reason through a target domain (Nersessian, 2002). However, visual modeling is intended to represent further reasoning than just utilizing mental imagery. Visual modeling means that model-based reasoning uses mental models or external models that catch the very structure of the target domain (Johnson-Laird, 1983). Thus, visual modeling means that mental models keep specific modalities of the real-world phenomenon, one of which can be visual aspects of the target domain. One example of visual modeling for model-based reasoning would be the famous physicist Michael Faraday using force lines to draw out how forces were locally and directionally working around a magnet (Faraday, 2004). This external representation of a mental model was used to understand and communicate how these forces were working. In essence, notational systems (such as force lines) are being used to support model-based reasoning. They allow for information to be organized and more easily incorporated into the phenomena of interest by capturing modalities of the mental model. The importance of visual modeling in model-based reasoning was shown by Lehrer et al. (1994). When studying model-based reasoning in children, students struggled to understand why certain sums of two random numbers between one and four were more likely than others. Had the students used an external representation of the information recorded with mathematical notation, they may have noticed that certain sums have more combinations of numbers than others. The literature gives many examples of visual modeling as a form of model-based reasoning, such as vector diagrams in physics, circuit diagrams in electrical engineering, or bonding diagrams in chemistry (Quillin & Thomas, 2015).

Finally, the literature supports a third type of reasoning that is used in model-based reasoning referred to in the literature as simulative modeling, although some of the literature around model-based reasoning uses the term thought experiments when explicitly referring to the manipulation of mental models (Buckley, 2012; Ifenthaler & Seel, 2013; Nersessian, 1999, 2002; Seel, 2003). Simulative modeling takes mental or physical models together with the understanding of the constraints to which the model is subject (Nersessian, 2002). Seel (2003) described this type of model-based reasoning when writing:

> This occurs when an individual interacts with the objects involved in a situation in order to manipulate them mentally in such a way that the cognitive operations simulate specific transformations of these objects that may occur in real-life situations. Such simulation models operate as thought experiments, producing qualitative inferences with respect to the situation to be mastered (p. 61).

Thus, simulative modeling manipulates mental models to understand better and make inferences about how the real world operates. This type of model-based reasoning, simulating a mental model based on real-world constraints, must not be done entirely in mind. Dennett (2000) wrote that "just as you cannot do very much carpentry with your bare hands, there's not much thinking you can do with your bare mind" (p. 17). This means that for simulative modeling, it may often be helpful to have external representations of the information to encode aspects of the mental model. Externally representing the mental model is done to understand better the mapping between the mental model and the external reality (Nersessian, 2007). External representations can commonly be mathematical or computational in models. We encode information from our mental models into these tools to simulate an environment (that is still distinctly separate from the actual real-world around us). Results from these simulations are then on a feedback loop in which they help us change or concretize our internal mental models of the phenomenon of interest (Quillin & Thomas, 2015).

It is worth emphasizing that the cornerstone of modeling activities in education classrooms is often the building of the actual model (R. Lesh et al., 2000). Notational systems are often used in building the model, whether these be a mathematical, programming language, or just drawing out a picture (such as in Faraday's case). Nersessian (1999) wrote that "the mental models perspective hypothesizes that the external visual representations support the construction of an internal model" (p. 17). Therefore, visual and simulative modeling represents the most explicit connection between theories of model-based reasoning (dealing with mental models) and the external representations of information we call mathematical or computational models. These external models allow us to organize our thoughts and structure cognitive information in ways that enable us to further develop our understanding of the target domain under investigation. Much like writing out combinations of numbers may help children develop better mental models of probability, creating mathematical and computational representations of information and simulating them may further concretize mental models and essentially allow students to reason between their mental model and reality.

These reasoning processes can be used together and often make inferences about the world around us. For example, let us use a relatively common (although somewhat outdated) metaphor of the engineering education pipeline (Tillman, 2013). This pipeline uses analogous modeling that allows us to use our mental models of a leaky pipeline and give its attributes to STEM education, understanding it may leak and that students (water) flow through it. Often, we draw out this diagram of a pipeline showing where engineers enter or exit the education system, thus using visual modeling. From there, we can use simulative modeling to bring constraints to how we understand the world to conjecture and make inferences why students may enter or leave the engineering pipeline (this information in the mental model embedded within both our understanding of pipes as well as our understanding of the STEM education process in the United States). Thus, these processes are not mutually exclusive but are used hand in hand to reason through problems.

As far as explicit modeling activities in the classroom, everyday actions within modeling activities lend themselves to the model-based reasoning processes above. Modeling activities are complex initiatives and ultimately one of the most challenging aspects of STEM professions, specifically within engineering (Gainsburg, 2006). Modeling activities allow students to externalize their internal mental models. Jonassen (2009) proposed that “constructing computational models of the world using computer-based modeling tools can serve to externalize learners’ mental models of the phenomena that they are studying” (p. 56). This relationship means that mathematical and computational models directly reflect an internal mental model within the student. Nersessian (2007) suggested that analogical modeling, visual modeling, and simulative modeling can be broken down within modeling activities to allow individuals to abstract a phenomenon across cases, simulate outcomes, evaluate how good a model is as a predictor of the real-world, and adapt the model to necessary constraints. Another example of model-based reasoning within modeling activities tasks in the classroom would be a study by Ifenthaler and Seel (2013). They argued that tasks common to modeling practices such as simulation, diagnosis, planning, training, controlling, and debugging use model-based reasoning. Thus, model-based reasoning is helpful in both building an understanding of reality in its current state with activities such as evaluation, debugging, or diagnosis; as well as making inferences as to the manipulation of reality into some future form with activities such as simulating outcomes, abstracting across cases, or planning a solution. These activities are embedded within modeling activities at an educational level (Louca & Zacharia, 2012). We will investigate this close relationship later.

3.6 Characterizing and measuring model-based reasoning processes

There are many ways of characterizing model-based reasoning within the literature; however, this characterization does not come without limitations. The ultimate struggle with describing or measuring any cognitive process is that we are limited to the artifacts and products of the participants under study, usually relying on secondary indicators of the cognitive process as opposed to a direct measure of the participants' physical reality during a reasoning process. In any case, there are multiple examples within the literature on how to characterize or measure model-based reasoning processes within participants and learners. While both qualitative and quantitative methods are used to characterize model-based reasoning in the literature, many studies focus on more qualitative data such as interviews, think-aloud interviews, and classroom observations. A minority of the studies used more quantitative data such as time on task, explicitly scoring reasoning tasks, or looking at performances based on different education interventions.

Some of the studies intended to measure model-based reasoning tasks by giving students reasoning tasks and measuring time to complete or process to complete (Ifenthaler & Seel, 2011, 2013; Pirnay-Dummer et al., 2012). A variation of this was attempted to measure model-based reasoning skills by giving students reasoning tasks looking at the correctness of the students' response, as well as mean response time in solving reasoning problems (Vandierendonck, 2002). Reasoning problems were also given to students to look at percentages of predicted behavior versus actual behavior to characterize how reasoning behaviors matched learners' expectations using mental models (Johnson-Laird et al., 1999). However, these were rarely looking at the actual reasoning process, often looking typically at the performance given different forms of "treatment" based on model-based reasoning instead of traditional logical reasoning. Thus, these were not direct measures nor characterizations of model-based reasoning, but rather indirect measures of the potential effects of teaching with curriculum or using a specific type of format to reasoning questions.

Alternatively, interviews can be used to characterize how students use model-based reasoning. One example would be the one from Gentner and Gentner (1983). They used interviews to understand how students' understanding of electricity changed based on the source domain in which they constructed their mental model (flowing water vs. moving crowds). Interview or think-aloud interviews are typical for capturing how students reason through a problem (Gentner & Gentner, 1983; Lehrer et al., 1994; Pirnay-Dummer et al., 2012; Raghavan & Glaser, 1995; Taylor et al., 2003). In previous studies, interviews and think-aloud interviews allowed the researchers to understand exactly how the students worked through the given problem or the reasoning task. However, there was caution expressed against interviews by Jonassen et al. (2005), who wrote that "the analysis of interview and conversation protocols is very difficult and time-consuming and is plagued with reliability problems" (p. 16). Thus, one must proceed cautiously. It may be challenging to understand cognitive processes simply from what a participant says in an interview or that the interviewee is even accurately describing how they are thinking through a problem.

One interesting study used interviews and student surveys to understand how students characterized internal mental models and how these mental models developed within an astronomy intervention (Taylor et al., 2003). Once characterized, researchers were able to look at how mental models developed throughout the intervention by asking questions related to the mental model at three different points during the research and mapping these responses as learning gains as students' mental models developed. A similar course of action could characterize and measure model-based reasoning by understanding how mental models develop through reasoning practices between external and internal models, looking at similarities and differences. Hesse (2000) proposed that mental models can be updated based on reasoning between the external and internal models and updating the mental model accordingly.

In addition to interviews and quantitative data, observation of modeling tasks may help understand how students leverage model-based reasoning during activities. There are multiple studies and papers that reference using observations or the benefits of observations in characterizing model-based reasoning (Harrison & Treagust, 2000; Jonassen et al., 2005; Nersessian, 2002; Raghavan & Glaser, 1995; Taylor et al., 2003). One study of particular interest by Raghavan and Glaser (1995) used in-class video-taped observations to evaluate the effectiveness of a model-based reasoning curriculum implementation. Then using these observations, investigators rated students on different levels of model-understanding to look at trajectories of reasoning with models through the duration of the curriculum. Observations may avoid some bias in writing interview questions and capturing behavior unknown to the participant; however, they may lack the detail of an interview (Driscoll, 2011).

This idea of change over time was used in varying contexts throughout the literature to look at model-based reasoning and the change of mental models. For example, Lehrer (1994) looked at the difference between how students reasoned in making predictions of probability functions (such as spinners spinning or dice rolling) and compared predictions to the actual probability distributions. Different scenarios were given to the students sequentially, with the researchers investigating how the students' predictions changed over time and the reasoning for doing so. The researchers then built a cognitive mental model development sequence to provide evidence that the students were using mental models to understand how probability exercises worked and how previous activities were feeding into the mental model to update predictions. This same idea of investigating model-based reasoning through the changes in mental models was echoed by Ifenthaler and Seel (2013). They explored how students used model-based reasoning to solve various reasoning tasks by looking at the quality of the strategy used to solve any given problem. Participants were given varying tasks over many weeks. They were asked to solve the reasoning task and write out why the strategy was used. The quality of the solutions, the quality of the strategy taken, and a logical reasoning rating were given to each solution from the student. The study's goal was to understand how often students developed new mental models versus how often students used reasoning processes to update previous mental models.

3.7 Consensus and disagreement about model-based reasoning

There are multiple areas of consensus and a few areas of disagreement in the literature around model-based reasoning. Many of the disputes have to do with the cognitive aspects of model-based reasoning, how mental models are stored and operationalized within memory, and the temporal nature of mental models. Although there is consensus about a relationship between mental models and schema, the nature of the relationship is vague and ambiguous throughout the literature, with some seemingly disconnects between how the two constructs relate to one another. Additionally, while many studies agree that visual modeling is essential to model-based reasoning, there are some disagreements about how useful this reasoning process is in helping students learn content about the world around them. Finally, the literature is ambiguous around the term model-based reasoning. It is commonly used to describe artificially intelligent systems with an increased research focus on artificial intelligence.

There is consensus within the literature that memory and prior experiences play a significant role within model-based reasoning (Gentner & Gentner, 1983; Leutner et al., 2009; Nersessian, 1999, 2002, 2007; Quillin & Thomas, 2015; Vandierendonck, 2002). However, where mental models are stored and how they are cognitively operationalized is up for debate. The body of literature indicates disagreements on how mental models and model-based reasoning are a product of long-term memory versus operationalized working memory (Nersessian, 1999). For example, Mayer (2009) proposed that mental models are constructed within the working memory at a given moment with prior knowledge being used that is pulled from the long-term memory to use model-based reasoning about the world. However, other studies such as Ifenthaler and Seel (2013) had students iterate based on mental models over weeks and months. This indicates that mental models do not exist solely within the individual's working memory. One interesting example of mental models and memory is the idea from Gentner and Gentner (1983) of using a model of flowing water to understand electricity. Are these two domains of flowing water and electricity called upon each time one wants to reason about the behavior of electricity? Or is this mental model that has been created stored in longer-term memory so that every time one reasons about electricity, the association with flowing water is already built-in? This seems to be a fundamental question and source of uncertainty throughout the literature.

Relatedly, the relationship between schemas and mental models is somewhat ambiguous within the literature in terms of how each is used to perform model-based reasoning. Schemas are longer-term understandings that are structural blocks of knowledge that people understand certain phenomena (Ginsburg & Opper, 1988).

Others in the literature indicate that schema are pulled together to build mental models in the working memory, insinuating that mental models may be short-term constructs and not held within the long-term memory (Nersessian, 2002). Thus, this disagreement in the literature is tied to the dispute regarding mental models and memory. The literature also describes schema-based reasoning as an alternative to model-based and case-based reasoning (Krampe & Lusti, 1997; Stroulia et al., 1992). Whereas model-based reasoning is used to describe ill-defined domains to the learner/user, schema-based reasoning uses “highly situation-specific knowledge” (Stroulia et al., 1992, p. 3). While not necessarily in disagreement, the connection between schema and mental models and, consequently, schema-based and model-based reasoning could be more clearly defined for a complete understanding of mental model theory to be operationalized.

Another area of disagreement in the literature is to what extent external modeling is critical to the model-based reasoning theory. There is a general consensus that external models play a large part in the model-based reasoning process (Johnson-Laird, 1983; Lehrer et al., 1994; Leutner et al., 2009; Nersessian, 1999, 2002; Quillin & Thomas, 2015). However, there is debate about whether external representations of information help reasoning processes with internal mental models. For example, some studies have found that using drawings and external models helps offload some cognitive processing and allows students to develop further their understanding of a phenomenon (Harrison & Treagust, 2000). Other studies have found that creating external models may create additional cognitive load (Jonassen et al., 2005; Leutner et al., 2009). This is important within model-based reasoning because specific reasoning processes such as visual and simulative modeling may often use external representations of information to help organize and offload some cognitive processing from the mind. Many studies indicated that these external models often allow for refinement of the internal mental model by offloading processing from the mind and allowing for a more organizational structure to information (Jonassen, 2009; Jonassen et al., 2005; Nersessian, 1999, 2002; Quillin & Thomas, 2015). While most of the literature seemed to indicate that these external representations were helpful, there seem to be cases in the literature where they are not beneficial to learning.

Within the body of literature, model-based reasoning is also used to describe the use of machines or artificially intelligent systems for diagnostic purposes (Davis & Hamscher, 1988; De Koning et al., 2000; Koton, 1985; Rich & Venkatasubramanian, 1987). While this is not the explicit nature of the discussion here, the method by which artificial intelligence (AI) systems diagnose problems is similar to how humans construct mental models. This often occurs when the machines use existing models to predict future events, much like how humans use mental models to navigate their environment. This, along with other types of reasoning (such as case-based reasoning), has enjoyed a surge of research work given the current research interest in artificially intelligent systems.

3.8 Methodological implications of model-based reasoning

From the literature, there are multiple ways in which researchers have attempted to measure and characterize reasoning processes. Explicit quantitative measures in the literature were challenging to find, as most looked at the time on task or looked at performance on various reasoning tasks (Ifenthaler & Seel, 2011; Vandierendonck, 2002). Additionally, the questions these studies were asking are fundamentally different than the questions this study aims to answer. While previous work has looked for ways in which model-based reasoning can be measured, this study will look at how model-based reasoning will elicit, through modeling, computational thinking.

Specifically, the current study aims to use model-based reasoning as a theoretical framework for locating computational thinking structures. Thus, it becomes more of a qualitative characterization process instead of a measurement process. Multiple studies in the literature looked to characterize model-based reasoning often using interviews, think-aloud interviews, and classroom observations (Gentner & Gentner, 1983; Lehrer et al., 1994; Pirnay-Dummer et al., 2012; Raghavan & Glaser, 1995). One of particular interest would be the study by Gentner and Genter (1983), who looked at how mental models and reasoning around a process changed based upon the source domain. Similarly, I use classroom observation data for our study to characterize how students are reasoning through the model and simulation process. Because computational thinking is often a latent, not explicitly stated process, observations may be more suitable for characterization when possible. Observations allow for characteristics unknown to the participant (latent behaviors) to be captured more readily (Driscoll, 2011).

3.9 Theoretical implications of model-based reasoning

Of interesting note and aligned with the intent of the present study is the following quote about the use of external computational models through technology to enhance and develop internal mental models:

> Further, we argued that the most effective way to use technologies to foster mental model development is through the use and construction of computational models using model-based software (Jonassen, 2009, p. 72).

The idea that model-based reasoning promotes the development of an internal mental model through the use of an external computational model is the core of the theoretical framework for the current study. Model-based reasoning is the way and purpose for which external mathematical and computational models are created and operated within. Students build these external models as they reflect internal mental models held by the student. Even when students work in groups, model-based reasoning can occur based on the collective mental models.

> Group or collaborative mental models are those that are socially co-constructed by groups of individuals who are collaboratively focused on the same meaningful task (Jonassen, 2009, p. 54).

Thus, even when operating within a group and making external models, groups are co-constructing external representations based on their internal mental models.

Through a complete computational modeling and simulation process, students must create multiple iterations of an external model (each a representation of the information), going between the physical referent (problem statement) to the mathematical model to the computational model and then back again to the physical referent (Magana et al., 2012). Figure 5 below gives an overview of this process.

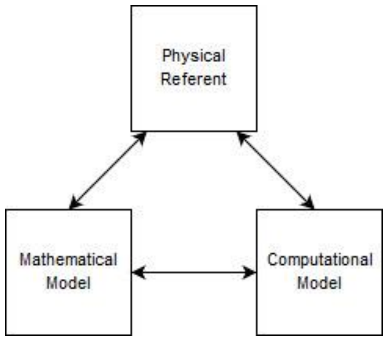

Figure 5. Overview of the computational modeling and simulation process.

The theoretical framework of model-based reasoning connects to this modeling and simulation process because students are using model-based reasoning to navigate at each transition within the modeling and simulation process. The model-based reasoning they exhibit during the transition is limited by and will exhibit characteristics of their mental models of the source domain (where they are transitioning the information from) and the target domain (where they are transitioning the information). Figure 6 below demonstrates an example of this relationship at one step in the modeling and simulation cycle, put into the context of the current study (building computational models of food sterilization).

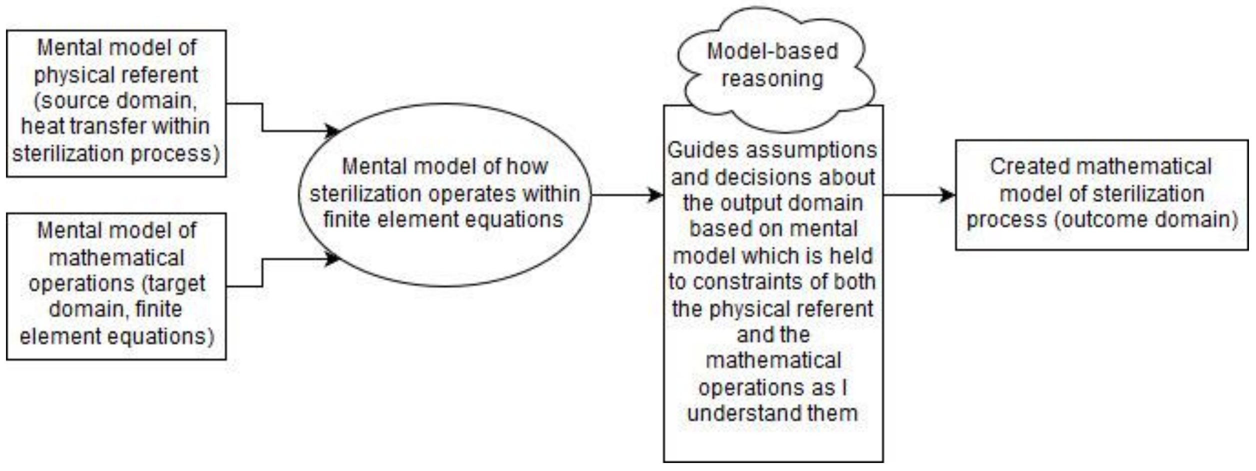
Mental model of physical referent (source domain, heat transfer within sterilization process)
Mental model of mathematical operations (target domain, finite element equations)
Mental model of how sterilization operates within finite element equations
Model-based reasoning
Guides assumptions and decisions about the output domain based on mental model which is held to constraints of both the physical referent and the mathematical operations as I understand them
Created mathematical model of sterilization process (outcome domain)

Figure 6. Model-based reasoning situated within one branch of the modeling and simulation cycle.

The theoretical framework of model-based reasoning allows us to understand that complex thinking patterns such as those described by computational thinking can be found in the student's reasoning based on the reflections of their internal mental models. Furthermore, the construct of computational thinking described in the literature (Selby & Woollard, 2013; Wing, 2006) has alignment within model-based reasoning. The central hypothesis of this study is that through model-based reasoning, as students engage with external mathematical and computational models, we can investigate how computational thinking is used. To demonstrate this alignment, I ground my approach on the list of reasoning processes that Nersessian (2007) listed as pieces of model-based reasoning:

> “**abstraction**: limiting case, generic, idealization, **generalization**; simulation: inferring outcomes or new states via model manipulation (mental or physical); **evaluation**: goodness of fit, explanatory power, implications (empirical, mathematical); and adaptation: constraint satisfaction, coherence, other relevant considerations.” (p. 706-707).

Conversely, the list of computational thinking-related practices is guided by Curzon et al. (2014) as algorithmic thinking, **evaluation**, decomposition, **abstraction**, and **generalization.** There is no doubt that there is an overlap between the model-based reasoning process and computational thinking practices. One can see that current definitions of both have overlapping terms. The two missing elements (algorithmic thinking and decomposition) are primarily physical manifestations that occur within the actual building of the external model during a classroom modeling activity.

3.10 Summary

This study will look into the reasoning that students use throughout the modeling and simulation cycle to understand the different ways in which students use computational thinking practices and how modeling and simulation practices can be used to elicit these crucial computational thinking behaviors. Model-based reasoning gives the research study a lens to understand how students are externalizing their internal mental models and the types of outcomes that can be expected to be found throughout this externalization process. Yet the modeling process is long, consisting of multiple tasks such as planning, programming, and evaluating. This study will look at how students are using computational thinking as they reason about their own and other students' models.

4. METHODOLOGICAL FRAMEWORK

4.1 Introduction

This chapter overviews the methodological framework used to frame the methods to approach the research question to better structure this study. For the current research, design-based research is used as the methodological framework. Design-based research (DBR) allows the study to contribute to broader theory while also allowing the delivery of practical results. Additionally, the naturalistic approach of design-based research allows the study to collect rich and authentic data to answer the research question.

4.2 Design as a form of education research

Design research has gone by many names in the literature on education, such as design-based research, design experiments, design research, development research, developmental research, and formative research (Wang & Hannafin, 2005). Even though there are many names, this form of research is a method that allows for designed educational interventions to be directly implemented and researched within a naturalistic setting, often in a classroom. This natural setting provides DBR the ability to contribute both to the practical application and local context and the larger theoretical body of literature.

The central tenet of DBR is that educational data and findings are embedded inseparably from their context and that results from educational studies are always dependent on the factors in which the data was collected (DBRC, 2003). There are multiple advantages to using design-based research instead of other methods available to those engaging in qualitative inquiry. First, it allows for various data analysis methods rather than a strictly confined data analysis procedure (Wang & Hannafin, 2005). This flexibility allows for the use of multiple different forms of data rather than being strictly relegated to any specific form. Additionally, design-based research contributes directly to the classroom by delivering pragmatic research results (DBRC, 2003; Wang & Hannafin, 2005).

While the opportunity to collect data directly in context is an approach that can immediately result in practical applications, there are several limitations to the method. Some common critiques of design-based research are that it cannot adequately contribute to theory and that the results of the studies cannot be generalized to more contexts than the one studied (A. E. Kelly, 2004). Additionally, the method has often been criticized for lacking a clear methodology (A. E. Kelly, 2004).

However, design-based research (DBR) is best defined by the characteristics it exhibits: (1) *Being situated in a real educational context,* (2) *Focusing on the Design and Testing of a Significant Intervention,* (3) *Using Mixed Methods,* (4) *Involving Multiple Iterations,* (5) *Involving a Collaborative Partnership Between Researchers and Practitioners,* and (6) *Evaluation of Design Principles* (Anderson & Shattuck, 2009). While this is not as structured as other qualitative or quantitative methods, it allows the researcher a framework and principle to conduct their research study. While these critiques hold some merit and a high level of importance, there are many reasons to believe that the strength of these critiques is limited if design-based research studies are done with enough rigor. Having trajectories of multiple studies is one way to address the limited generalizability of these methods by setting up similar interventions across multiple contexts and temporal distance (DBRC, 2003). If results during the research trajectory point towards similar or different results, the trajectory will begin to paint a picture to the larger research community of how generalizable the results are.

Finally, having predefined structures such as design and theoretical conjectures allows the researcher to clearly define how and in what ways their research contributes to the local context and the larger body of literature (Sandoval, 2014). Conjecture maps overview how to structure design-based research studies by (1) aligning the overall conjecture being made by the study, (2) defining the ways that a design will elicit certain behaviors and generate artifacts from the students, and (3) identifying the theoretical conjectures of what outcomes the behaviors and artifacts will elicit in the students.

4.3 Implications of design-based research for the study design

The methodological framework for the current study is design-based research, which aims to collect data that is embedded within its naturalistic context (DBRC, 2003). This approach is appropriate when paired with a case study in that it allows a design to be bounded to the phenomenon under study. In our case, this is computational thinking within a designed modeling intervention. When conducting design-based research, there are multiple parameters of the research that must be defined. Wang and Hannafin (Wang & Hannafin, 2005) defined design-based research by the following characteristics:

(1) Pragmatic: This study aims to add to both theory and practice, a staple of design-based research (Wang & Hannafin, 2005). This study will add to the theory of computational thinking and elicit it in educational environments. The study will add to practice by delivering design principles into a capstone engineering course.
(2) Grounded: The study's design is grounded in multiple bodies of theory and frameworks such as model-based reasoning, modeling-based learning, model eliciting activities, and productive failure.
(3) Interactive/Iterative/Flexible: The researchers worked intimately with the instructor to implement the designed intervention when conducting this study. Multiple rounds of data were collected, although only one iteration of the design is presented in this study.
(4) Integrative: This study uses qualitative analysis (thematic analysis) to understand how the themes were elicited and changed throughout the modeling intervention, integrating multiple forms of data such as classroom artifacts and audio of student group sessions.
(5) Contextual: The research is embedded within a naturalistic context of a capstone engineering course. The design then gives general principles for modeling interventions in other contexts.

Given these descriptions, it is clear that this study fits the criteria outlined by Wang and Hannafin regarding the characteristics of design-based research. Yet, design-based research does have both advantages and disadvantages.

4.4 Overview of design conjectures

One way to ensure rigor within design-based research is to build a conjecture map. Conjecture maps are tools for demonstrating how different intervention elements align with various conjectures posed by the research (Sandoval, 2014). There are multiple advantages to using a conjecture map. One is that conjecture maps make explicit and attempt to answer one of the central tensions within DBR, that DBR has a "dual commitment to improving educational practices and furthering our understanding of learning processes" (Sandoval, 2014, p. 20). Conjecture maps provide design conjectures that outline how to improve educational practices and theoretical conjectures that aim to further our understanding of learning at a basic level (Sandoval, 2014).

There are multiple elements to a conjecture map. The first box provides a high-level conjecture about the overarching study and the aims of the learning/outcomes of the study. The second box overviews the embodiment of the learning environment, showing the tools, tasks, participant structures, and any discursive practices. The third box looks at mediating processes, which are interactions and artifacts resulting from the embodiment. Between these two are the design conjectures, which posits that "if learners engage in this activity (task + participant) structure with these tools, through this discursive practice, this mediating process will emerge" (Sandoval, 2014, p. 24). In many ways, design conjectures align with the tension in which DBR constantly finds itself, addressing the practical implications and design principles for pragmatic results.

The second part of a conjecture map looks at the design's theoretical conjectures. The last box in a conjecture map is the outcome space that includes any learning, interest, motivation, or outcome one wants to investigate, given the current design. Between the mediating processes and the outcome are the theoretical conjectures which put forth the statement, "if this mediating process occurs, it will lead to this outcome" (Sandoval, 2014, p. 24). These conjectures deal with another tension in which DBR finds itself, contributing to a more extensive theory within the naturalistic context. Theoretical conjectures allow DBR researchers to understand fundamental learning questions by understanding the relationship between the mediating processes and the outcomes.

4.5 Summary

Design within educational research may go by many names. However, design-based research (DBR) as used in this study is a clear methodological framework that allows the current research to contribute to theory and practice. And while DBR has many advantages, there are some limitations to the method in terms of generalizability. These can be overcome by using conjecture maps that clearly define the structure of the design and the theoretical/design conjectures the intervention hopes to evaluate.

5. LEARNING DESIGN

5.1 Introduction

Under a methodological framework of design-based research, a learning design is required to test the design and theoretical conjectures and fulfill the intended outcomes. In this chapter, pedagogical frameworks used to create the design are overviewed, including models and modeling, the model-eliciting activity (MEA), and productive failure. These frameworks are then integrated into a single learning design, aligned via a conjecture map to structure the learning environment, theoretical and design conjectures, and outcomes expected from the learning environment. The chapter overviews the timeline of the intervention and various activities that the students are expected to perform as part of the learning design. Finally, the chapter looks at the context and learners into which this study specifically will implement the design, focusing on student needs during the learning process.

5.2 Process design: Models and modeling

There are multiple timelines and breakdowns of how modeling activities should be implemented within the engineering classroom. For example, Magana (2017) developed a learning progression derived from a four-part modeling and simulation process: construct models, use models, evaluate models, and revise models. Additionally, Shiflet and Shiflet (2006) divided the modeling cycle into a six-part process: analyze the problem, formulate a model, solve the model, verify and interpret the model's solution, report on the model, and maintain the model. Louca and Zacharia (2012) defined the modeling process as modeling-based learning (MbL), consisting of four distinctive phases: collection of observations and experiences, construction of the model, evaluation of the model, and revision of the model. While these are only a few examples, all point to a general framework by which modeling and simulation can be learned in engineering that we will continue to refer to as modeling-based learning (MbL).

MbL is a process by which students convert mental models into concrete models by transforming physical phenomena into abstractions, often mathematical (Louca & Zacharia, 2012). MbL is not new to engineering, with developments such as the model-eliciting activity (MEA) being heavily researched in engineering contexts (Diefes-Dux et al., 2004; Hamilton et al., 2008).

However, modeling through the lens of computation adds a layer, where the student must move the model from mathematical to computational abstractions. MbL has two unique phases: first the model is built, and then the model is evaluated and revised (Louca & Zacharia, 2012). Both stages of MbL by nature incorporate aspects of computational thinking presented by Weintrop et al. (2016); analyzing data, visualizing data, constructing computational models, among many others. Thus, MbL pedagogical strategies are uniquely positioned to promote and assess computational thinking at the undergraduate level in engineering education. In addition, computational modeling involves movement between mathematical and computational abstractions, thus allowing one to observe the fluency of this transition.

This research will utilize a model-based learning framework to understand how computational modeling activities in discipline-specific contexts can promote and support computational thinking in undergraduate engineering students. In addition, the research will support an in-depth understanding of the nature of computational thinking and how students effectively move between both mathematical and computational abstractions.

5.3 Structure design: The model-eliciting activity (MEA)

MEAs have been heavily studied within educational contexts, specifically beginning in the mathematics education literature with Lesh and colleagues (R. Lesh et al., 2000; R. A. Lesh & Zawojewski, 2007; R. Lesh & Harel, 2003), and has since been translated and studied within the engineering education literature by Diefes-Dux and colleagues (Diefes-Dux et al., 2004, 2006, 2013; Hjalmarson et al., 2006). MEAs are generally tied to six principles, although some of the names can differ based on application. Lesh et al. (2000) assert that MEA development is linked to six principles:

(1) model construction principle: the activity results in a model of a physical phenomena
(2) reality principle: the activity is embedded in a realistic context
(3) self-assessment principle: the activity encourages students to look at the usefulness of their own and other solutions.
(4) construct documentation principle: the activity asks students to explicitly state why and how they are thinking about the situation as they are.
(5) construct shareability principle: the activity asks the student to develop a model which is useful to others.

(6) effective prototype principle: the activity asks the student to provide a prototype that is useful in multiple situations.

These six principles work together to create learning environments and activities where students can build realistic models while documenting the entire process.

MEAs are a valuable tool in identifying student misconceptions within disciplinary content (Self et al., 2008). Additionally, MEAs can correct student misconceptions of the underlying physical phenomenon (Moore et al., 2013). They, and more broadly modeling activities, allow students to move through multiple different representations of the information, creating an environment to learn the physical concepts more deeply. However, MEAs can create challenges on the assessment end, with the open-ended nature making grading assignments much more of a gray area (Diefes-Dux et al., 2012). For example, with these complex solutions, instructors tend to give redundant feedback and lack praise, which can negatively affect the students' future performance (Jung et al., 2015). However, the use of MEAs in the classroom has a positive effect on students learning and on instructors who, through the use of MEAs, often move towards more student-centered teaching methods (Moore et al., 2015).

5.4 Pedagogy design: Productive failure

Learning from failure is not a new concept; however, research has primarily focused on learning through student success (Kapur, 2008)**.** Kapur and colleagues have shown how giving minimal structure to problems leads to increased performance in future activities, terming this pedagogy productive failure (Kapur, 2008, 2010, 2011; Kapur & Bielaczyc, 2012). Productive failure as a teaching pedagogy has two distinct phases: (1) creation and exploration of representations and solution methods (RSM's), and (2) comparison and contrasting of one's solution to those of other peers (Kapur & Bielaczyc, 2012). Students are asked to look at different solution pathways with minimal help from the instructor during the first phase. In this phase, instructors are told not to answer questions but encourage students to do their best to solve the problem at hand. Students are asked to compare their own solutions with expert solutions to the problem in the second phase. In this phase, instructors are told to encourage students that there are no right or wrong answers to a problem.

However, productive failure is not an entirely new concept on its own. It is derived from a broader class of learning pedagogies based on inquiry-based learning through lived experience, promoted by famous theorists previously such as Dewey and Piaget (Dewey, 1938; Ginsburg & Opper, 1988). These lines of constructivist theory encourage educators to consider education as a combination of lived experiences where students are asking questions that lead to new insights and pathways as they move through the solution at hand. In addition, productive failure’s use of contrasting solutions is a proven concept (Schwartz & Bransford, 1998). However, productive failure’s emphasis on complete lack of structured support and students reaching a failure point adds to the broader discussion.

Productive failure essentially can be encapsulated into a three-stage process, where educators put together the (1) activity, (2) participant structures, and (3) social surroundings (Kapur & Bielaczyc, 2012). This three-part structure gives guidelines for how to build and scaffold the activity, have the participant interact with the activity, and set up the environment around the student to best promote learning through the productive failure structure (Kapur & Bielaczyc, 2012). This study describes a specific educational intervention that combines productive failure pedagogy within a model-based learning framework to promote computational thinking through engineering computational modeling activities.

5.5 Conjecture map and explanation for the current study

A conjecture map for the current study was developed to align the various pedagogical methods and theories in terms of design and theoretical conjectures. The conjecture map is presented from left to right, starting with the overall conjectures and ending with the theoretical conjectures and outcomes. Figure 7 depicts the conjecture map aligning the various theories and practices used within the current study.

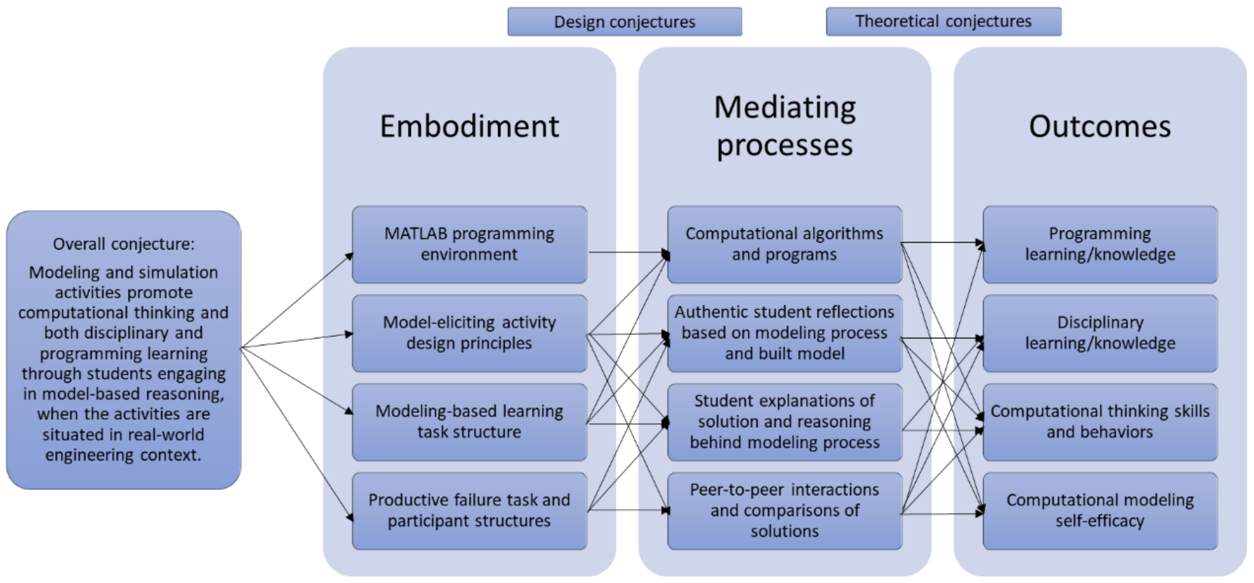
Design conjectures
Theoretical conjectures
Embodiment
Mediating processes
Outcomes
Overall conjecture: Modeling and simulation activities promote computational thinking and both disciplinary and programming learning through students engaging in model-based reasoning, when the activities are situated in real-world engineering context.
MATLAB programming environment
Model-eliciting activity design principles
Modeling-based learning task structure
Productive failure task and participant structures
Computational algorithms and programs
Authentic student reflections based on modeling process and built model
Student explanations of solution and reasoning behind modeling process
Peer-to-peer interactions and comparisons of solutions
Programming learning/knowledge
Disciplinary learning/knowledge
Computational thinking skills and behaviors
Computational modeling self-efficacy

Figure 7. Conjecture map for the overall line of research.

The overall conjecture for the study is that *modeling and simulation activities embedded in authentic, real-world context promote computational thinking and disciplinary and programming learning through students engaging in model-based reasoning.* This means that first, I must discuss how modeling and simulation activities embedded in real-world context can be developed and implemented. Thus, I must discuss the embodiment of the activity.

The embodiment of the activity is four-fold, three of which have previously been covered in this chapter: (1) modeling-based learning, (2) model-eliciting activities, and (3) productive failure. The tool being used by the students is the programming environment, MATLAB. The choice of this tool for the current study is due to its prevalence and use among the engineering and scientific communities, especially in engineering education settings (Froyd et al., 2012). However, MATLAB does little to structure the actual nature of the activity and the participant structures, being relied on mainly as a medium to produce the computational models. The embodiment is connected to the mediating processes through the design conjectures.

The first mediating process would be the creation of the actual mathematical/computational model that is delivered in the form of commented MATLAB programming files. Multiple pieces of the embodiment lead to this mediating process. The embodiment tool of the MATLAB programming environment directly connects to this mediating process. It is the direct tool used to build the actual model physically. The MEA structure leads to this computational model through the *Model Construction Principle*, which dictates that an actual model be built (Diefes-Dux et al., 2004). Additionally, the MbL framework also leads to this mediating process through the second phase of *Constructing the Model* (Louca & Zacharia, 2012). Thus, the emergence of programming algorithms and computer programs is the most straightforward mediating process to predict the emergence of within a design conjecture.

The second mediating process is authentic student reflections based on the modeling process and the built model. In terms of design conjectures, multiple pieces of the embodiment allow us to predict the emergence of these authentic reflections. First, the MEA framework delivers this mediating process through the *Reality Principle, the Self-Assessment Principle,* and *the Construct Documentation Principle* (Diefes-Dux et al., 2004). The *Reality Principle* assures that the problem is situated within an authentic, real-world engineering context. The *Self-Assessment Principle* ensures that the problem has explicit criteria to evaluate their model for reflection. Finally, the *Construct Documentation Principle* dictates that students produce documentation that records their ideas throughout the learning process, allowing for continuous reflection. Additionally, MbL allows for authentic reflections through the *Evaluate the Model* and *Revise the Model* steps as students test their model and look at ways to revise their model based on the limitations found through the evaluation process (Louca & Zacharia, 2012). Finally, the second phase of productive failure allows students to reflect on the problem-solving process by working with peers to understand why some solutions are better than others and foster engagement through group activity (Kapur & Bielaczyc, 2012).

The third mediating process is the student's explanation of their model and the reasoning behind their modeling process. Multiple embodiment elements lead to the design conjecture that this will emerge as a mediating process. The first would be the MEA framework via the *Construct Documentation Principle,* which dictates that students document their learning process through each phase of the modeling activity (Diefes-Dux et al., 2004). The MbL framework leads to this through the nature of the *Revision of the Model* step back to the *Construction of the Model* step as students reason through why the model has limitations and how to make the prediction of the model closer to the actualization of reality (Louca & Zacharia, 2012). Finally, the productive failure framework leads to this mediating process through both phases. Students explore why different models are different and why some models are better than others in given situations with the reasoning for the differences (Kapur & Bielaczyc, 2012).

Finally, the last mediating process is peer-to-peer interactions and comparing solutions. This mediating process is directly related and expected due to the productive failure framework. The first and second phases encourage group interaction and comparison amongst student solution methods (Kapur & Bielaczyc, 2012). However, the MEA framework also contributes to this mediating process through the *Self-Assessment Principle*, with prescribes the activity having measurable attributes and criteria to which the models can be assessed, in this case, specifically against peer solutions (Diefes-Dux et al., 2004).

5.6 Theoretical conjectures of the designed learning intervention

Beyond design conjectures, there are also theoretical conjectures in DBR that allow the results of this methodology to contribute to the larger body of theory. In the case of this study, there are four expected outcomes based on the embodiment and the mediating processes: disciplinary learning, programming learning, computational thinking, and increases in computational self-efficacy.

The main theoretical conjecture of this study leads to an expectation of computational thinking skills to emerge from the mediating processes. The mediating processes such as explaining the model, the reflection of the model, peer-to-peer interactions, and algorithm development will lead to model-based reasoning intertwined with computational thinking. For example, abstraction, generalization, and evaluation are three critical pieces of computational thinking (Curzon et al., 2014; Grover & Pea, 2013; Selby & Woollard, 2013). All three of these (abstraction, evaluation, and generalization) are associated with model-based reasoning (Nersessian, 2007). Additionally, algorithmic thinking and decomposition of a problem are two other commonly cited skills related to computational thinking (V. Barr & Stephenson, 2011; Weintrop et al., 2016; Wing, 2008). The mediating process of producing an algorithm should lead to algorithmic thinking and decomposing a problem as a skill. Altogether, the theoretical conjecture here is that developing computational models in an authentic engineering context will produce computational thinking skills because the main form of reasoning throughout the modeling process (model-based reasoning) overlaps significantly with computational thinking skills. We expect this outcome to be demonstrated through student explanations, reflections, algorithms, and peer interactions.

The following two outcomes of the study go hand in hand, the programming and disciplinary learning from the intervention. For programming learning, the mediating processes of building the actual computer algorithm and peer-to-peer interactions are expected to contribute to this type of learning. Programming the model will contribute to learning programming knowledge through an authentic experience situated in a realistic context. The constructivist viewpoint informing this study emphasizes the role of experience in education, highlighting that knowledge is constructed from previous experiences through which we view the world (Dewey, 1938; Ertmer & Newby, 1993). Another theoretical conjecture is that the peer-to-peer interactions in multiple phases of the designed intervention will contribute to disciplinary and programming learning by interacting with peers who can serve as knowledgeable others helping students navigate through their understandings (Vygotsky, 1978). When working with peers in planning and evaluating their models through the productive failure framework, students get exposure to other students of varying skill levels in an environment where it is safe to ask questions and understand differences between models.

Additionally, one would expect authentic reflections based on the modeling process and student explanations of the solution and reasoning behind the modeling process to lead to disciplinary learning. Generating authentic explanations and reflections about a model must have reasoning processes proceeding them (Buckley, 2012). Using our theoretical framework of model-based reasoning, when we explain and reflect on a model and modeling process (which are external visual representations of information), we are explaining and reflecting an internal model-based reasoning process based on internal mental models of the phenomena (Craik, 1943; Johnson-Laird, 1983). Suppose these external models conflict with the internal mental model. In that case, the understandings that form the basis of the mental model must be adjusted to compensate, eventually solidifying to schema or other long-term understandings (Ifenthaler & Seel, 2013). Both disciplinary and computational learning will not be explicitly investigated in this study but will be addressed in future research.

The final outcome we expect to see would be neutral or positive impacts on student computational self-efficacy. The theoretical conjectures in the conjecture map have three mediating processes that lead to this outcome: student reflections, peer-to-peer interactions, and building the actual computational model. Hutchison et al. (2006) found that students' perceived abilities with mastery experiences in computing and computer applications, such as building and programming complex computational models, correlated with positive self-efficacy beliefs. They found that vicarious experiences, such as working closely with team members and having supportive team members, positively impacted self-efficacy beliefs (Hutchison et al., 2006). Magana et al. (2016) found similar results in that computational education presented within authentic engineering contexts can significantly increase student self-beliefs. Additionally, self-reflection allows for students to be able to "expand their self-knowledge of what they can and cannot do" (Bandura, 1998, p. 11). Thus, we would expect all of these factors to positively impact student self-efficacy as students gain mastery and vicarious experiences and reflect on these experiences. Self-efficacy gains are external to this study, and findings from this intervention regarding self-efficacy can be found in other studies from the researcher (Lyon, Jaiswal, et al., 2020).

5.7 Overview of the designed learning environment

The different aspects of the embodiment and theories lead to a four-phase design of the learning environment, including (1) planning the model, (2) building the model, (3) evaluating the model, and (4) reflecting on the model.

In the *Planning the Model* phase, students combine their own experiences and knowledge to generate and explore different ways the problem could be solved. Students construct at least one of the various identified solution pathways in the Building the Model phase. In the *Evaluating the Model* phase, students meet with multiple other students from the class to compare the different modeling decisions made and the impact of those decisions. Finally, in the *Reflecting on the Model* phase, students reflect on how they might revise the model in future iterations and the modeling process overall. The physical manifestation of the modeling problem follows MEA principles.

During each phase of the activity, students were asked to fill out templates to organize and somewhat standardize the information collected from each student. Additionally, a programming file template was given to the students to demonstrate what proper commenting within a programming file should include. These templates are included in Appendices 1-5.

5.8 Summary

To conclude, the learning design directly impacts the research goals, most specifically the research's ability to deliver pragmatic results and meet the needs of students and instructors in a specific context. Three frameworks are used to construct the learning design: (1) the model-eliciting activity to frame the structure of the assignment, (2) productive failure to design the learning environment and pedagogy used, and (3) models and modeling to frame the learning process throughout the activity. A conjecture map is then used to map these frameworks to both the design and theoretical conjectures that the study is making, focusing on the elicitation of computational thinking. However, to study the elicitation of computational thinking, the study must have a proper research design to draw out the rich data needed to understand emergent outcomes.

6. RESEARCH DESIGN

6.1 Introduction

This study investigates the use of modeling and simulation learning environments and their ability to elicit computational thinking within the undergraduate classroom. This study uses a descriptive approach through the larger paradigm of qualitative inquiry to understand the various computational thinking outcomes that result. In doing so, a pedagogical approach is designed and implemented in the form of a single case study within an upper-division engineering class. Design-based research is the specific approach taken to data collection through classroom artifacts and audio/video of classroom discussions. Deductive thematic analysis is used to analyze the various computational thinking practices used during each phase of the modeling and simulation cycle. These methods are used to answer the one research question primarily:

(1) *How do modeling and simulation activities elicit computational thinking practices in the context of undergraduate engineering education?*

6.2 The research paradigm of qualitative inquiry

Qualitative inquiry is based on interpretivism and constructivism, emphasizing process and meaning, often with smaller sample sizes (Sale & Brazil, 2002). This is in opposition to quantitative research, which is derived from positivism and understanding truth, often focused on finding an external truth to human reality. This difference in approach leaves qualitative methods the unique ability to dive deep into a specific context, although it costs them the broader generalizability that quantitative methods often enjoy.

However, there are many distinct advantages to a qualitative inquiry over quantitative research methods. Guba and Lincoln (Guba & Lincoln, 1994) identified that some of the significant benefits of qualitative inquiry are its ability to account for context, its ability to include the meaning and purpose of the data, the understanding of broader theory with a local context, and the applicability of qualitative research to individual cases. Whereas this research expects that computational thinking is often not explicit, qualitative research is expected to overcome this by looking into the underlying text of student artifacts and discussion to uncover the meaning, purposes, and context within which the computational thinking practices are sitting.

6.3 Case study

The data collection method for the current study uses a case study framework. A case study is a method that can be used within either qualitative or quantitative paradigms. A case study can be used in a wide variety of contexts such as public policy, political research, management studies, and more academic uses within the social sciences (R. K. Yin, 1994). Whereas other qualitative methods have a strict type of data to be analyzed, a case study as a method does not require any specific type of data to be collected (R. K. Yin, 1981). Additionally, case studies can be used to answer multiple different kinds of questions. The type of research question will determine what type of case study is being conducted: exploratory, descriptive, or explanatory (R. K. Yin, 1994). Exploratory case studies are often performed as an initial study as a basis for future research. In contrast, explanatory case studies look for causal investigations, and descriptive cases build on previous descriptive theory (Tellis, 1997). Other types of case studies would include multiple-case, intrinsic, instrumental, and collective; and explanations and examples of these can be found in the literature (Baxter & Jack, 2008)

There are multiple defining features of a case study that can aid researchers in knowing if a case study is an appropriate route for a given research inquiry. Yin (2009) listed two defining features of a case study: (1) it investigates a phenomenon within a real-world context, and (2) the boundaries of the phenomenon are not clear. For case studies, one must also select the unit of analysis, which is often a particular system under a given time frame for which the research is interested in studying (Tellis, 1997). Thus, a case study is primarily a bounded phenomenon under which a specific system is under investigation within its real-world context.

During the data collection phase of a case study, triangulation is often used amongst varied data sources (Tellis, 1997; R. K. Yin, 1994). By using multiple data sources, triangulation within a case study allows for increased reliability by allowing each of the data courses to corroborate the evidence of the others (Tellis, 1997). A further variant of this idea of triangulation is the idea of crystallization, in which multiple approaches to data are used amongst various lived truths and allow them to converge to a cohesive story of the phenomenon under investigation (Ellingson, 2009). While this research does not use multiple perspectives to investigate the data, multiple different data sources converge a specific localized truth regarding computational thinking in our designed intervention.

6.4 Advantages and limitations of a case study

Yin (2009) described five different applications of case study: (1) to explain the causal links in an intervention, (2) to describe an intervention and its real-world context, (3) to illustrate specific topics, (4) explore interventions with no clear outcomes, and (5) a study of an evaluative study. Thus, the case study represents a valuable and advantageous ally in pursuing research in any of these situations. Another advantage of a case study is that the researcher and the participant can often work together (Baxter & Jack, 2008). This is in line with the idea that case studies are embedded within a real-world context. Because of this intimate relationship between the research and the context in which the research occurs, it allows for a case study to collect extremely rich data.

One common limitation of the case study is the lack of generalization provided by a single case study or by limited groups of case studies (Tellis, 1997; R. K. Yin, 1994). In the case of this study, this is something that the author fully acknowledges, understanding that a trade-off is made between generalizability and the ability to impact the classroom environment through the design of the research. Another limitation often cited is the limited rigor shown by many who are doing case study research (R. K. Yin, 1994). However, this is more of an indictment of the researchers conducting the research than the method itself.

6.5 The present study

Given the definitions provided above, it becomes imperative to define certain qualities and phenomena of the current study to place it within the case study methodology properly. Namely, define what the phenomenon under study is, the boundaries of the present study, the unit of analysis regarding the data, and the data sources used to triangulate the study results.

The phenomenon under study is computational thinking and how it is characterized within a modeling intervention in a capstone engineering course. A capstone engineering course was chosen for the case study for multiple reasons. First, the capstone engineering course is situated within bioengineering, which traditionally has more gender representation than other engineering disciplines. Additionally, much of the literature around modeling and simulation focuses on first-year students or students early in their degree program, making this context one that the research can address gaps in the literature. Finally, a capstone course was chosen because students would already have classes in both the disciplinary content and programming content needed to understand how computational thinking is used within an ill-structured modeling and simulation environment.

This descriptive case study aims to describe the elicitation of computational thinking and the real-life context in which it is occurring, which is a computational modeling intervention within a capstone engineering course. The boundaries of the case study are only to investigate the effects of the computational modeling exercises. While enrolled in the course, students produced a variety of artifacts, including homework, tests, and other project deliverables. However, this study only looks at the artifacts produced and the conversations that are had while completing the modeling activities.

The unit of analysis for the study is a single DBR iteration of a computational modeling activity within a capstone engineering course. While multiple activities occur over the semester, the case study only looks at one specifically. However, within this unit of analysis, there are many different sources of data available to analyze the elicitation of computational thinking. These include reports produced by the student, notes taken by the student, audio and video recordings of conversations while they complete the modeling activity, and programming files with in-code comments produced by the student. These multiple forms of data allow for triangulation of the results among the data streams and corroboration of results.

6.6 Participants and the current study

The data for this study is obtained from a class covering food and pharmaceutical processing. Students worked within groups of three to four to plan out how they were going to computationally model a food canning line that was sterilizing different food products, looking at both the microbial concerns of the product and the nutritional degradation concerns of the food product. The students were upper-level undergraduate students in an agricultural and biological engineering program, most of whom were in their final year of undergraduate studies. This means that most of them had taken the majority of their disciplinary classwork to this point, including heat and mass transfer, fluid mechanics, reaction kinetics, mathematical modeling/numerical methods, as well as at least one programming course.

Race and gender data were not collected as part of the learning intervention. The engineering department from which the data was collected had slightly more students that identified as females than males at the time of data collection. Given that the department was relatively small and there is only one main section of this required class per semester, the students in the class were roughly representative of the department. All names given in the results are pseudo names to protect student identity. The research aimed to illuminate different ways in which engineering students used computational thinking within a modeling and simulation activity. The total size of the population who consented to the research study was 45 students, which was most of the students in the class.

The class in which this was implemented followed a two-part structure, with both lab time and lecture time. The lab portion of the course met two times a week for two hours. The lecture portion of the class met three times a week for one hour. The class consisted of regular homework assignments, a senior design project, and regular quizzes in addition to the mathematical modeling activities outlined as the basis of this dissertation. Lecture periods were generally taught in a traditional format, which included the instructor using the blackboard and projector slides to convey information to the students while the students took notes. The mathematical modeling activities outlined in this study occurred during the lab portions of the class. The overall learning objectives of the course as taken from the syllabus include: (1) An understanding of the principles and design/scale-up aspects of various unit operations and processes utilized by the biological and food process industries, (2) Develop self-learning techniques to acquire new knowledge for lifelong learning, (3) A capacity to apply scale-up principles for the development of typical industrial processes, and (4) Develop unit operation designs that account for the effect of process variables when producing high quality, cost-effective and safe product using the minimum of resources.

The intervention outlined in this study required students to use multiple concepts from across the curriculum such as heat and mass transfer, reaction kinetics, numerical methods, and computer programming. For example, they needed heat transfer to model the sterilization process, kinetics to understand how the bacteria were destroyed, and numerical methods to model the process in the MATLAB programming environment. Yet, the problem they were working on was one that they had not completed a similar problem previously. Hence, the students had to pull together multiple strands of knowledge and transfer it into a new context. Because of the productive failure framework in the design, students were not given an elaborate introduction to the problem and solution process, but rather were expected to pull together many of these topics on their own and try to chart a solution path on their own. In turn, this prepared them for their capstone projects in the spring where they would be expected to do much the same thing with little introduction or initial guidance from the instructional team.

6.7 Data collection method

Different data sources were collected within each step of the modeling intervention to get a better holistic view of how computational thinking was used when students were completing the activities. This falls in line with the case study approach of using multiple sources of data and the design-based research approach, which allows for the collection of many different forms of data (A. E. Kelly, 2004; R. K. Yin, 1994). The following table, Table 1, overviews the data sources collected at each stage in the intervention.

Table 1. Overview of data sources at each stage of the research design.

Week	**Phase**	**Data Sources**
1	Planning the model	Template: Planning the model Audio/video: Groups meeting and planning their modeling strategies.
1-3	Building the model	Template: Building the model Template: MATLAB programming files
3	Evaluating the model	Template: Evaluating the model Audio/video: Groups meeting to evaluate and groups meeting with other groups.
4	Reflecting on the model	Template: Reflecting on the model

These data sources were chosen to collect rich and expansive data across the dataset. When available, audio and video recordings were taken of classroom interactions to collect information on how students used computational thinking while solving the actual problem. Audio and video were not available during the building or the reflecting phases of the modeling activity. This was supplemented during the building the model phase by having the students write highly detailed programming comments to try and capture the students thinking process during the building phase. Thus, two data sources were used for each of the first three phases of the modeling activity.

Each phase had a report or note template for the students to write and answer questions regarding their model. Additionally, each phase used either detailed programming comments (similar to a journaling technique) or audio/video data to collect more detailed process data. The audio and video files were then transcribed to investigate the classroom interactions. The templates and programming comments were already in textual form and ready for analysis. Thus, all of the data obtained was in a rich text format that was ready to be analyzed for themes regarding the computational thinking outcomes demonstrated as a result of the intervention. The reflecting on the model phase was limited in that students only have a report. Because of the similarities between the reflection and evaluating process, these two phases were analyzed as a whole.

These sources were chosen to answer the research question because, when available, the artifacts captured the final state and thinking of the student (through the artifacts) and the process students take (through the audio/video and, to an extent, the programming comments). By using data that is in rich text format, the goal was to capture a broad range of computational thinking outcomes by understanding how students explain themselves, their solutions, and reason through their problem-solving process.

6.8 Data analysis method

Thematic analysis was used as the analytical procedure. This was chosen for the current data set because thematic analysis can capture both manifest (explicit) and latent (underlying) aspects. Thematic analysis is also not based on the frequency of categories but the importance of categories. Because of the complexities of the artifacts (repetitive nature, open-endedness), the frequency seemed of minor importance, ruling out other qualitative analysis measures such as content analysis and making thematic analysis the natural choice for the data set. Finally, thematic analysis fit the data set in that there were already a set of deductive categories for which the researcher coded (computational thinking practices).

Thematic analysis has its roots in a constructivist view of the world, one in which there is no absolute truth but locally and personally constructed truths built through experiences (Neuendorf, 2019). This means that thematic analysis is often tied to the underlying meaning of texts, combining the analysis of both the latent and manifest content of the data (Vaismoradi et al., 2013). There are multiple definitions of thematic analysis within the literature. Braun and Clarke (2006) defined thematic analysis as a "method for identifying, analyzing, and reporting patterns (themes) within data" (p. 6). Joffe and Yardley (2004) added that thematic analysis improves understanding of the meaning of the text within its broader context compared to other text analysis methods such as content analysis. This makes thematic analysis more qualitative than traditional textual analysis methods.

Thematic analysis often gauges the importance of a theme based on how important the theme is to answer the research questions (Vaismoradi et al., 2013). This is because themes are derived not only from the context in which they are found but also from the meaning they stand for (H Joffe & Yardley, 2004). Joffe and Yardley (2004) used an example of a code of "stigma" in a coding process. In thematic analysis, if a person refers to the idea of stigma or avoiding something but doesn't use the word explicitly, it can be coded into a theme, thus bringing underlying meaning into the analysis. Thematic analysis, therefore, often looks at both the manifest and the latent aspects of the data as opposed to looking at one or the other as the two are often intertwined (Vaismoradi et al., 2013).

Thematic analysis has been used as a term used for generalized qualitative coding and has been a method often neglected of a clear definition in the literature. However, the seminal work of Braun and Clarke (2006) outlined the method of thematic analysis. They described the method as consisting of a six-step process of (1) familiarizing oneself with the data, (2) generating initial codes and coding structures, (3) searching for themes within the data set, (4) reviewing the themes, (5) giving definition to and naming the themes, and (6) producing a report around the found themes.

Familiarizing oneself with the data step is very similar to the preparation phase of other textual analysis methods such as content analysis (Vaismoradi et al., 2013). Data can come from various backgrounds, whether collected by yourself or somebody else. So, the first phase is to get familiar with the data. This process involves reading through the data multiple times while searching for initial ideas about patterns and meanings (Braun & Clarke, 2006). Often, if data is verbal, this familiarization process begins and is conducted in part during the transcription of the data into a written format (Braun & Clarke, 2006).

The second phase of the process is generating initial codes. This process has already begun during the first phase as early patterns and meanings were looked into during the familiarization process (Braun & Clarke, 2006). The codes are not the bigger themes but will be combined to make themes. In an inductive approach, the themes are data-driven and thus will emerge from the data itself; in a deductive approach, the themes are theory-driven, and results will focus on a particular theory investigated within the data (Braun & Clarke, 2006).

In the third and fourth phases, the coded items are searched for how they could be grouped into themes and then reviewed (Braun & Clarke, 2006). At the initiation of the third phase, the first set of themes will certainly not be the same set of themes at the end of the fourth phase as themes split into multiple themes and multiple themes are condensed into singular themes. At the end of the fourth phase, determined themes are arranged into a sort of relationship, understanding how each of the themes relate to each other. This can be done with a thematic map that organizes all themes (Braun & Clarke, 2006).

In the fifth phase, themes are given a definition and a name (Braun & Clarke, 2006). As thematic analysis aims to get at the underlying meaning, Braun and Clarke (2006) cautioned against just paraphrasing the contents of each theme, but instead talk about "what is interesting about them and why!" (p. 22). Additionally, while it is important to discuss what themes are, it is also important to discuss what they are not so that the reader can understand clear definitional lines between themes.

The sixth phase is the reporting phase, where results are put together into a cohesive narrative giving the themes and putting them into a coherent and convincing story (Braun & Clarke, 2006). Quotes should be given regarding the different identified themes (Braun & Clarke, 2006). Whether or not interrater reliability should be used in the thematic analysis method is a contended point in the literature. Some researchers pointed out that interrater reliability often devolves to one researcher convincing another as to how to look at and interpret the data (H Joffe & Yardley, 2004). However, there is still a movement within the qualitative research community to keep this measure as a source of validity within this methodology (Neuendorf, 2019).

6.9 Deductive vs. inductive thematic analysis

Thematic analysis can be deductive or inductive (Neuendorf, 2019; Vaismoradi et al., 2013). Deductive analysis is when theory, literature, or prior work is used to predefine categories or themes to solidify or test the previous theory (Elo & Kyngäs, 2008; H Joffe & Yardley, 2004; Vaismoradi et al., 2013). This is useful when the research aims to prove how predefined categories are used within a new context or to validate the findings of previous studies. Thematic analysis can also be inductive, where themes and types emerge from the data itself (Braun & Clarke, 2006;

Elo & Kyngäs, 2008; H Joffe & Yardley, 2004; Vaismoradi et al., 2013). Inductive analysis is useful when no prior framework informs the analysis, allowing the themes to be intimately tied to the data itself. Inductive analysis is often a far richer analysis because of how the categories and themes are directly emergent from the data instead of trying to fit previous theories and categories to existing text (Vaismoradi et al., 2013). In an inductive thematic analysis, the researcher is looking to arrive at a point of saturation of the themes (Case & Light, 2011).

When the thematic analysis method is used deductively, one may start with a set of codes. Still, the technique encourages the codebook to stay flexible if the data shows additional codes, as saturation is important in the thematic analysis method (Neuendorf, 2019). Saturation is when further data analysis does not seem to be substantially adding any new themes or changing the findings (Case & Light, 2011). As to where other textual analysis methods are more concerned with frequencies of themes and characteristics of the text, thematic analysis is more fundamentally concerned with a saturation of the different themes. This means that the thematic analysis results are the themes themselves (Neuendorf, 2019).

6.10 Advantages and limitations of thematic analysis

There are many advantages to using thematic analysis over other qualitative data analysis methods. The biggest is that thematic analysis allows the researcher to look at the underlying meaning of the data by focusing on the context in which text is said or used (Vaismoradi et al., 2013). Unlike other forms of qualitative data analysis, such as content analysis, thematic analysis allows the researcher to look at both the manifest and latent content with the sample. Whereas in other methods, the frequency might be used to determine the importance of a theme, thematic analysis is much more qualitative by judging the importance of a theme based on how it relates to and addresses the underlying research questions (Vaismoradi et al., 2013).

While thematic analysis certainly has advantages, there are other clear disadvantages to the method. The biggest drawback to thematic analysis is the often clear lack of methodology (Braun & Clarke, 2006). However, Braun and Clarke (2006) outlined a multi-step method that was followed for this study and, by doing so, addresses the limitations provided by thematic analysis. Additionally, as with all qualitative analyses, validity and reliability issues can plague study results. In this study, metrics such as inter-rater reliability and transparency of the research are used to address this limitation and will be described in more detail later.

6.11 Reliability and trustworthiness

The study uses three approaches to ensure the reliability and trustworthiness of the results. The first is inter-rater reliability, which often appears in percent agreement. Percent agreement does have drawbacks, the biggest of which is its inability to integrate the probability that two observers give the same rating due to chance (Hunt, 1986). However, this metric is often used in these studies for its ability to quickly assess the degree of overlap between two raters and its ability to incorporate non-mutually exclusive categories easily.

Additionally, the study uses transparency or rich and thick descriptions of the data to provide credibility to the findings (Onwuegbuzie & Leech, 2007). These thick descriptions offer both data and production transparency. Readers can better understand what the raw data looked like and how the researchers arrived at their conclusions based on that data (Moravcsik, 2014). Throughout the analysis, the reader will see example quotes given often, in an attempt by the researcher to maximize the amount of transparency between the researcher and the reader. By doing so, the reader will have a richer understanding of the data and better understand the transferability of the findings and how the results relate to other contexts and environments (Onwuegbuzie & Leech, 2007).

Finally, the study uses triangulation to add trustworthiness to the results by ensuring that the results were found through multiple different data sources. Triangulation is a typical application of trustworthiness within a case study methodology (Tellis, 1997). Triangulation generally helps protect the results from researcher bias and works towards the validity of the results.

6.12 Ethical conduct of the research

The Institutional Review Board approved all procedures under IRB protocol number 1806020715. All research participants signed consent forms and were free to withdraw from the research at any point in the study. Additionally, all efforts have been made to protect the identity of all participants in the study.

6.13 Researcher bias and perceptivity

The researcher acknowledges that all research contains some bias of the individual conducting the research. The literature recognizes that this can be especially problematic within qualitative spaces of inquiry (Chenail, 2011). Researcher bias can be either active or passive and involve anything from the clothes that a researcher wears, all the way to the way they talk to participants that may influence participants given prior researcher beliefs (Onwuegbuzie & Leech, 2007). Additionally, bias within the analysis is possible as I acknowledge that many researchers, including myself, are biased towards finding positive results from an intervention or a course of inquiry.

To combat this bias, multiple approaches are taken to ensure rigor and validity within the study results. First, interrater reliability is used in some places to ensure that another researcher aside from the primary researcher can see similar results and interpret the data in similar ways. Additionally, transparency is attempted by providing portions of the raw data to the reader to allow the reader to understand how the researcher is obtaining results from the data itself. Finally, triangulation of multiple data sources is used to limit researcher bias by situating the results within multiple different data sources.

6.14 Structure of the three studies

The three studies in this dissertation look at each of the steps of the modeling and simulation cycle separately. In the first study, an initial codebook of computational thinking outcomes is developed from the building the model step, where students design and program their solutions to the modeling challenge given to them. In study two, this initial codebook is expanded using the data collected from the planning the model phase, where students worked as teams to put together their initial plans for how to solve the modeling challenge. Finally, in the third study the codebook is again expanded using data from the evaluation and reflection phases of the modeling and simulation cycle.

Thus, each study uses the same thematic analysis method of identifying outcomes in the data, but each study uses different data sources to do so. For example, the first study uses the building the model template along with the programming files to develop the initial codebook. The second study uses the planning the model templates along with the recordings of the classroom planning sessions. And finally the third study uses the evaluating and reflecting templates along with the class discussion where students compared their developed models. The data from all three studies is from the same intervention, but separate and unique data sources are used for each study.

6.15 Summary

To conclude, this study uses a case study methodology to collect classroom data from an upper-division senior design course on food and pharmaceutical processing. While there are limitations to a case study methodology, its advantages are that it allows researchers to collect data in a naturalistic environment. Thematic analysis is used to analyze artifacts and observations from the case study to identify computational thinking behaviors and skills that emerged during the modeling intervention. Reliability is ensured using inter-rater reliability, transparency, and triangulation in parts of the analysis. And finally, steps have been taken to limit research bias through the analysis and ensure the research's ethical conduct.

7. THE USE OF ENGINEERING MODEL-BUILDING ACTIVITIES TO ELICIT COMPUTATIONAL THINKING: A DESIGN-BASED RESEARCH STUDY

Lyon, JA, Magana, AJ. The use of engineering model-building activities to elicit computational thinking: A design-based research study. *J Eng Educ*. 2021; 110: 184-206.
https://doi.org/10.1002/jee.20372

7.1 Abstract

Background Computation and computational thinking are of great interest to both engineering research and teaching communities. Effective learning environments are needed to incorporate computational thinking within the engineering disciplines. Design-based research is uniquely positioned to address this need for designing effective learning environments.

Purpose This design-based research study characterizes the different ways in which students used computational thinking when building computational models. The design of the model-building activities was grounded in productive failure and model-eliciting activities.

Design The design-based research study implemented the computational modeling activities within an engineering capstone course. The research question was: *What types of computational thinking outcomes emerge when engineering students build computational models?* Thematic analysis was used on individual student artifacts to identify key computational thinking outcomes that were elicited as a result of the intervention.

Results Throughout the building of the model students demonstrated the use of computational thinking outcomes, mainly: abstraction, algorithmic thinking, evaluation, generalization, and decomposition. However, the diversity and density of use for each outcome were different and unique.

Conclusions This study shows how building computational models, when guided by current educational theories, can allow for students to practice the use of computational thinking, and for educators to incorporate these key practices into their engineering classrooms.

Keywords: computational thinking, modeling and simulation, design-based research, STEM

7.2 Introduction

Computational thinking (CT) has been of particular interest to both the engineering research and educational communities since the term emerged from the field of computer science (Wing, 2006). Consequently, there have been many different proposed definitions of CT as well as discourse as to what behaviors CT is comprised (Kalelioğlu et al., 2016). This has led to much recent work that focuses primarily on the definitional aspects of CT (Grover & Pea, 2013). Much of the CT literature also describes the context in which CT is useful, such as technology, programming, 3D modeling, and education (Ilic et al., 2018). In part, the rise of interest in CT and more broadly, computational sciences, can be attributed to national calls to increase computation and computer science at all levels of education (National Research Council (NRC), 2011; President's Information Technology Advisory Committee (PITAC), 2005). The calls are, in part, based on concerns of national organizations as it relates to national security and economic competitiveness (President’s Information Technology Advisory Committee, 2005). In response, programs such as the *Computer Science for All* initiative have emerged to support computer science and CT curriculum throughout the primary and secondary school years (United States Office of the Press Secretary, 2016).

The field of engineering is not immune to these national calls and must respond by increasing computation education for engineering students. One pathway for achieving this integration is through the practices of modeling and simulation, which are commonly used in engineering professional practice (Gainsburg, 2006; Magana & Silva Coutinho, 2017). Modeling and simulation have been heavily studied in the context of engineering education (Diefes-Dux et al., 2004, e.g., 2006; Lyon, Fennell, et al., 2020; Lyon & Magana, 2020a; Magana, 2017; Magana et al., 2019, 2020). The majority of these studies have primarily focused on the design and evaluation of learning interventions that engage students in mathematical modeling (e.g., Diefes-Dux et al., 2006; Lyon & Magana, 2020a) or use of computational simulations to improve conceptual learning (e.g., Alabi et al., 2015; Brophy et al., 2013; Mansbach et al., 2016). Other studies have investigated modeling and simulation practices in engineering education at the intersection of computation or computational science (e.g., Magana et al., 2013, 2016, 2019; Vieira et al., 2016). These studies have identified the challenges students encounter as they program their computational models or modify existing computational models (e.g., Magana et al., 2017; Vieira et al., 2018), and pedagogical strategies that can support students’ in engaging in the programming of computational models more effectively (e.g., Vieira et al., 2017, 2019). Thus, modeling and designing model-based learning environments may be a good entry point to understanding CT within the context of engineering education.

The goal of this study is to operationalize CT within an engineering classroom by using computational modeling and simulation practices as a mechanism to elicit CT outcomes. In order to effectively incorporate computational modeling into the classroom, design-based research (DBR) was used as the research method to understand how a designed modeling-based learning environment could be incorporated into a naturalistic setting. As such, the research question is, *What types of CT outcomes emerge when engineering students build computational models?* By approaching this question our goal is to build knowledge that can allow educators to better incorporate CT into the undergraduate classroom in practical ways, as well as to identify key ways in which students use CT within these practical applications. In addition, by applying-DBR as the methodological approach, this study demonstrates how it is possible to increase the use of evidence-based teaching in STEM education. The study presents the first iteration of a DBR classroom intervention. An early version of this first DBR intervention has been published previously (Lyon et al., 2019). The goal is to continue to iterate upon the intervention design, and in the process improve the learning materials and classroom orchestration, and to gather insights on how CT outcomes emerge in the context of modeling and simulation. In doing so this study addresses the persisting gap and need for discipline-based education research that connects theory and practice, especially in the fields of science and engineering (NRC, 2012; Borrego & Henderson, 2014). Practice-oriented research methods such as DBR, participatory action research, and case studies are needed to better understand how educational theory can best be implemented into the classroom environment (Design-based Research Collective, 2003; Miskovic & Hoop, 2006; Yin, 1994). By applying DBR methods our goal is to not only meet local educational needs but to also generate and add to existing theory (Wang & Hannafin, 2005).

7.3 Computational Thinking (CT)

In recent years, CT has been researched to not only understand what it is but to also describe ways in which it can be used throughout the entire educational process (Ilic et al., 2018). CT can be used as both a guiding framework by which to structure an entire course (Evia et al., 2015; Jeon & Kim, 2017; Lyon & Magana, 2020b), as well as a construct to be measured within student performance (Flanigan et al., 2017; Korkmaz et al., 2017). As such, the construct is able to lend itself as both a framework in which to guide the creation of educational interventions, as well as a construct to be learned from an educational intervention.

Because the nature of CT is complex, there are multiple emergent and competing definitions to operationally define CT. For this study we define CT in terms of practices. Practices refer to actions and representations of what practitioners do as they engage in their work (Lee et al., 2013). Thus, practices are a necessary part of what students must do to learn a subject and understand the nature of the field (Reynante et al., 2020). A few of the commonly cited works operationalizing CT in the context of research, policy, and industry are listed in Table 2.

Table 2. Common CT practices from the literature and from related industry sources.

Barr & Stephenson (2011)	**Grover & Pea (2013)**	**Selby & Woollard (2013)**	**College Board (2013)**	**Google (2015)**
• Data collection, analysis, and representation • Problem decomposition • Abstraction • Algorithm procedures • Automation • Parallelization • Simulation	• Abstraction and pattern generalization • Systematic processing of information • Symbol systems and representations • Structured problem decomposition. • Iterative, recursive, and parallel thinking • Conditional logic • Efficiency and performance constraints • Debugging and systematic error detection	• Algorithmic thinking • Evaluation • Decomposition • Abstraction • Generalization • A thought process	• Connecting computing • Developing computational artifacts • Abstracting • Analyzing problems and artifacts • Communicating • Collaborating	• Abstraction • Algorithm design • Automation • Data collection • Data analysis • Data representation • Decomposition • Parallelization • Pattern generalization • Pattern recognition • Simulation

As it can be observed, there are multiple practices that are repeated throughout these CT categorizations. One that is reiterated throughout every source is abstraction, suggesting it as a key component to CT. This is supported by the literature as the most common element defining CT (Kalelioğlu et al., 2016). When initially popularizing the term, Jeanette Wing (2008) wrote that "the essence of computational thinking is abstraction" (p. 3717). Grover and Pea (2013) also called abstraction the keystone of CT. Thus, abstraction is a core component to the operational definition of CT. Other practices such as decomposition, generalization, and algorithmic thinking were common to the definitions. Because of this pattern, the CT practices proposed through the body of work by Curzon et al., (2014) and Selby and Woollard (2013) were deemed appropriate as those overlapped among many of the sources. The practice definitions proposed by Selby and Woollard (2013) and adapted by Curzon et al. (2014) are:

1. Abstraction: making problems easier to solve/think about by selectively removing/hiding unnecessary complexity.
2. Algorithmic thinking: setting up problems so that they can be solved or solving problems in a stepwise manner.
3. Evaluation: comparison of one's own solution against other ideals and metrics to ensure the solution fulfills its purpose.
4. Generalization: the use of previous solutions/ideas for current or future problems.
5. Decomposition: thinking about complex problems in terms of a series of smaller problems.

Additionally, these practice definitions provide practical operationalizations that could be used as a starting point for the goals of this research (Curzon et al., 2014). Other definitions of CT have been converted into taxonomies by listing specific practice and outcomes in order to better define the CT-enabled professional (Magana, 2017; Malyn-Smith & Lee, 2012; Weintrop et al., 2016). Yet, even in light of these practical taxonomies and desired outcomes associated with CT, there continues to be limited research that has practically implemented these practices and outcomes into the classroom, with much of the research continuing to be definitional in nature (Grover & Pea, 2013; Kalelioğlu et al., 2016). As such, research studies both promoting theory and practice are needed within the CT literature to add to an undertheorized and under operationalized construct.

7.4 Methodological Approach

Researchers in engineering education have articulated the need for theoretical and empirical work on engineering learning that can be supported by the learning sciences (e.g., Johri & Olds, 2011). The present study aims to (a) elicit CT outcomes by situating learning experiences in an engineering context by including the physical and social aspects of the discipline and (b) understand how students built their computational models and how within this process students enacted CT outcomes. This study used design-based research (DBR) as a methodological approach to design, implement, and evaluate a modeling learning intervention that promoted CT. DBR has been defined as “an interdisciplinary mixed-method research approach conducted in the field that serves applied and theory building processes” (Markauskaite et al., 2011, p. 8). DBR is an approach that builds upon a body of methods that go by other names in the literature such as design experiments, design research, development research, developmental research, and formative research (Wang & Hannafin, 2005). DBR brings educational research into the naturalistic classroom to produce results that are embedded and inseparable from their educational contexts (Barab & Squire, 2004). A design study is an iterative investigation of educational interactions and outcomes elicited by a set of designed, learning designs (Confrey, 2006). The learning designs often take the form of a whole learning environment including activities, materials, tools, and notational systems, including means for sequencing and scaffolding (Reimann, 2011, p. 37). Thus, DBR provides us with a series of approaches that will allow us to “engineer” and study particular forms of learning that will be subject to test, revision, and iteration among the products of this research (Cobb et al., 2003). DBR has five main characteristics that define it; it is pragmatic, grounded, iterative and flexible, integrative, and contextual (Wang & Hannafin, 2005). Pragmatic means that DBR contributes to both educational theory and practice. The output of a DBR study should, in part, lead to design principles that can be used in the educational classroom (Barab & Squire, 2004). Grounded means that DBR should pull from current educational theory or evidence-based pedagogies in order to generate designed interventions. Interventions should be iterative, meaning that the design is continuously refined (Wang & Hannafin, 2005). The integrative nature of DBR means that it can use multiple methods to ultimately answer the research question, often using a mix of methods in order to do so (Wang & Hannafin, 2005). Finally, DBR is contextual, meaning that it is imbedded within a classroom context.

The aforementioned characteristics suggest that DBR considers education as an applied field where researchers have transformative agendas (Barab & Squire, 2004). As such, they develop contexts, frameworks, tools, and pedagogical models with the intent to produce new theories, artifacts, and practices that can impact teaching, learning, and engagement in naturalistic settings (Barab & Squire, 2004). Although in part, the contextual nature of DBR limits the generalizability of this approach, it also creates localized theories that are useful in a pragmatic way. As stated by the Design-based Research Collective (DBRC) this contextual nature is “precisely what educational research most needs to account for in order to have application to education practice" (DBRC, 2003, p. 6). Thus, for what DBR lacks in generalizability it gains in its ability to take broad research theory and put it into practice. This study presents the first iteration of a DBR cycle.

7.5 Theoretical Framework and Implications for the Study

A DBR study starts by first identifying a learning theory that will guide the design of the educational intervention. The cycle then continues with the development and deployment of the educational intervention, and concludes with a thorough evaluation of the learning outcomes. To do so, education research is then performed to evaluate the impact of the intervention. As this cycle is enacted iteratively in classroom settings, the ultimate goal is to contribute to theory by identifying design principles for creating discipline-based learning experiences. The guiding theory for the design of our intervention is model-based reasoning (Ifenthaler & Seel, 2013).

Model-based reasoning has its roots on research from the cognitive sciences that explains how new concepts are formed and how they are related to existing concepts (Nersessian, 1999). Associated with a logical positivist view, model-based reasoning treats conceptual structures as languages for "explaining the nature of the logical and interpretive relations between the old and new conceptual structures and between concepts and the world" (Nersessian, 1999, p. 5). That is, model-based reasoning is one form of scientific thinking concerned with describing how novel scientific representations are created from existing representations (Nersessian, 2002). As individuals engage in modeling processes, they use or create models by connecting phenomena in the natural world to language, from existing knowledge to new knowledge, and from conception to experiment (Nersessian, 1999).

Model-based reasoning supposes that every individual has a unique set of mental models in which they understand and interact with the world (Ifenthaler & Seel, 2013). Sometimes, these mental models are externalized as mathematical or computational models as a way of offloading or externalizing some of the required processing (Jonassen, 2009; Quillin & Thomas, 2015). Because the computational model is an extension of an internal mental model that guides a student's understanding of the world, when investigating a student's modeling process, the reasoning behind it should reflect the student's interpretation of the physical phenomena. Assumptions, design decisions, and reflections upon the results should all indicate how the student understands the underlying phenomenon being modeled. It is within this reasoning process, that our study aims to identify the different ways that students represented their physical model, and to describe how the commonly identified components of CT were embedded throughout this modeling process. Furthermore, model-based reasoning guided the design of the data collection instruments, by asking students to breakdown their model and explain in detail why they were making the design decisions that they made.

7.6 Theoretical and design conjectures

An important step in the DBR cycle is to articulate a way of conceptualizing and carrying out research on learning environments (Sandoval, 2014). To align our learning design with our research design we employed a conjecture map. Conjecture mapping is a useful tool for aligning the design conjectures of a DBR study with the embodiment (based on research grounded in theory), along with mediating process and the observed outcomes of the study (Sandoval, 2014). A conjecture map distinguishes conjectures about how a learning design (e.g., an educational environment) should function based on theoretical conjectures, and explains how the functioning produces intended learning outcomes. Figure 8 depicts a conjecture map for the design of the learning environment and the intended learning outcomes for this study. The conjecture map presents a high-level conjecture that is a principle of how to promote some desired learning or student attribute. The embodiment refers to the features of the learning environment in terms of tools, pedagogical practices, discursive practices, and task structures. For our intervention, these included the MATLAB programming environment, model-eliciting activity (MEA) design principles (R. Lesh et al., 2000), and productive failure (Kapur & Bielaczyc, 2012). The mediating processes refer to those salient performances or products expected to result from the embodied elements. The outcomes are the result of the mediating processes and these are the elements that are ultimately measured (Sandoval, 2014).

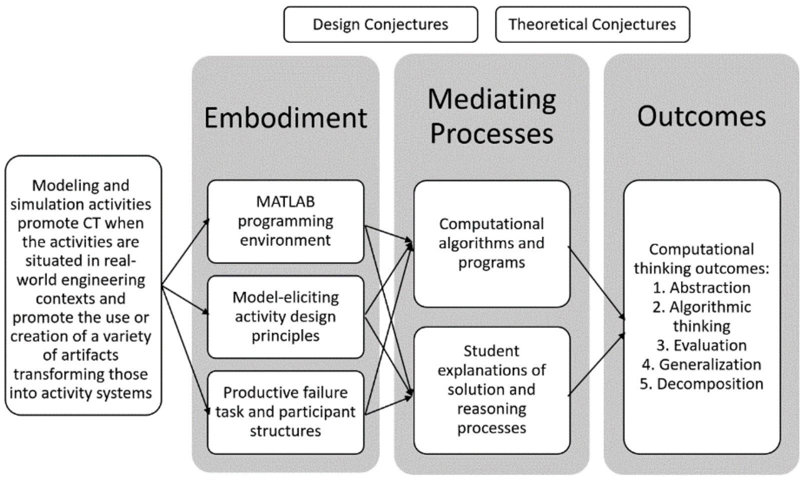
Design Conjectures
Theoretical Conjectures
Embodiment
Mediating Processes
Outcomes
Modeling and simulation activities promote CT when the activities are situated in real-world engineering contexts and promote the use or creation of a variety of artifacts transforming those into activity systems
MATLAB programming environment
Model-eliciting activity design principles
Productive failure task and participant structures
Computational algorithms and programs
Student explanations of solution and reasoning processes
Computational thinking outcomes: 1. Abstraction 2. Algorithmic thinking 3. Evaluation 4. Generalization 5. Decomposition

Figure 8. Conjecture map overviewing the design of the Building the Model phase.
© 2021 American Society for Engineering Education

Our high-level conjecture is grounded in a situative perspective, such as model-based reasoning. A situative perspective considers learning as an activity grounded in a social context where multiple people generate and use a variety of artifacts to transform those into productive practice (Sawyer & Greeno, 2009). Thus, our higher conjecture is that modeling and simulation promotes CT when the activities are situated in real-world engineering contexts, and when engaged in the process of creation and use of artifacts (i.e., computational models). As displayed in Figure 8, the three pieces to the embodiment that constitute the elements of situatedness are (1) the incorporation of the MATLAB programming environment, (2) the use of model-eliciting activity (MEA) design principles to design the activity (Diefes-Dux et al., 2004), and (3) the use of productive failure to design both the task and participant structures (Kapur & Bielaczyc, 2012). The MATLAB programming environment enabled the students to create computational algorithms and programs. Likewise, the MEA design principles required students to create models, also leading to the formation of computational algorithms and programs. The MEA design principles, via the construct documentation principle required students to evaluate and explain their models, leading to the mediating process of student explanations of solution and reasoning processes (Diefes-Dux et al., 2004). Finally, the use of productive failure in the design guided students in working together to produce pieces of the work, leading to the necessary student explanations as they worked within a diverse team.

It was hypothesized that these two mediating processes (i.e., computer algorithms and student explanations), lead to CT outcomes. First, abstraction was accomplished by the student during the building of the algorithm as a mediating process. Abstraction consisted of essentially stripping away details, something the students constantly did to the physical system as they moved into the computational domain, and then made explicit when delivering explanations of their models. Algorithmic thinking was elicited during the programming phase as students put computational commands and programming logic into a certain progression, in order to accomplish the goals of the assignment. Evaluation was elicited within the student explanations, as students validated and verified the output of their models and compared them against their own expectations, as well as compared to external expectations (such as the instructor or the requirements of the assignment). Generalization was prompted within the programming of the model itself as students reused code, as well as how students explained their solution would be useful in other contexts. And finally, decomposition was used both within the breaking up of the algorithm, as well as the student explanations of why they did certain tasks separate of others.

7.7 Pedagogical Design

Models and modeling and productive failure were the two pedagogical approaches used for designing the modeling intervention (Kapur, 2008; Louca & Zacharia, 2012). Specifically, the models and modeling perspective informed the activity design to elicit model-based reasoning, while productive failure largely informed the design of the participant structures and learning environment (Kapur & Bielaczyc, 2012; Louca & Zacharia, 2012). The activity itself was also structured through MEA design principles (Diefes-Dux et al., 2004). Assessments were rubric-based and designed in order to assess the students' ability to develop their modeling solution. The pedagogical design was implemented in a capstone design course within a biological engineering department focused on food and pharmaceutical process design. The learning objectives for the course were (1) the ability to develop unit operation designs (in our case through modeling), (2) improving computer skills, and (3) developing an ability to work in teams. While the rubrics were used for classroom grading purposes, for research purposes we qualitatively analyzed the raw data in the form of student-generated artifacts, instead of quantitative data condensed in the rubric scores.

7.8 Models and modeling

Modeling is a common engineering practice and has been studied in multiple contexts within engineering education. One mechanism that has been commonly used to incorporate modeling into engineering contexts has been via model-eliciting activities (MEAs). MEAs have been heavily studied by the engineering education research community (Diefes-Dux et al., 2004, e.g., 2006; R. Lesh et al., 2000). Modeling-eliciting activities were designed guided by six principles (Diefes-Dux et al., 2004; Zawojewski et al., 2008). The six principles along with how those were applied to the design of the MEA are described in Table 3.

Table 3. MEA design principles and their application in our intervention.

Principle	Definition	Application
Model-construction principle	"The Model-Construction Principles ensures that the activity requires the construction of an explicit description, explanation, or procedure for a mathematically significant situation" (Diefes-Dux et al., 2004, p. 5).	In our intervention, the students built an algorithm consisting of mathematical and computational structures in order to describe a canning sterilization process.
Reality principle	"The Reality Principle requires that the activity be posed in a realistic context and designed so that students can interpret the activity meaningfully from their different levels of mathematical ability and general knowledge" (Diefes-Dux et al., 2004, p. 5).	In our intervention, the students were given an email from a systems' engineer, very similar to what could be expected as industry professionals. In addition to requirements, the email included blueprints and data.
Self-assessment principle	"The Self-Assessment Principle ensures that the activity contains criteria the students can identify and use to test and revise their current ways of thinking" (Diefes-Dux et al., 2004, p. 5).	In our intervention, the students were able to visually output their models using MATLAB and to compare them to given operating conditions (desired output temperatures of the food product and amount of destruction of harmful bacteria).
Model-documentation principle	"The Model-Documentation Principle requires students create some form of documentation that will reveal explicitly how they are thinking about the problem situation" (Diefes-Dux et al., 2004, p. 5).	The students were asked to complete and turn in documentation for each phase of the activity, asking them to explain their decision-making processes thoroughly.

Table 3 continued

Construct share-ability principle	"Often termed the Generalizability Principle, this principle requires students produce solutions that are shareable with others and modifiable for other situations" (Diefes-Dux et al., 2004, p. 5).	In our intervention, students developed models of sterilization processes that could be used in a host of various sterilization contexts. The students also discussed and decided on different assumptions made and the effects of those assumptions.
Effective prototype principle	"The Effective Prototype Principle ensures that the model produced will be as simple as possible yet still mathematically significant" (Diefes-Dux et al., 2004, p. 6).	In our intervention, students developed and applied their understanding of how sterilization, heat transfer, and degradation happen within biological processing systems.

Another important aspect to MEAs is the need for different correct ways to solve the learning activity. In our design, there were different assumptions or mathematical ways in which the problem could have been solved. For example, the students were given enough information to solve microbial degradation in multiple ways. Students could have also assumed various geometric qualities of the container, in this case a can, within the sterilization process (e.g., as a slab or as an infinite cylinder). Additionally, different qualities of the heating process could have been assumed, such as whether or not convection was important at the boundary, causing the resulting solution method to differ. As such, there were multiple different relevant solution pathways to the given MEA.

7.9 Productive failure

Productive failure is a method of teaching that allows students to explore different solution pathways to complex and ill-structured problems without instructors providing structural support, often leading to students failing initially before arriving at an acceptable solution (Kapur, 2008). Students who learn this way often perform better on subsequent assessments (Kapur, 2008; Kapur & Bielaczyc, 2012). Productive failure naturally fits well within the domain of biological processing systems as designing these systems involves engagement in complex and ill-structured problem solving practices, a beneficial context for productive failure (Kapur, 2008). Such practices are often encountered by professional engineers (Gainsburg, 2006). Whereas MEA design principles were used to design the activity itself, productive failure was used to deliver the activity and orchestrate the implementation.

Productive failure has both design requirements for the activity and for the environment around the activity. Productive failure operates within two phases: generation and exploration of RSMs and consolidation and knowledge assembly (Kapur & Bielaczyc, 2012). In the first phase, students should be given a problem in a narrative form, and be embedded in an environment that allows for collaboration among students (Kapur & Bielaczyc, 2012). For our intervention, students were given a complex modeling problem on food sterilization. During the early phases of the problem, researchers and the instructor did not give direct help to the students but rather encouraged them to explore the different ways to solve the problem. For instance, rather than directing the student to a best process for solving a particular aspect of the problem, the instructor probed the students to think about different assumptions that could be made and the repercussions of those assumptions.

In the second phase of productive failure, students should have access to other student and expert created solutions to the problem (Kapur & Bielaczyc, 2012). In our intervention, after the models were built, students had the opportunity to look at other student solutions and discussed how the models operated differently than theirs. This process was documented by the students so that they could think through what decisions the others made and compared them their own.

The food sterilization problem was adopted from previous iterations of a biological engineering course. This problem was then adapted to integrate MEA design principles and orchestrated to follow a productive failure approach. The adaptation of the original problem to follow the MEA design principles and orchestration via productive failure was performed as part of a semester-long learning design project for a cyberlearning research and development doctoral course. Feedback on the redesign the problem was then gathered from subject matter experts in engineering education, learning sciences, and the domain expert who was also the instructor of the course. The results of this study report findings from the first classroom implementation of the problem; however, the problem has been deployed a second time with minimal changes.

7.10 Research Design

Once the learning intervention was designed guided by pedagogical principles, the next step in the DBR cycle was to implement and test it in a working classroom. Our guiding research question was: *What types of CT outcomes emerge when engineering students build computational models situated in real-world contexts?*

7.10.1 Context and participants

The intervention was deployed in a required upper-division senior engineering capstone course on food and pharmaceutical processing within a biological engineering department. This context was selected for multiple reasons. First, the discipline of biological engineering, although highly interdisciplinary, does not get substantial exposure to computation at the undergraduate level compared to other fields such as electrical and computer engineering. Second, we wanted the findings to be derived from male and female populations, and biological engineering was an ideal candidate due to the traditionally higher rates of female enrollment in biological and agricultural sciences (National Science Board, 2018). Third, the research team has extensive disciplinary expertise in this domain, and therefore qualified to identify the interplay of domain knowledge and CT. The intervention was given over approximately four weeks in the middle of the semester. The class focused on a senior design project in addition to multiple smaller design/modeling problems throughout the semester on various food manufacturing processes. The designed educational intervention focused on food sterilization, specifically within a canning operation. As such, the current designed intervention primarily focused on students working alongside team members on a computational modeling assignment to design a sterilization line (Appendix A).

The students already had exposure to programming coursework and basic principles of transfer physics in earlier engineering courses. However, computer programming abilities within this engineering discipline are far often less than other engineering disciplines such as electrical or computer engineering. Much of the classwork up until the final year was focused on transfer physics and fundamental sciences (i.e., chemistry and biology). Most students had taken a modeling class prior to the course in this study, which was a requirement for their major plan of study. Additionally, all students had previous exposure to MATLAB as part of their first-year introductory engineering coursework. For this intervention students used MATLAB software to build their computational models. The students worked within teams of three to four members to both plan and evaluate their models, however each student was expected to do the modeling assignment individually.

The total number of participants in the study was 45 students distributed among 15 teams. Students were not asked to report their gender or race information at any point during the study. However, according to publicly available statistics from the biological engineering department at the institution, there are approximately 52% female students in the entire program. In the same institution, the percentage of all female students throughout the entire college is approximately 26%. Since the class where the intervention took place was a required course for all students in the major, the class' demographics was considered to be representative of the biological engineering department. Fifteen individual student solutions (n=15), consisting of reports and MATLAB code representative of all available groups were chosen from the 45 samples for analysis in this study. The fifteen individual students were chosen purposefully based on being exemplar work from each of the teams. That is, from each of the 15 teams, the most representative solution was selected for further analysis.

7.10.2 Classroom Implementation

Given that in DBR implementations can vary based upon both actors and the context (Kelly, 2004), in order to ensure transparency of the research, the implementation as it played out in the classroom needs to be described. The MEA and productive failure principles were combined to create a four-phased learning intervention that allowed students to go through an entire modeling and simulation cycle. The classroom implementation is outlined in Table 4 showing what the activities were, what the students did each day, and the role the instructor took. Note that for the purposes of this study, only the second phase, building the model, was used for data analysis purposes.

Table 4. Four-phased learning design of the modeling process, highlighting the phase used for this study.

Phase	In-Class Activity	Student Deliverable	Instructor Role
Planning the model (Week 0)	Students work in teams to produce an outline of a plan of assumptions, equations, and layout of computational model.	Students individually turned in a template with a description overviewing their plan for the model.	Instructor encourages students to explore problem space.
Building the model (Weeks 1-3)	Students individually build their computational models in MATLAB programming environment.	Students individually turned in a report template with their detailed solution and commented computer code.	Instructor refrains from answering questions in specific detail, but encourages problem space exploration.
Evaluating the model (Week 3)	Students meet in multiple groups to evaluate their own model against their peers.	Students individually turned in their generated notes from their multiple group discussions.	Instructor emphasizes that there is no one way to correctly answer the problem.
Reflecting on the model (Week 4)	Students individually answer targeted reflection questions about their entire modeling process.	Students individually turned in a completed reflection template.	Instructor encourages authentic reflection of the process.

In the first class, students were given an overview of the problem and were asked to work within an already assigned group on planning how they would solve the problem and build the computational model. During this time the instructor and researchers mainly answered questions about the problem design, without telling the students how to solve the problem but rather to expand their thinking. Students were then assigned to build the computational model and given both a report template and programming file template to be filled out (Appendix B and C), which is the primary data set for the current study. Both of these artifacts, the report template and programming file template were produced by the students individually and not as part of a team exercise.

Between the first class where the students were given the MEA and the second class three weeks later where the students brought in their solutions, there were a host of different tasks that students accomplished. First, during this time, the instructor lectured over various subtopics regarding food sterilization such as microbial death kinetics, finite difference analysis, and heat transfer. Additionally, students had other assignments and activities as part of the class relating to their capstone projects. These included homework assignments on more basic problems from the textbook. Students were also expected to be working on their MEA solution during this timeframe and be incorporating aspects from the subtopics being covered in class.

In the second class, approximately three weeks after the first class, students brought in their modeling solutions consisting of both the report template and completed programming template (Appendices B and C). The students then got into their groups and discussed how they had ended up solving the problem. Students also met with randomly assigned other groups to look at how their models differed from a diverse group of student solutions. Instructors acted as facilitators, having students move between groups but not providing them with specific answers to solve the problem. Finally, all students individually completed a short reflection report that asked them to think about how they solved the problem and what changes they would make to their solution if given more time. This allowed students to think about ways they would improve their models and performance on the modeling assignment beyond the multiple iterations they had already gone through.

7.10.3 Data Collection

All of the data was collected in the naturalistic context of a classroom environment. Various templates were used to elicit detailed student responses of their reasoning and decision-making processes through the problem. This study focuses on the model-building process, looking at how students used CT during this phase of the intervention. Multiple artifacts were collected including a model-building template which asked the students to write up their modeling process and thinking process (Appendix B). A second programming file template was given to them with instructions on how to comment the code and what to include in the commented programming files (Appendix C). While students were allowed and encouraged to work together if they needed support, all students individually had to turn in each of these artifacts.

7.10.4 Data Analysis

Data analysis was performed on each artifact students individually created during the building the model phase and were used as evidence of CT outcomes within modeling and simulation solutions. To approach our analysis, we used deductive thematic analysis. Thematic analysis at its core is “a method for identifying, analyzing and reporting patterns (themes) within data” (Braun & Clarke, 2006, p. 79). Thematic analysis can be either inductive, where the themes are completely emergent from the data themselves, or deductive (theoretical) where the themes are emergent based on prior theory and literature (Braun & Clarke, 2006). Thematic analysis can account for both manifest and latent content within the dataset (Helene Joffe, 2012). For our data analysis procedure, the data was coded by identifying utterances and student developed structures that showed the presence of CT as defined categorically by Curzon, Selby, and colleagues (Curzon et al., 2014; Selby & Woollard, 2013). Once the initial CT practices were identified, we applied an inductive approach to further characterize CT outcomes. The data analysis procedure followed a six step process (Elo & Kyngäs, 2008): (1) familiarizing oneself with the data, (2) defining initial codes from the literature, (3) search for themes amongst the coded data, (4) reviewing the identified themes, (5) naming and defining the themes, and finally (6) reporting out on the themes.

7.10.5 Initial coding categories

Our research operationalized CT into five separate practices as initially proposed by Selby and Woollard (2013).These are categorically abstraction, algorithmic thinking, evaluation, generalization, and decomposition. Initial definitions were adapted from Curzon, Selby, and colleagues (Curzon et al., 2014; Selby & Woollard, 2013), but were further refined as we iteratively moved throughout the thematic analysis, and clarity emerged for the coding rubric. Table 5 lists the refined definitions for each CT practices.

Table 5. Final refined definitions of CT practices as adapted from the literature.

CT Practice	Definition
Abstraction	Making problems easier to solve/think about by selectively removing/hiding unnecessary complexity by making assumptions about how the problem should operate or by neglecting or adding components to the problem.
Algorithmic Thinking	Setting up problems so that they can be solved in a stepwise manner by looking at the procedural aspects of the problem; or by acknowledging the uses and effects of solution structures temporally and procedurally throughout the solution.
Evaluation	Comparing one's own solution against various criteria either given by the problem itself or against criteria brought to the problem by the student themselves. This can also be a statement describing desirable qualities of the final solution in terms of function, display, or other final form characteristics.
Generalization	Reusing previous solutions, methods, or constructs for the current problem; or the use of current solutions, methods, or constructs from the current problem to hypothesize about their use in future contexts.
Decomposition	Thinking about how the pieces of the problem can be used separately, strategically breaking down the problem for easier solution or use, or breaking the problem down into subcomponent structures, pieces, or areas.

7.10.6 Emergent themes

For our deductive thematic analysis process, student *Building the Model* templates (see Appendix B) and coding file templates (see Appendix C) were coded for utterances and developed structures that fell into one of the initial coding CT practices listed in Section 6.5. Each utterance and structure were then broken into different outcomes or themes that mapped to the corresponding CT practices. CT outcomes were defined as the fine-grained subcategories that were observed from the student artifacts that fell within each of the CT practices. The unit of analysis was at the idea or sentence level. After the initial coding process, definitions for each of the outcomes were further developed and the transcripts were again reanalyzed a second time by the primary researcher for (a) refining the coding structure and looking for additional uncoded observations and (b) evaluating already coded utterances and structures. Thus, themes of outcomes emerged from within each of the CT practices.

7.10.7 Trustworthiness considerations

Multiple steps were performed in order to ensure trustworthiness and reliability throughout the data analysis process. First, there were multiple data streams analyzed through both the student reports and the student coding files allowing for some triangulation of results (Nowell et al., 2017). Participant quotes were used often to show that the results are an accurate reflection of the data itself (Elo et al., 2014). The initial CT practices and CT outcomes of the codebook were developed by the lead researcher and validated and refined with three other subject matter experts, one in the field of computer science, a second one in the field of computational biology, and a third one in the field of computational science education.

Additionally, interrater reliability was used as a metric for determining the trustworthiness of the thematic analysis (Vaismoradi et al., 2013). For this study, a second independent researcher was provided with a sample of data, representative across analyzed transcripts, at the unit of analysis level (sentence/idea level). The lead researcher overviewed some training data and introduced the codebook to the second rater. Then, the independent researcher individually coded each sentence into a CT practices and outcome. The first round of interrater reliability was performed with a small sample of the data. After the first round of interrater reliability, the two raters came together to make final improvements to the definitions in the codebook. The final round of interrater reliability with the second independent researcher achieved a percent agreement of 82% on the CT practices and 81% on the CT outcomes, focusing on over 20% of the coded items. This was determined to be satisfactory given the deep disciplinary knowledge embedded within many of the student quotes and that the lead researcher had met with three interdisciplinary experts about the robustness of the codebook.

7.11 Results

There were multiple coded themes found in each of the CT practices with *evaluation* having the highest number of outcomes (7) and *generalization* having the lowest number of outcomes (1). Each of the practices had at least one outcome that emerged from the 15 transcripts. Table 6 overviews each of the CT practices along with the outcomes that were found for each of them.

Table 6. Outcomes for each of the CT practices.

Abstraction	Algorithmic Thinking	Evaluation	Generalization	Decomposition
Dynamic to static	Indication of later use/effect	Time efficiency	Reuse of code	For ease of use
Infinite to finite	Conditional logic	Code flexibility		For organization
Multiple to single		Design Criteria		
Ranges to values		Code accuracy		
		Code complexity		
		Solution usability		
		Debugging		

Abstraction and *evaluation* were seen in all 15 students' reports throughout the thematic analysis, with *algorithmic thinking* being seen in 13 student reports. *Decomposition* was seen in 14 student solutions, with *generalization* being the least seen practice, only being seen in seven student solutions. For example, for abstraction, the theme of infinite to finite was seen in ten student solutions while the ranges to values code was seen in eleven student solutions. In the subsequent sections, we will go over each of the themes and give definitions as well as quotes from the students work in order to provide evidence for each of the outcomes within each of the CT practices.

7.11.1 Abstraction

The keystone of CT is abstraction (Grover & Pea, 2013). For our framework, the definition considered for abstraction entailed making problems easier to solve/think about by selectively removing/hiding unnecessary complexity (Curzon et al., 2014). There were multiple outcomes that manifested themselves throughout the student solutions that were coded as abstraction due to their simplifying nature within the dataset (see Table 7).

Table 7. Observed outcomes within the practice of abstraction.

Abstraction	Definition	Quote(s)
Dynamic to static	Making factors or variables that vary in respect to other variables or aspects of the problem and making them constant or unchanging.	"Thermal properties **stay constant** throughout temperature change."
Infinite to finite	Making aspects of the problem or system that are infinite or continuous and making them finite or discrete in nature.	"This was assumed to be true because there was no way to truly calculate the average temperature of the can **unless infinite nodes were used**."
Multiple to single	Simplifying aspects of the problem that have multiple dimensions or factors and simplifying the dimensions or factors considered in the solution through choice of one or neglection of factors; or creating limits or boundaries to what is considered to be affecting or influencing the considered system.	"The times for each bacteria are outputted **and the largest is used** as the sterilization time."
Ranges to values	Making factors or variables that have multiple possible values (within a range or list) and simplifying the condition into a single value, such as worst-case scenario or average across a range.	"The value for **Ea was on the low end, and the value of Z and D250 were on the high end of the ranges** given for C. Botulinum."

Many of these abstraction outcomes mapped to various simplification outcomes that are indicative of problem solving by reducing the complexity of the problem. Each of the outcomes was seen primarily in different areas of the problem. For example, taking the infinite to the finite was primarily demonstrated when converting mathematical operations into their computational counterparts. This was observed as student moved between the infinite world of calculus to the finite world of loops.

Many of the other themes were a result of problem structure. For example, the problem gave the students many different microorganisms to focus on throughout the sterilization process. One example of taking the multiple to the single was students choosing to focus on one specific microorganism rather than focusing on all of them. Additionally, taking ranges to values was often the result of problem structure as students had to take ranges (such as operating temperature ranges) and make decisions about how to handle these in their computational models.

7.11.2 Algorithmic thinking

Algorithmic thinking is often hard to distinguish from mathematical thinking and CT (Weintrop et al., 2016). For this analysis, we considered algorithmic thinking as setting up problems so that they can be solved in a stepwise manner (Curzon et al., 2014). This outcome manifested itself in multiple ways throughout the transcripts analyzed for each of the 15 students. Table 8 further elaborates on examples and definitions.

Table 8. Observed outcomes within the practice of algorithmic thinking.

Algorithmic Thinking	Definition	Quote(s)
Indication of later use/effect	Indicating that information will be useful for a later process or a later point in the code. Indication of what certain variables or structures will cause later in the solution or code or explanations of location of code for effect.	"%It is important that this code is after the center so the initialization of the array at 6 is not ran through the loop, this for loop was made so it creates the temperature profile for the cylinder points 3-6 boundary."
Conditional logic	Indication of an either/or condition based on previous conditions.	"**If the conditional statement is not met, then the solution is unstable** and it is not worth continuing the program."

The themes within algorithmic thinking demonstrate a progression and temporal understanding that the students exhibited during their model-building process. Algorithmic thinking was shown in multiple ways throughout the student model-building solutions. Students, when discussing their code, discussed the effects and use of certain variables later in the code. For example, students discussed how a variable defined at the beginning of the code was used later in the code or how looping outputs were used later in the code. Additionally, some students discussed how certain aspects or variables within their code would affect the code later on.

7.11.3 Evaluation

Evaluation as a practice had the most themes identified through the thematic analysis. Evaluation was considered as the act of comparing the solution to either an ideal, another person's solution, or to the requirements of the problem (Curzon et al., 2014). Evaluation was constantly used throughout all of the modeling and simulation process, specifically for validating a solution (Czocher, 2018). Throughout the themes, it can be seen that this practice manifested in multiple ways such as comparing the solution to a time ideal, to a stakeholder desire, or to an accuracy requirement. Definitions and corresponding samples of quotes are provided in Table 9.

Table 9 Observed outcomes within the practice of evaluation.

Evaluation	Definition	Quote(s)

Time efficiency	Evaluates code or solution based on the amount of time taken to arrive at solution or time wasted.	"**This saves computational time** if a bad system has been created."
Code Flexibility	Evaluates code or solution based on an ability to easily change later.	"**The for loop was useful because it allowed easy changing of the amount of time the can was to be sterilized** while still calculating the temperature profiles."
Design Criteria	Evaluating the code or solution against the wishes and requirements of stakeholders.	"The temperature matrices were then converted to F **because that was the unit the customer requested the report be in."**
Code Accuracy	Evaluating the code or solution in regards to the actual or perceived accuracy of solution method decisions.	"**It is unlikely that the value is off by any significant amount** as models using more nodes showed similar values."
Code Complexity	Evaluating the code or solution in regards to the amount of work, time, or effort needed to create the solution.	"These functions were chosen to…**utilize the power of built in Matlab functions**."
Solution Usability	Evaluating the code or solution in regards to ease of use for themselves or others.	"This allows us to see how temperature changes over time and position, and **formats the graphs in a clear way."**
Debugging	Evaluating the code or solution periodically throughout the creation or solution process to ensure ability to run, reasonableness, or to collect feedback about the solution.	"%is just a plot of the heating temperature profile, I had this to just check that the heating profile looked correct."

Evaluation was prevalent throughout the student solutions and thus had the most themes of any of the CT practices. Students used evaluative outcomes to compare their solution against other ideals for efficiency, flexibility, the desires of the stakeholders, accuracy, complexity, and usability. Some of these outcomes were likely the result of the problem structure, such as stakeholder requirements.

Additionally, evidence of debugging was exhibited, which is evaluating the algorithm as one is creating it. While many of the other evaluating outcomes were observed in the context of assessing the final solution (accuracy, efficiency, and usability), some of the outcomes were evaluations during the modeling process itself (debugging, complexity, and flexibility). These in-process evaluations informed the students about how their code should have been structured.

7.11.4 Generalization

Generalization had the least number of themes emerging from the analysis with only one outcome. Some definitions of CT combine abstraction and generalization into one CT practice (Grover & Pea, 2013). However, for this study, we broke them into two specific coding practice. Generalization was considered as instances where the student used previous solutions or ideas for the current problem, or used current solutions or ideas to project onto future problems (Curzon et al., 2014). In contrast to abstraction, which was coded as structures that aimed to simplify the problem in order to make it easier to solve, generalization was primarily connected to how students transferred information within and across contexts. Table 10 provides definitions for this theme and corresponding samples of quotes.

Table 10. Outcome observed within the practice of generalization.

Generalization	Definition	Quote(s)
Reuse of code/equations	Reusing or repeating portions of code or equations throughout the solution method.	"Additionally, the thermal properties are redefined for each temperature using the **same method described above**."

Generalization was rare within the building the model phase, and thus is only represented with one theme. The ways in which students used generalization, consisted of reusing codes and equations for different purposes within their modeling process. For example, students may have reused a section of code containing a while loop to both calculate the heating profile of the can as well as to calculate the number of microorganisms later in the code. Merely copy and pasting code in multiple places was not considered reuse, but rather, where students mentioned in their report writing or code comments the reuse of the code or equations.

7.11.5 Decomposition

Finally, decomposition had two themes emerging from the dataset. Decomposition was considered as any outcome where the student intentionally broke the problem into smaller pieces for the purposes of solving, using, or thinking about their solution (Curzon et al., 2014). While there were not many themes that were coded as decomposition, the majority of participants at least exhibited some form of decomposition during the solution process. Table 11 outlines the various emergent themes.

Table 11. Outcomes observed within the practice of decomposition.

Decomposition	Definition	Quote(s)
Organization of larger solution method	Decomposing the code or solution method as an organizational tool.	%INITIALIZE OUR DATA MATRIX … %FINITE DIFFERENCE METHOD AND CALCULATION

Ease of use of program	Decomposing the code or solution method so that the program or solution is easier to use.	(Student broke up code into large sections.) "The run time for this section is large, so **it should only be run once to determine the temp."**

While there are two themes that emerged from the decomposition practice, decomposition outcomes were often implicit in students' artifacts and explanations. For example, organization of larger solution methods was often students breaking their code into multiple subsections, with commented titles or commented section headers. This allowed for a student to identifying sections or pieces of the code within their programming script. The ease of use of program, was often students breaking apart their code into separate functions, allowing them to be called upon at any time in the larger programming script. Although students rarely discussed in detail their decomposition outcomes, it was apparent through their solutions and created artifacts that students were in fact breaking the problem down. Within decomposition, repetitive outcomes were only coded once even if there were multiple occurrences of the same outcome. For example, if a student broke their code into five functions, it was only coded as *decomposition* once, not five times.

7.12 Discussion

It was evident that all of the CT practices were used within the process of building a computational model to varying degrees. In this context, each theme was coded as a practice and each had at least one CT outcome identified. This characterization lends itself to supporting the claim that activities that build computational models are prime for eliciting and practicing CT outcomes within an undergraduate engineering setting. Therefore, a first contribution of this study characterizes CT in the context of modeling and simulation as described on Table 12.

Table 12. Characterization of CT in the context of building a computational model

CT Practice	Definition in the context of building a computational model
Abstraction	Simplifying problems through assumptions that simplify dynamic variables to static variables, simplify infinite factors to discrete factors, remove dimensionality through variables or boundaries, or simplify ranges to single values.
Algorithmic Thinking	Setting up problems so that they can be solved in a stepwise manner by indicating the uses and effects computational structures will have temporally through the code or indicating the effects of previous conditions on computational structures.
Evaluation	

	Comparing one's own solution against various criteria such as design criteria, the flexibility of the code, the efficiency of the code, the accuracy of the code, the complexity of the code, the usability of the solution, or the runnability of the code through debugging.
Generalization	Intentionally reusing of mathematical methods, computational structures, or equations throughout the solution.
Decomposition	Strategically breaking down the problems into pieces either for the functional use of each piece of the solution or for the organization of the larger solution method.

However, the degree to which each practice of CT was used varied within the model-building activity. The student solutions provided substantial evidence of algorithmic thinking, abstraction, and evaluation outcomes, along with corresponding outcomes. This was expected as students wrote a computer program (algorithm) that was a model (abstraction) of a real-life process. Modeling is a skill closely tied to abstraction (Gainsburg, 2006; Magana & Silva Coutinho, 2017). Gainsburg (2006) argued that "school modeling problems are essentially generalization or abstraction exercises" (p. 30). Thus, this close tie to abstraction outcomes within the modeling and simulation process was expected, as it has been considered as cornerstone to CT (Grover & Pea, 2013; Wing, 2008). The prevalence of algorithmic thinking within the modeling and simulation process was also expected. Algorithms are essentially a "step-by-step set of instructions that can be carried out by a device" (Selby, 2015, p. 81). A computer program, like the ones that the students created in the modeling activities, is of this same nature. So, it was expected that algorithmic thinking would be demonstrated in full display. Algorithmic thinking was often displayed by students when they were explaining why they had developed the model the way they did, looking at temporal effects and uses of information, and programming structures throughout the coding files. The programming process pushed students to think about where information should be put within the programming file, how it would be called upon, as well as the effects it would have later on in the program.

The high prevalence of evaluation outcomes within the data set was encouraging, in that students were demonstrating that they were continuously evaluating their model within a host of different contexts such as accuracy, efficiency, purpose, usability, and solution flexibility. Previous studies have found that evaluation activities, specifically validation of the model, are continuously used and engaged in during the modeling process (Czocher, 2018). The results of this study certainly align with this trend. Additionally, our results show the presence of both validation and verification outcomes within large parts of the model evaluation process (Magana, 2017). For instance, validation outcomes such as evaluating for solution accuracy and evaluating for stakeholder desires were observed. Verification outcomes such as debugging and solution usability were also observed. Understanding that the model does what it was designed to do (verification), as well as demonstrating that it actually represents the physical phenomenon desired (validation), are both needed for an accurate computational model.

Decomposition was prevalent but was rarely explicitly discussed, rather implicitly showed up as breaking the code into separate functions or subsections. Generalization was scarce, indicating that either the problem was not particularly well scaffolded to allow for this outcome to emerge, or that this outcome was less prevalent in the process of building a model. We speculate that generalization and explicit decomposition would be more prevalent within other phases of the modeling process.

A second contribution of this study relates to the exemplification on the use of DBR where outcomes of CT were operationalized in an engineering education context. The proposed learning design resulted in findings that are a discipline-specific given the situated characterization of CT within our DBR implementation. Continued iterations of the DBR cycles should reveal different and more comprehensive ways CT is used, as well as the ways that learning designs can be adjusted to better elicit CT outcomes. This continued work will be used to build an understanding of how CT is elicited within our discipline-specific setting. While not generalizable to all settings, our goal is to continue to refine an outcome space to describe how CT is used within modeling and simulation practices.

7.12.1 Implications for teaching and learning

The results of this study have multiple implications for teaching and learning in an undergraduate setting. Specifically, the use of activities that promote building computational models can elicit and allow students to practice critical CT practices. Currently, much of the work on CT has been aimed at the K12 level (Kalelioğlu et al., 2016). However, when initially proposing the idea, Jeanette Wing (2006) suggested that “to reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability” (p. 33). Modeling and simulation activities allow for this seemingly fundamental skill to be incorporated into any classroom within an undergraduate engineering curriculum.

Additionally, our results indicate that some CT outcomes are used in more diverse ways than others within a model building process when guided with educational theories and practices such as modeling-based learning and MEAs. For example, while evaluation encompassed multiple outcomes, generalization only had one. Supplementary scaffolding within the activity structure may allow for more generalization and explicit decomposition to be made evident. However, structuring activities guided by MEA principles as well as productive failure principles allowed for all of the CT outcomes to emerge.

The results suggest that CT, within our DBR intervention, can be elicited by applying specific principles that were found to be useful as students generated artifacts associated with the building of a computational model. These principles include the following:

1. Create modeling and simulation experiences that combine engineering disciplinary knowledge with programming knowledge in the context of real-world experiences.
2. Elicit from students to create and connect multiple forms of representations such as diagrams, equations, flow charts, algorithms and computational models.
3. Provide opportunities to explore multiple approaches and solutions and provide guidance to consider other perspectives.
4. Orchestrate participant structures by allowing students to share knowledge and also compare and contrasts their models.
5. Prompt students to explain their thinking, assumptions, limitations and reasoning processes throughout the model building process and documentation.

However, the DBR method requires the learning design to be continuously improved by adjusting and improving the intervention based on previous iterations. In future iterations of the design, the researchers plan to incorporate: (1) more reflective opportunities throughout the entire process to encourage students to engage in more metacognitive behaviors during the entire modeling process and (2) implement more scaffolding into the report writing (in the form of technology-enhanced learning environments such as MATLAB Live) in order to streamline the coding and report writing process. Through these modifications the goal is to elicit the mediating processes of student explanations and meaning making reflective practices that may allow them to connect disciplinary knowledge with CT outcomes. An updated conjecture map is provided in Figure 9 reflecting these changes.

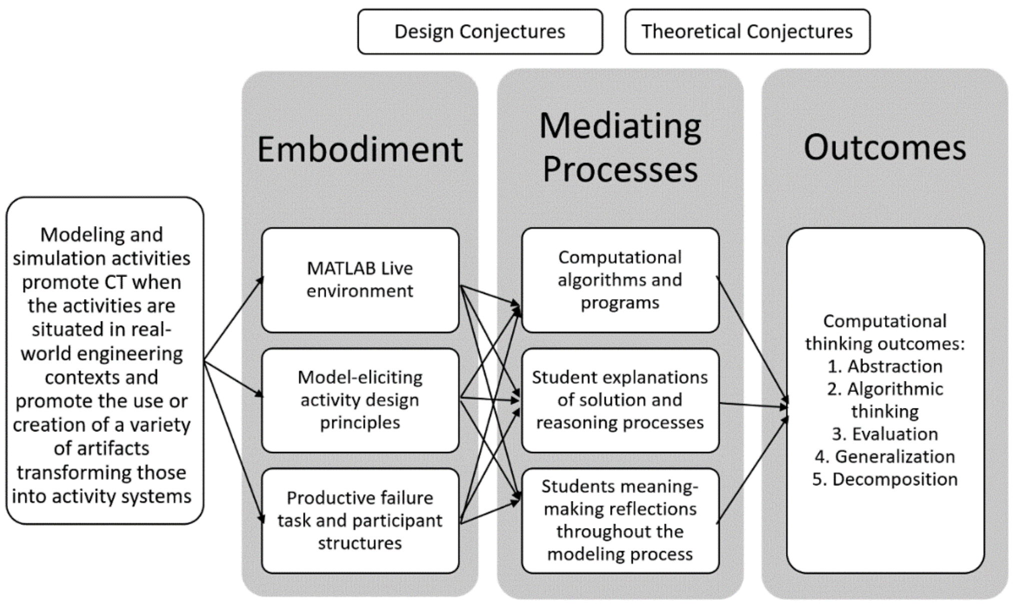
Design Conjectures
Theoretical Conjectures
Embodiment
Mediating Processes
Outcomes
Modeling and simulation activities promote CT when the activities are situated in real-world engineering contexts and promote the use or creation of a variety of artifacts transforming those into activity systems
MATLAB Live environment
Model-eliciting activity design principles
Productive failure task and participant structures
Computational algorithms and programs
Student explanations of solution and reasoning processes
Students meaning-making reflections throughout the modeling process
Computational thinking outcomes:
1. Abstraction
2. Algorithmic thinking
3. Evaluation
4. Generalization
5. Decomposition

Figure 9. Updated conjecture map overviewing the design of the Building the model phase.
© 2021 American Society for Engineering Education

7.12.2 Implications for engineering education research

This study has provided ways in which CT outcomes were elicited with modeling and simulation activities and has also provided a mechanism to close the gap from theory to practice by following a DBR approach. While modeling an d CT have been connected in the literature (Magana & Silva Coutinho, 2017; Sanford & Naidu, 2017; Weintrop et al., 2016) the nature of the relationship has been fuzzy at best. The results of this study demonstrate that modeling and simulation exercises within the engineering classroom can be used to elicit CT outcomes. However, future quantitative analysis is needed to further understand (a) how much CT-specific knowledge and programming skills students acquired during the modeling activities and (b) how the design of these activities impacted the learning of disciplinary knowledge along with CT-specific knowledge.

By implementing a DBR cycle throughout this study, we have demonstrated how we started from the educational problem of developing a computationally-skilled engineering workforce, and addressed it by designing scaffolded learning interventions grounded in theory. Results from our study therefore contribute to engineering education research not only by characterizing the CT outcomes elicited through the modeling activities, but also by demonstrating how we connected research to practice and then back to research via DBR. Specifically, in this study we described (a) how modeling activities can be designed and scaffolded with evidence-based pedagogical approaches such as the MEA and productive failure (Diefes-Dux et al., 2004; Kapur, 2008) and (b) how to conduct a qualitative investigation by leveraging student-generated artifacts, such as models and evidence of their reasoning with models, as data collection instruments. Engineering faculty have argued that including computation and programming within their classrooms is challenging due to an already crowded curriculum (Magana & Silva Coutinho, 2017). The results of this study show that modeling and simulation can be taught when combined with disciplinary material within the engineering classroom.

Furthermore, this study provides an example to other engineering education researchers on how to implement a DBR study by aligning theoretical conjectures with design conjectures and in the process contribute to student learning and the development of engineering-specific pedagogical frameworks. DBR is still an emerging method and one that continues to need clarification and demonstration (DBRC, 2003; Kelly, 2004). This paper demonstrated an approach to align DBR goals via conjecture mapping (Sandoval, 2014) within the discipline of engineering education. More DBR studies, especially in engineering education, are needed to connect educational theory to practice, and further contribute with theory development about how students learn with and about CT within the classroom (Wang & Hannafin, 2005).

7.13 Conclusions, Limitations, and Future Work

To conclude, this study used DBR to design an educational intervention that incorporated modeling and simulation activities into the engineering classroom. The intervention was based on design principles of multiple frameworks including: productive failure and MEAs. Together, these frameworks allowed for designing of a modeling and simulation approach that can be used across the engineering education curriculum. This approach resulted in a four-phased process which included: planning the model, building the model, evaluating the model, and reflecting on the model.

As with many DBR studies, our limitations relate to the contextualized generalization of our results. That is, the results presented in this study may only be generalizable to similar or related contexts. Additionally, only written work of the students after the intervention was analyzed and not behavioral data during the building phase of the intervention, thus there are likely some CT outcomes that could have happened, but were not elicited through the learning activities, or were not captured by the artifacts students created. Also, there were certain common components to CT that were not identified in the data set, such as pattern recognition, parallelization, and trial and error. One reason for this may be that these CT outcomes will be more emergent in other phases of the modeling process. Other phases of the modeling process would elicit different outcomes. As we continue our research to characterize CT outcomes within the entire modeling and simulation process our expectation is that additional aspects of CT that were not captured in this study, would be captured in other phases. Finally, gender, race, and ethnicity information were not collected from students, thus creating limitations as to the generalizability of the results to people from varying and diverse backgrounds, or when looking at populations primarily consisting of underrepresented groups in engineering.

For future work, because of the smaller sample size, the results presented here could be expanded using different student populations. This could include investigating the effect of prior modeling and programming coursework on student CT outcomes. It could also include expanding the intervention to include student reflections pertaining potential revisions to their models, and comparing their ideas of how their models should be changed versus the changes they actually made. However, concerning the current study, we believe that (a) our learning design properly elicited CT outcomes in the context of modeling practices, and (b) our detailed qualitative approach provided a lens to the characterization and understanding of how CT takes place in the context of engineering education. These two contributions provide an exemplar to connecting research to practice, and are also necessary for the development of discipline-based specific learning theories (Borrego & Henderson, 2014; National Research Council, 2012).

Our results suggest that abstraction, algorithmic thinking, decomposition, and evaluation were commonly and diversely used by students while engaging in the model building process, whereas generalization was used less frequently. However, the results indicate that building computational models do elicit all kinds of CT practices and are an effective way for integrating foundational CT practices into the undergraduate engineering curriculum.

7.14 Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. (DGE-1842166) as well as the National Science

Foundation under Grant No. (EEC-1449238). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. CHARACTERIZING COMPUTATIONAL THINKING IN THE CONTEXT OF MODEL-PLANNING ACTIVITIES

8.1 Abstract

Computational thinking (CT) is a critical skill needed for STEM professionals. While based in computer science, skills associated with computational thinking are necessary across industries and disciplines. Because of this, educational interventions that emphasize CT are needed. In engineering, one potential pedagogical tool to build CT is modeling. Modeling is an essential skill for engineering students. They apply their scientific knowledge to real-world problems involving planning, building, evaluating, and reflecting on created systems to simulate the real world. We evaluated a model-planning activity in a final-year undergraduate engineering classroom. Our study involved a case study methodology that implemented a modeling intervention to elicit CT practices in students as they planned their modeling approach. Thematic analysis was then used on student artifacts to triangulate and identify diverse ways that students used CT practices. We find that model-planning activities are useful for students to practice many aspects of CT, such as abstraction, algorithmic thinking, and generalization. We report implications for instructors wanting to implement model-planning activities into their classrooms.

Keywords: computational thinking, model-eliciting activity, models and modeling, engineering education

8.2 Introduction

Computational and data science are quickly emerging as needed skills across various industries and academic disciplines, yet educational institutions often fail to implement these skills into curricula (Finzer, 2013; Irizarry, 2020). This is partly because it has been challenging to assign departments within universities to teach computational and data science skills to undergraduates (Finzer, 2013). This issue is amplified by instructors who may not feel comfortable teaching these topics and expect students to learn these skills in other courses (Magana & Silva Coutinho, 2017). The net result is university graduates who cannot think through complex computational and data science problems.

These issues are especially apparent within engineering curricula, which are already perceived to be packed with courses by many engineering faculty (Magana & Silva Coutinho, 2017). One alternative to incorporating more courses into the curriculum is integrating computational thinking skills into existing courses. Research has shown that incorporating modeling activities into engineering classrooms allows students to simultaneously learn disciplinary and computational skills (Fennell et al., 2019; Lyon & Magana, 2021; Magana et al., 2013; Vieira et al., 2018). The additional benefit of this is that programming and disciplinary learning emerge through modeling, but computational thinking emerges through the applied model-building process as well (Lyon & Magana, 2021).

Models allow students to apply mathematical and computational concepts to solve real-world problems, and they have been heavily studied in engineering (Diefes-Dux et al., 2006; Jung et al., 2015; Lyon, Fennell, et al., 2020; Lyon & Magana, 2020a). Often the modeling process is divided into multiple phases, including planning, building, evaluating, and revising (Louca & Zacharia, 2012; Lyon & Magana, 2021; Magana, 2017; Shiflet & Shiflet, 2014). Each phase is crucial to developing practical, real-world modeling skills in engineering practice.

Here we investigated how computational thinking emerged during the planning stages of the modeling process. This is part of a larger study looking at computational thinking through the modeling cycle and builds on a previous study that looked at computational thinking that emerges as part of the model-building process (Lyon & Magana, 2021). For this work, our research question is: *What computational thinking practices emerged when students participated in model-planning activities?*

8.3 Background

Multiple bodies of the literature informed this study, including (1) computational thinking, (2) project planning, and (3) model-eliciting activities. Collectively these bodies of literature helped inform the design of our data collection and methods used to analyze the data. We give background information on each of the three bodies of literature below.

8.3.1 Computational Thinking (CT)

In recent years, computational thinking (CT) has been of particular interest to researchers and educators alike (Kalelioğlu et al., 2016; Lyon & Magana, 2020b). CT is essentially an umbrella term for different common thinking practices derived from computational sciences (Wing, 2006). While the exact definition of CT has been debated for some time, many definitions include common elements such as abstraction, generalization, algorithmic thinking, evaluation, and problem decomposition (Curzon et al., 2014; Kalelioğlu et al., 2016; Selby & Woollard, 2013; Weintrop et al., 2016).

Additionally, studies incorporating CT in the classroom have been surpassed mainly by researchers discussing CT in more general and definitional ways (Grover & Pea, 2013; Ilic et al., 2018; Lyon & Magana, 2020b). When operationalized, it is often in K12 settings rather than the undergraduate level (Ehsan et al., 2021; Rehmat et al., 2020). This has left a significant gap in the literature for concrete implementation strategies of CT or best practices for developing CT in undergraduate educational settings. When implemented, studies often use computational thinking to structure course content instead of finding concrete evidence or emergent practices of CT in students' work (Lyon & Magana, 2020b). This study addresses many of these gaps by focusing on evidence-based emergent CT practices using a defined CT framework (Curzon et al., 2014; Selby & Woollard, 2013).

8.3.2 Solution Planning

In terms of solving complex problems, planning often involves setting goals or deliverables of a problem and deciding on tasks and actions that are needed to solve the problem (Lawanto, 2010). Students' additional considerations while planning should be time constraints and meeting other criteria of the given problem (Dvir et al., 2003). Planning a solution pathway is a cognitive skill that plays a vital role in students' final solution (Lucangeli et al., 1998). This effect is seen even more when the problem that the student is planning to solve involves more transfer from material they are comfortable with or is ill-structured (Shin et al., 2003). Thus, planning the model before solving is essential for student success for ill-structured modeling activities, such as the one in this study.

While the literature has shown that planning is a valuable mechanism for problem-solving, many modeling interventions tend not to include it as a cognitive or metacognitive step before constructing models. Some modeling-based learning frameworks have students collect their experiences and observations before building their models, so having some level of planning is not without precedent (Louca & Zacharia, 2012). Within modeling interventions, students who plan their solution before often starting exhibit behaviors more aligned with subject-matter experts, focusing on their conceptual understanding of the material and avoiding trial-and-error solutions (Magana et al., 2019).

8.3.3 Model-Eliciting Activity (MEA)

Model-eliciting activities (MEAs) were originally used in mathematics contexts and have subsequently been used in engineering contexts for using real-world modeling problems in the engineering classroom (Diefes-Dux et al., 2004; R. Lesh et al., 2000; Z. Liu & Xia, 2021; Moore et al., 2013). MEAs help students learn complex problem-solving, mathematical modeling, and teaming skills. The activities are built on six core principles (Diefes-Dux et al., 2004): (1) model-construction principle, (2) reality principle, (3) self-assessment principle, (4) model-documentation principle, (5) generalizability principle, and (6) effective prototype principle. Collectively these inform the creation of the activity used for this study.

MEAs are embedded in real-world contexts and involve creating a model of the context in which the activity is embedded. MEAs require that teams of students document each step of the modeling process and develop models that can be assessed by the students themselves through criteria given in the problem. The end solutions should be generalizable to other situations and contexts. In addition to learning from the mathematical model, this structure allows instructors to create modeling activities for their classrooms. Such activities enable students to transfer information from other areas of their education, engage students through team-based activities, and push the curriculum to be more student-centered and active (Diefes-Dux et al., 2006; Hjalmarson et al., 2006; Moore et al., 2013, 2015). For these reasons, the MEA framework was used to structure the modeling activities for this study.

8.4 Methods

We used a specific case and the MEA framework to design the learning intervention (Diefes-Dux et al., 2004). Classroom artifacts were then analyzed using thematic analysis to identify computational thinking themes throughout the assignments.

8.4.1 Participants and Context

Participants (N=26) of this study were mainly in the final year of their undergraduate education within an Agricultural and Biological Engineering program. All students in the course were majoring in bioengineering or food process engineering. At the time of the research, the program reported having slightly more students that identified as female than males. This was the only section of the course and was required of all students, so the class was approximately representative of the department.

The course was the first of a two-part capstone class covering unit operations within the food and pharmaceutical industries, focusing on designing such systems using thermodynamics, heat and mass transfer, and reaction kinetics, which had all been covered previously in the program. The class met twice a week for a one-hour lecture and twice a week for a two-hour lab. The students had previous computer programming training in earlier engineering coursework, although the coursework and prior experience varied. Much of the prior experiences in computer programming came from introductory engineering coursework, an intro to computer science course, and previous projects from other classes within their major.

8.4.2 Learning Intervention

This study investigated the initial modeling and planning phases, where students were instructed to plan out a computational solution to a real-world engineering problem. It was implemented into the only section of the course. The learning intervention was a four-part modeling sequence that followed the structure of an MEA. It consisted of: (1) planning the model, (2) building the model, (3) evaluating the model, and (4) reflecting on the model. The results presented here are focused primarily on the first phase of the learning intervention, planning the model. The entire four-part modeling sequence, along with intervention documents, can be found in previous publications by the authors (Lyon et al., 2019; Lyon & Magana, 2021).

During the planning the model phase, students were tasked with modeling the temperature distribution inside a can during a sterilization process. Students were given the problem in the form of an email from a systems engineer that gave them specific properties of foods that were being processed and blueprints for the processing line they were modeling. This was done to simulate a realistic scenario, a vital tenet of the MEA framework (Diefes-Dux et al., 2006). Students were divided into teams of three or four to plan out how they would model the process. The questions posed to the students included:

1. What equations would they use?
2. What food properties were needed for their model?
3. What assumptions would they need to make?
4. What computational technique would they use?
5. How would they use all of these things together to construct a computational model?

Students were given a brief introduction to the problem and related concepts before the planning phase. Thus, they were prompted to recall details from previous coursework to try and create a potential modeling solution. The teams of three to four students were given the entire two-hour lab period to develop a potential plan for their computational model.

The problem was sufficiently complex that it required students to recall information from multiple previous courses, including heat transfer, thermodynamics, unit operations, mathematical modeling, and programming courses. The problem given to them was ill-defined, meaning it had multiple possible solutions. Features such as nonrelevant details and some relevant variables were provided in ranges. These types of details require students to understand what they are doing and require them to make difficult decisions and assumptions about how to use the given details. This type of ill-structured problem is useful for promoting multiple solutions and increasing the ability for future problem-solving (Kapur, 2008).

8.4.3 Data Collection

Two types of data were collected as part of this study: audio/video recordings (of four of the teams during the two-hour lab period) and planning templates (from each of the teams was collected at the end of the lab period). Appendix A shows the model-planning template that each team filled out. One researcher transcribed audio/video data of the group planning process to transform it into a format suitable for thematic analysis. Table 13 shows the data collection and the total participants from the study.

Table 13**.** Data overview for the current study

Data Source	**N**	**Description**
Audio/Video Files of In-Lab Discussions	4 groups (15 students total)	Four of the groups were randomly selected to be audio/video recorded during their entire planning process during lab time.
Planning Model Templates	15 (one for each group)	All groups turned in a planning model template, one of which is used in this analysis for each group in the study.

*four participants overlap the two data sources for a total of N=26 participants

8.4.4 Data Analysis

We used thematic analysis and followed the methodology as defined by Braun and Clarke (2006, p. 87). Their six-step process includes: (1) *familiarizing yourself with your data*, (2) *generating initial codes*, (3) *searching for themes*, (4) *reviewing themes*, (5) *defining and naming themes*, and (6) *producing the report*. We coded transcripts and course assignments (one assignment from each group for a total of 15) to develop a codebook of computational thinking themes that emerged from the model-planning process. First, each data point (transcripts and course assignments) was coded for significant computational thinking practices listed in Table 14 (Curzon et al., 2014; Lyon & Magana, 2021).

Table 14. Practices within computational thinking (CT) and definitions (Lyon & Magana, 2021).

CT Practice	Definition
Abstraction	Making problems easier to solve/think about by selectively removing/hiding unnecessary complexity by making assumptions about how the problem should operate or by neglecting or adding components to the problem.
Algorithmic Thinking	Setting up or talking about problems so that they can be solved in a stepwise manner by looking at the procedural steps/aspects of the solution; or by acknowledging the uses and effects of solution structures temporally and procedurally throughout the solution.
Evaluation	Comparing one's own solution against various criteria either given by the problem itself or against criteria brought to the problem by the student themselves. This can also be a statement describing desirable qualities of the final solution in terms of function, display, or other final form characteristics.
Generalization	Reusing previous solutions, methods, or constructs for the current problem; or the use of current solutions, methods, or constructs from the current problem to hypothesize about their use in future contexts.
Decomposition	Thinking about how the pieces of the problem can be used separately, strategically breaking down the problem for easier solution or use, or breaking the problem down into subcomponent structures, pieces, or areas.

After coding for these broad categories, we did the second coding round to create subthemes within each CT category. As a starting point, subthemes for each category as defined by Lyon and Magana (2021) were used, with the entire codebook used as a starting point shown in Appendix B. The frequency of each category and subtheme was tracked throughout the analysis to understand what CT categories were emerging and how often they showed up in student work. Depending on what emerged from the model-planning data, one researcher removed or added subthemes throughout the analysis process. This process resulted in a new codebook with some of the same subthemes (some removed because they did not emerge) and some new codes from the data.

8.4.5 Trustworthiness

Multiple approaches were taken to ensure trustworthiness. First, multiple types of data are used as a form of triangulation of the results, with classroom observations via audio/video recordings being used to understand the content of the student artifacts more deeply. Additionally, after the first researcher completed the coding, a second coder (a qualitative educational researcher with a background in the physical sciences) coded at least 20% of the statements previously coded by the first researcher for subthemes. This was done to make sure subtheme definitions were sound, and there was little overlap between the different identified subthemes in the data developed by the first researcher. After the coding round, the two researchers discussed differences and adjusted the codebook accordingly. This was done until the two researchers reached a greater than 80% agreement on the dually-coded statements.

8.5 Results

8.5.1 Overview of the Results

Many of the CT outcomes identified were similar or the same as the previous study (Lyon & Magana, 2021). However, some key differences to the codebook appear in bold text in Table 15.

Table 15. Finalized codebook of CT practices for model-planning

Abstraction	Algorithmic Thinking	Evaluation	Generalization	Decomposition
· Ranges to values · Multiple to single · Dynamic to static · Geometric relationships · Similar systems · Infinite to finite	· Indication of later use/effect · **Stepwise approach** · **Parallel methods** · Conditional logic	· Solution accuracy · Solution complexity · Time efficiency · Design criteria · Solution usability · Solution flexibility	· **Previous coursework or experience** · **External applications**	· Organization of larger solution method · **Allocation of resources**

***bold** denotes a newly identified CT outcome

As one can see from Table 15, all CT practices other than *evaluation* had at least one new identified CT outcome that was observed from the model-planning process. Additionally, some of the practices that were identified in the model-building process (Lyon & Magana, 2021) were not identified in the model planning data.

8.5.2 Abstraction

From previous work, *abstraction* was one of the most commonly identified CT practices in the model-building process (Lyon & Magana, 2021). Our results indicate that the model-planning process is no different, with *abstraction* being the most frequently identified CT practice within the model-planning data. Table 16 below shows the finalized definitions and representative quotes of each CT outcome identified from the model-planning data.

Table 16**.** Definitions and quotes of CT outcomes identified for *abstraction* (letters denote individual deidentified students).

Outcome	**Definition**	**Quote**
Ranges to values	Simplifying a range or list of values into a single value (e.g., worst-case scenario, average across a range, and picking the highest value) or a single function (assume that *x* follows function *y*).	“So z-value is the number of degrees that requires that variation. So I feel like we should choose the highest one again.”
Multiple to single	Simplifying aspects of the problem that have multiple dimensions or factors and simplifying the dimensions or factors considered in the solution through choice of one of neglect of factors; or creating limits or boundaries to what is considered to be affecting or influencing the considered system.	“A: The tin has, can we assume like no heat loss with the…I don’t know what I’m trying to think. Can we assume the can is negligible, like the metal part? D: Oh yeah, conduction through can wall is negligible because it’s so thick.”
Dynamic to static	Making factors or variables constant, uniform, or unchanging that would normally vary in respect to other variables or aspects of the problem.	“D: Density, like constant pressure means you have a density that’s not relying on pressure. It’s more relying on the temperature.”
Geometric relationships	Simplifying problems by assuming a geometric characteristic or relationships between variables, constants, or factors within the problem space.	“D: Yeah so we will have to assume.. uh C: Cylindrical. D: Cylindrical yeah. So we’ll get..”
Similar systems	Aligning properties of a current unknown system or property of a system with knowns of a system that is well known or familiar.	“B: What assumptions will you make to solve the problem? C: Assume, pumpkin pie filling is about like applesauce.“
Infinite to finite	Making aspects of the problem or system that are infinite or continuous and making them finite or discrete in nature.	“But uh depending on our can dimensions, we could do the whole like infinity, can split by infinity slabs, or just infinite cylinder.”

There were two new codes for *abstraction* found during the thematic analysis process. One is about aligning properties with similar systems. This makes sense that it would appear during the model-planning process as students are just seeing the problem for the first time and are thinking through ways to simplify the problem from systems that they have seen in problems previously. The second new CT outcome is geometric relationships. This is likely showing up in the model-planning data because students have to decide fairly early on what geometry they plan to assume to solve the problem. This decision is likely already made during the model-building process and not at the forefront of their minds. One item to note is the scarcity of the infinite to finite code during the model-planning process compared to the model-building process. This could be due to students not thinking about numerical methods or programming as much during the model-planning process, which is deeply important to the model-building process.

8.5.3 Algorithmic Thinking

Algorithmic thinking was seen throughout the dataset as a common CT practice, with multiple observed outcomes. Table 17 shows the different CT outcomes observed in the data from the practice of *algorithmic thinking*, with both definitions and representative quotes.

Table 17. Definitions and quotes of CT outcomes identified for *algorithmic thinking* (letters denote individual deidentified students).

Outcome	Definition	Quote
Indication of later use/effect	Indicating that information will be useful for a later process or a later point in the code. Indication of what certain variables or structures will cause later in the solution or code or explanation of location of code for effect.	"All I know is that we are going to need to determine like the alpha, so we need like the *k*'s and all of those things so that might be important then, like how much moisture there is, to determine the *k* value and like the c_p"
Using a stepwise approach	Giving things a logical order or listing steps to follow in order to solve the problem. This can include listing steps explicitly (e.g., 1, 2, 3…), diagramming, giving things logical order (i.e., we should do *x* first, *y* second, and then *z* third), or thought experiments (i.e., if we do *x*, then *y* will happen, and then *z* will likely follow).	"It's saying, yeah. It is trying to say, okay, I don't know my next value, but I know my temperatures right now throughout my area. Okay. Estimate, if I step one point in time, like one second, what will the temperature at that point I want to look at become. And then you take that value, you just guessed that your model and then plug it back into the equation for the next time around."
Considering parallel methods	Indication that there are parallel or redundant solution methods or information given.	D: So in this equation they use E_a, since uh temperature changes [inaudible] effect. And that's what *z* value does. A: Oh okay, so *z* already takes that into account? D: Yeah so *z* and *D* are intertwined like E_a and *D* are intertwined. […] Since we have both *D* and *z* we don't need E_a."
Conditional logic	Student making an if/else statement or conditional statement based on previous or future conditions (i.e., if I do *x*, *y* will happen)	"I don't, I don't remember them. But like for a tuna can you probably shouldn't assume it, because it's really flat and not much curvature. But for like a soup can you might get away with it. [laughs] Why'd you choose it? Engineers instinct."

As shown in Table 17, there are two new subthemes identified. The first, named *stepwise approach*, is where students outline a solution process step by step, diagramming, or through thought experiments. Compared to the model-building phase, this likely showed up because students needed to plan out their solutions, which often included putting together a step-by-step plan of how they intended to solve the problem. Figure 10 below shows a diagram submitted as part of a student report that shows this stepwise approach code.

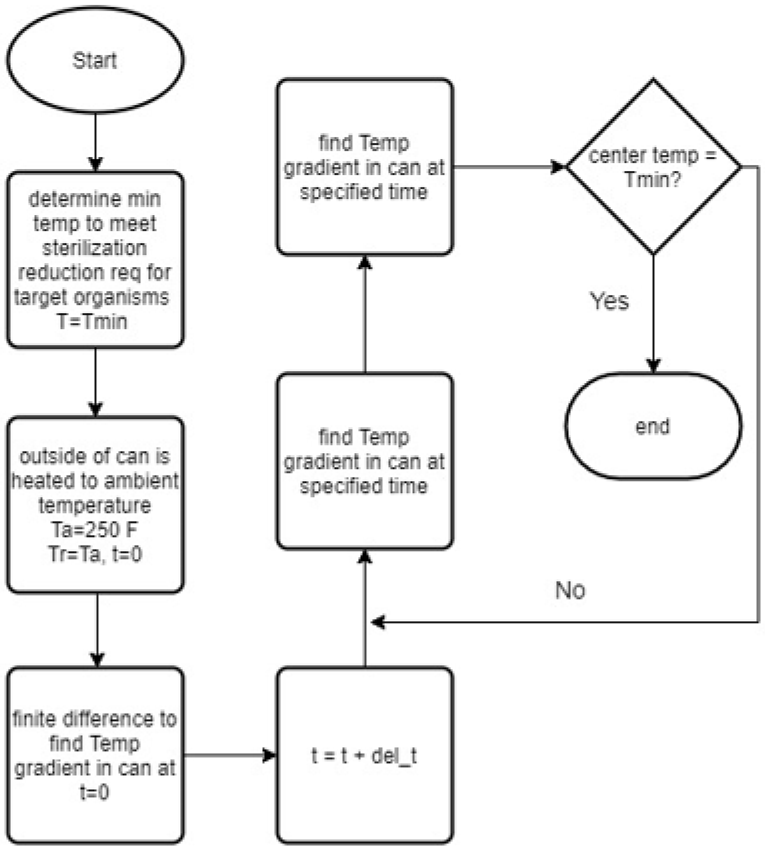
Start
determine min temp to meet sterilization reduction req for target organisms T=Tmin
outside of can is heated to ambient temperature Ta=250 F Tr=Ta, t=0
finite difference to find Temp gradient in can at t=0
t = t + del_t
find Temp gradient in can at specified time
find Temp gradient in can at specified time
center temp = Tmin?
Yes
No
end

Figure 10. Diagram coded as stepwise approach as a type of *algorithmic thinking*.

Additionally, *considering parallel methods* was a new code, where students often pointed out overlapping or duplicative information that showed multiple or parallel ways of solving the same problem. The majority of the CT outcomes observed in the practice of *algorithmic thinking* were an *indication of later use/effect*. This is likely because students needed to project forward what they needed to solve the problem and why they needed it by indicating how they planned to use the information or how it affected their algorithm.

8.5.4 Evaluation

Evaluation was the one category where no new outcomes emerged from the coding process; however, nearly all outcomes from the previous codebook were observed. Table 18 outlines the definitions and quotes associated with each observed outcome.

Table 18. Definitions and quotes of CT outcomes identified for *evaluation* (letters denote individual deidentified students).

Outcome	Definition	Quote
Solution accuracy	Evaluates the solution or the code in regards to the actual or perceived accuracy of the solution method decisions.	"The exact properties of Nacho Cheese may not be available. I think it will limit accuracy."
Solution complexity	Evaluates the solution or code in regards to the amount of work, time, or effort needed to create the solution.	"What are the benefits of this technique? Uh, it's simplistic, it's not insane."
Time efficiency	Evaluates the solution method based on the amount of time the code or computer takes or wastes to arrive at the final solution.	"I chose this method because it is computationally fast."
Design Criteria	Evaluates the code or solution against the actual or perceived wishes and requirement of the stakeholders or problem statement.	"This way if the moisture is content is lower, we will still be within the desired amount of sterilization."
Solution usability	Evaluates the solution or code in regards to ease of use for themselves or for others.	"…allow us to create a usable solution."
Solution flexibility	Evaluates the solution or code based on the ability to easily change or pursue future changes.	"This method gives us a lot of control over the parameters and the 'machinery' of the program."

One of the most significant results from the analysis is that *debugging* is absent as an outcome in the model-planning process. This is likely due to *debugging* being closely tied to programming the solution, which is more central during the model-building process. Additionally, many of the coded statements focused on the mathematical method for the solution rather than the computational structuring of the solution. This demonstrates that students were much more in the mindset of solving the problem mathematically during the model-planning process. They were often evaluating their solution in a more mathematical sense.

8.5.5 Generalization

The CT practice of *generalization* was much more prevalent during the model-planning data than during the model-building data. Two new outcomes were identified for *generalization* and one missing from the original codebook. Table 19 outlines the definitions of the identified outcomes and representative quotes.

Table 19. Definitions and quotes of CT outcomes identified for *generalization* (letters denote individual deidentified students).

Outcome	Definition	Quotes
Drawing from previous experiences	Use of previous coursework or personal experiences to inform the current solution method. This can include previous coursework, professional experiences, or explicitly related concepts from previous courses.	"C: Oh I remember this equation from [another course], the *d* equals d_0, times… B: Oh yeah C: or [another course] I don't know, *d* equals d_0 times ten to the t_0 minus *t* over z."
Projecting to other or future applications	Reference to other problem spaces, external to the current problem, where the current solution would be useful or not useful based on that nature of the current problem or solution.	"So the model will not account for really hot or really cold days."

This was the only CT practice that had completely new themes from the original codebook. Whereas the original codebook had *reuse of code* as an identified outcome, during the model-planning activity, two new outcomes emerged. The first, *drawing from previous experiences*, naturally connects to any planning activity. Students were often making explicit references to previous coursework, assignments, or work experiences and how they were helpful in the current problem. The other new outcome, *projecting to other or future applications*, had students looking at how different decisions they made during the planning phase would impact the future applications that their model could have.

8.5.6 Decomposition

The CT practice of *decomposition* was found in the dataset at a relatively low frequency. However, two different outcomes were identified through the model-planning process. Table 20 shows the different outcomes, definitions, and quotes identified.

Table 20. Definitions and quotes of CT outcomes identified for *decomposition* (letters denote individual deidentified students).

Outcome	Definition	Quote
Organization of larger solution method	Decomposing the code or solution method as an organizational tool.	*See Figure 2
Allocating resources	Breaking up a problem amongst group members in order to reduce complexity of the work required or to increase quality of the solution method	"Do we wanna split up this in any way? Is that allowed? What we're doing. Um. So like we'll have to individually look up a bunch of equations and the limitations and benefits of the computer systems we decide to use. Or like the solving method we decide to use."

One newly identified outcome was *Allocating resources*, where students look at decomposing the problem solution by breaking up work or doing different parts of the planning process synchronously. Another practice was where students created diagrams as an organizational tool during the planning process. Figure 11 shows an example of this coded behavior.

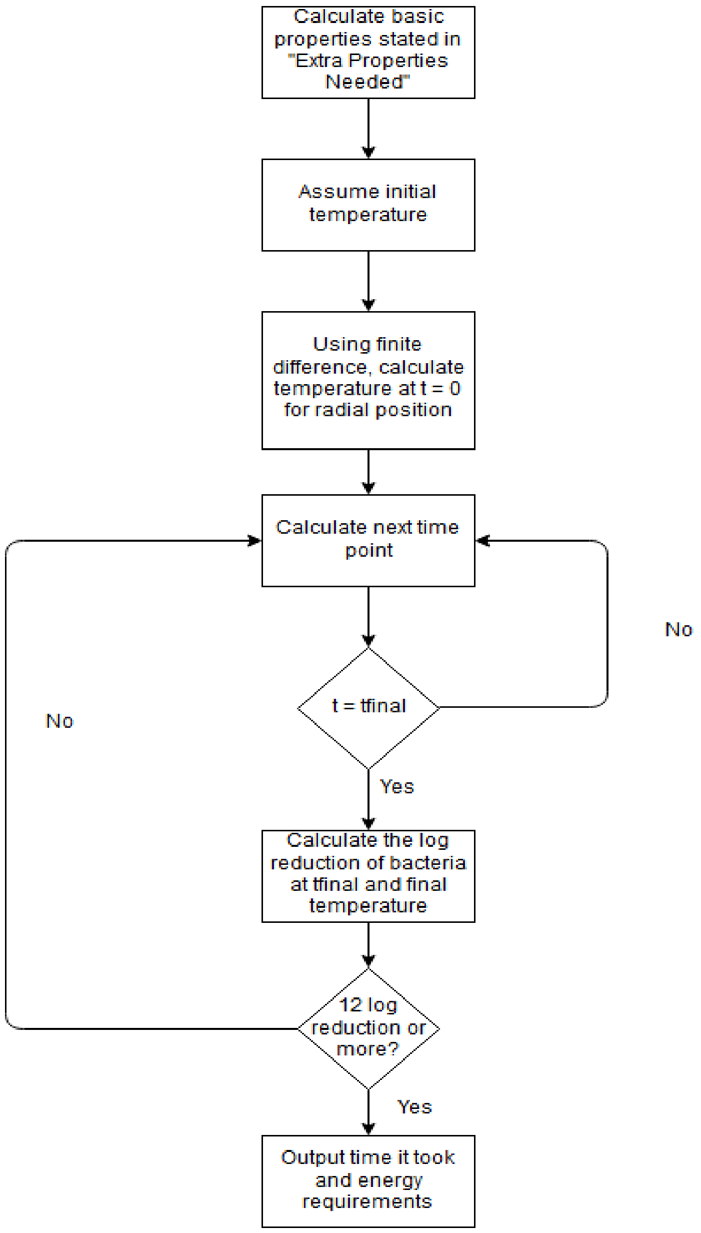
Calculate basic properties stated in "Extra Properties Needed"
Assume initial temperature
Using finite difference, calculate temperature at t = 0 for radial position
Calculate next time point
t = tfinal
No
Yes
Calculate the log reduction of bacteria at tfinal and final temperature
12 log reduction or more?
No
Yes
Output time it took and energy requirements

Figure 11. Example diagram demonstrating *organization of larger solution method.*

Diagrams such as Figures 10 and 11 demonstrate two processes that were individually coded. First, the problem space is broken into multiple pieces, a *decomposition* practice coded as the *organization of a larger solution method*. Second, an algorithmic practice of putting these pieces into order was coded as *a stepwise approach*.

8.6 Discussion

Results of this study continue to provide evidence that modeling activities are effective methods for integrating CT into the undergraduate engineering classroom. While the connection between CT and modeling has been made previously in the literature (Weintrop et al., 2016), the results of this study further this connection by showing how CT is specifically practiced by students going through the planning process of a modeling cycle. Our results show that all five types of CT practices (i.e., abstraction, algorithmic thinking, evaluation, generalization, and decomposition) were found in students' model-planning solutions. This is understandable because modeling and CT are two processes that have often been tied together in the literature (Magana & Silva Coutinho, 2017; Sengupta et al., 2013; Weintrop et al., 2016). Our results further develop the nature of this relationship by showing that CT is used within the planning stage of the modeling process and that unique and different outcomes seem to emerge as students plan their computational solutions.

One consistent theme through much of the results was the lack of emphasis on the programming aspects of the assignment. During the building phase of the modeling process, students often focus on the code or the computational elements of the modeling problem (Lyon & Magana, 2021). Many of the more computationally focused themes from the original codebook, such as *debugging* and *code reuse*, were not observed during the planning phase. However, even with the change in emphasis, many of the same outcomes appeared in the planning phase, as observed in the modeling process's building phase. This lack of focus on the computational aspects of the problem may be attributed to many different factors, such as students' lack of familiarity with programming compared to mathematics (Magana & Silva Coutinho, 2017) or their lack of self-efficacy with making programming or computing decisions (Busch, 1995; Hutchison et al., 2006). In either case, students' planning processes lacked emphasis on the computational elements of their modeling solutions.

Students' plans seemed to focus on many of the realistic or mathematical elements of the problem, likely as they attempted to abstract the information from the problem statement into a mathematical form. This aligns with the modeling literature, which describes the first step of the modeling process as one where students are collecting as much relevant information from the real world as it relates to their solution (Louca & Zacharia, 2012; Lyon & Magana, 2020a; Magana et al., 2020). The planning phase is where students are "breaking down the phenomenon under study into small pieces that can be incorporated into an external model" (Louca & Zacharia, 2019, p. 241). Our results align with this in that students break the problem statement down from a real-world context into smaller pieces that they can make sense of and understand how they could put together into a mathematical context.

Another significant finding of this work is the prevalence of *generalization* in the planning phase of the modeling process. We found two new outcomes for *generalization*, one backward-looking and one forward-looking. The first new *generalization* theme, *drawing from previous experiences*, is an outcome where students look at ideas, concepts, or methods that they have learned in the past that can be used for the presented problem. The second new *generalization* theme, *projecting to other or future applications,* is a primarily forward-looking outcome. Students think about how decisions about their model may make their model useful or limited in those contexts. It is an expected outcome because our previous experiences are foundational to education and learning (Dewey, 1938). Yet, the outcome seems to go one step further: students are transferring information not into the same context but to new contexts.

This is an encouraging finding that students are not only thinking externally about their presented problem but also thinking of ways to transfer the information to new contexts. These backward- and forward-looking *generalization* practices are likely emerging as students try to fit the current problem into their mental model of the phenomenon. The process of modeling is where students externalize their mental model or their detailed understanding of a phenomenon and how it works into a physical, real-world mathematical, or computational model (Jonassen, 2009; Nersessian, 2007). It appears that during this planning phase of the modeling process, students are engaging with their understandings of the phenomenon and generalizing what information will be useful to the current problem and how that may make the model useful or limit its use in the future. These practices are desirable as students develop their adaptive expertise in computation (Magana et al., 2019).

Both new *generalization* themes seem to be about transferring information from previous situations and future situations indicative of expert-like practice (Bohle Carbonell et al., 2014; Hmelo-Silver & Pfeffer, 2004). The literature around adaptive expertise has shown that transferring information into new and novel situations is one way experts are differentiated from novices (Delany, 2008; Hatano & Inagaki, 1984; McKenna, 2007). Our learning design of giving students very little instruction before planning their models may have encouraged them to try to adapt and innovate on their prior experiences to make sense of the problem.

8.6.1 Implications for Teaching and Learning

This study provides two main implications for instructors looking to implement modeling activities in their classrooms. The first is unique CT outcomes that students can practice by planning their modeling solutions. While the benefits of metacognitive practices in modeling are already well documented (Jaiswal et al., 2021; Magana et al., 2019; Vieira et al., 2019), our study furthers these implications by showing that building CT practices is an additional benefit to having planning as part of the modeling process. One issue with computational modeling problems is that some students seem to start coding without necessarily planning out their solutions, which indicates novice problem-solvers (Magana et al., 2019). Suppose instructors give specific time for students to plan out their solutions. In that case, it may help students move toward expert-like practice and away from the novice-like practice of immediately solving the problem.

Furthermore, instructors should tailor modeling activities they are developing to amplify outcomes that were observed in our results. That means activity designers should incorporate opportunities for students to explore their previous experiences, consider future model applications, or integrate flow diagrams to incorporate *algorithmic thinking* or *decomposition*. By tailoring activities in this way, instructors can amplify naturally occurring outcomes during the modeling process. One opportunity to do so would be to engage in design-based research, similar to this study, which allows educators to understand the connection between the pedagogical design decisions and student outcomes (Barab & Squire, 2004; DBRC, 2003)

8.7 Conclusions

There are some limitations to our study. First, the activity was situated within the specific context of food process engineering. Second, demographic information on the participants was not collected as part of this study. These two factors may limit the study’s generalizability to other contexts and should be considered in future research applications. Additionally, the work is limited by what the students talked about or were aware of as they solved the modeling problem. There are likely other computational thinking outcomes that students are doing that they didn’t verbalize or were not aware they were doing.

Our future work will consider additional modeling cycle steps and how CT is elicited in those settings. This study is part of a larger effort to understand CT within the modeling and simulation process. Future efforts will continue to look at other steps of the modeling cycle. Additionally, understanding how different learning designs for modeling and different disciplinary context will be crucial in understanding the entire scope of CT within modeling activities.

Computational thinking is becoming a critical skill for learners across disciplines. Modeling is an engineering activity that naturally lends itself to students practicing all aspects of CT. We studied students planning their modeling solutions, and we identified various themes of CT outcomes that emerged while going through the planning process. Our results indicate that the planning process has distinctly different CT outcomes from the model-building process and amplifies various aspects of CT. Practices such as generalization were more prominent in the planning phase as students demonstrated specific outcomes indicative of expert-like practice. Because of this, instructors should prioritize incorporating time for students to plan their solutions when incorporating modeling activities into their classroom if their goal is to give students ample practice in all aspects of CT.

8.8 Acknowledgements

This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under Grant Number DGE-1842166 as well as the National Science Foundation under Grant Number EEC-1449238. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

9. CHARACTERIZING COMPUTATIONAL THINKING THROUGH MODEL EVALUATION AND REFLECTION

9.1 Abstract

Evaluation and reflection are critical skills to the modeling and simulation process. It involves (a) looking at models students have built, (b) understanding how useful they are, (c) identifying how they could be improved, and (d) reflecting upon what things the student did during the model-building process. As such, these skills allow students to improve their current work and future working endeavors. At the same time, computational thinking is an essential skill of increasing interest and significance across STEM disciplines. There is a need for studies that show how to incorporate and build computational thinking into various disciplinary curricula. This study meets this need by identifying how computational thinking is used during the evaluation and reflection process of the modeling and simulation cycle.

The study participants were final-year engineering students enrolled in a capstone course. The research used the model-eliciting activity framework to build a modeling activity that included an evaluation and reflection phase. Thematic analysis was then used to identify different computational thinking practices and outcomes that were elicited during the evaluation and reflection phase of the assignment. Our results indicate that model evaluation and reflection are appropriate curricular activities to elicit complex and unique computational thinking outcomes.

9.2 Introduction

Modeling and simulation are critical skills to an engineer's education (Gainsburg, 2006; Lyon & Magana, 2020a; Magana, 2017). They allow engineers to make predictions about the world around them and are useful across engineering disciplines. This is apparent in the research on identifying how to best teach modeling and simulation activities in the engineering classroom, especially at the undergraduate level (Diefes-Dux et al., 2004; Magana et al., 2013; McKenna & Carberry, 2012). This has led to many different teaching modeling and simulation frameworks across STEM disciplines (Diefes-Dux et al., 2004; Louca & Zacharia, 2012; Shiflet & Shiflet, 2014). However, one thing remains certain: modeling and simulation are valuable and necessary skills for engineering students as tools for academic purposes and industry applications (Magana & Silva Coutinho, 2017).

Yet, modeling and simulation are skills built on many other underlying pieces of knowledge, such as disciplinary knowledge and programming knowledge (Vieira et al., 2017). Because of this, educators must make sure that students have the appropriate practice of both these sets of skills before embarking on a modeling and simulation problem. While the disciplinary content is usually well covered in the engineering curriculum, this is often not the case for programming content. One issue for educators is that programming education is often not as integrated into the engineering curriculum due to space issues and lack of comfort from engineering professors in teaching computer programming (Magana & Silva Coutinho, 2017). Because of this, it is often through modeling and simulation activities that students can practice programming skills they otherwise might not (Carey & Gougis, 2017; Lyon & Magana, 2020b; Magana et al., 2017; Yan et al., 2015).

This programming knowledge is closely related to a set of skills commonly known in the literature as computational thinking. Computational thinking is a set of skills that are standard practices within computer science but that are applicable and useful to a wide range of disciplines (Wing, 2006). This skillset includes practices such as abstraction, algorithmic thinking, evaluation, generalization, and problem decomposition (Curzon et al., 2014; Grover & Pea, 2013; Lyon & Magana, 2021; Selby & Woollard, 2013). Together these practices allow students to solve complex problems from various disciplines. Yet, the research on these practices is often more definitional and less practical (Grover & Pea, 2013; Kalelioğlu et al., 2016). Our study addresses this by showing how computational thinking practices are explicitly used in the modeling and simulation, specifically within the model-evaluation and reflection process.

This study aims to answer one primary research question:

How are computational thinking practices elicited through the evaluation and reflection process of computational modeling?

This question will be answered primarily through thematic analysis performed on student artifacts generated while solving a modeling and simulation challenge in a capstone engineering course. It builds on previous studies that have looked at the model-planning and model-building processes (Lyon et al., n.d., 2019; Lyon & Magana, 2021).

9.3 Background

Multiple bodies of literature are used to build the framework for this study, including computational thinking, evaluative practices in modeling, and reflective practices in modeling.

9.3.1 Computational thinking

Computational thinking has been of increasing interest in the research literature to educators who seek to build these practices within their students (Kalelioğlu et al., 2016; Lyon & Magana, 2020b). What is included in the definition of computational thinking has mainly been a topic of debate and has differed from study to study; however, some prominent practices are identified as part of computational thinking, two of the most notable being abstraction and algorithmic thinking (Kalelioğlu et al., 2016). Other practices identified as part of computational thinking range from parallelization (V. Barr & Stephenson, 2011) to collaboration (CollegeBoard, 2013). However, a few practices seem to encapsulate many of the literature's ideas as part of computational thinking. One straightforward definition that appears to incorporate many of these ideas is the one put forth by Selby & Woollard (2013) and Curzon et al. (2014), which includes primarily five elements: abstraction, algorithmic thinking, generalization, evaluation, and decomposition. This study and authors use and adapt this definition for computational thinking when investigating modeling activities (Lyon et al., n.d., 2019; Lyon & Magana, 2021).

9.3.2 Evaluation in Modeling

The need to evaluate what we have produced is a natural response. In the case of modeling, many in the literature have included evaluation or assessment of the created model as a critical part of the process (Diefes-Dux et al., 2004; Louca & Zacharia, 2012; Magana, 2017). Two terms often included in this evaluation process of models are validation and verification (Shiflet & Shiflet, 2014). Shiflet and Shiflet (2014) define these processes when they write, "verification determines if the solution works correctly, while the process of validation establishes if the system satisfies the problem's requirements" (p. 9). Thus broadly, evaluation looks to ensure multiple things, such as how effective the model is at answering its original purpose and whether or not the mechanics of the model are working as designed by the modeler.

While evaluating a model is a broad term, it is undoubtedly a necessary component to any problem-solving activity to understand how the solution fulfills different criteria brought to the problem standards by which to measure our solution. Often solution evaluation is in the hands of the instructional team through traditional grading processes. Yet, the literature has indicated that allowing students to evaluate their work and progress may improve their overall learning, future self-efficacy on similar tasks, and motivation to learn (Schunk, 2003). And even when these self-evaluations are negative, they may promote positive behaviors such as seeking help, reselection of solution strategies, or modification to self-regulation learning strategies (Schunk & Ertmer, 2000). Thus, regardless of the perceived result of the student's evaluation of their work, there are potentially positive student outcomes.

9.3.3 Reflective Practices in Modeling

In addition to evaluating a modeling solution, another beneficial practice is to reflect on the modeling process that has been undergone. Reflective practices and general metacognitive practices are beneficial to student learning and growth towards more expert-like behaviors (Ertmer & Newby, 2015; Lew & Schmidt, 2011). Specifically, in modeling activities, reflection can help students move towards more beneficial learning practices and may also impact assignment performance (Jaiswal et al., 2021). Because of this, reflection is an essential step to any modeling process, but more broadly, any learning process.

There are multiple different types of reflection, with Ertmer and Newby (2015) breaking it up into cognitive, motivational, and environmental strategy reflection. These categories look at how students are doing or thinking about the task, how the topic relates to personal motivation, and the optimal conditions that a student can put themselves in to complete the task. By thinking about all three of these categories, students can move towards more expert-like practices that allow them to succeed in future learning endeavors.

9.4 Methods

This section outlines the methods used in this study including the context, learning design, and data analysis for the study.

9.4.1 Context and participants

The modeling intervention was deployed in a senior-level engineering capstone course. The discipline of the course was biological engineering. The primary focuses and interests of the students were in the pharmaceutical and food industries. The specific learning unit the modeling intervention overviewed was over sterilization and microbial death kinetics, where the students were learning about canning and retort operations in the food industry. The course covered other topics as well, such as fermentation, drying, and separation processes.

The students in the course were mainly in the final year of their undergraduate studies. They had previously taken multiple general and disciplinary courses covering related topics such as computational or numerical modeling, heat and mass transfer, and a general programming course. They had the tools to do much of the modeling activity in this study before the course but just needed to put multiple bodies of knowledge together. The class was reflective of the major given that it is a smaller department at the university, with the department having slightly more females than males at the time of the data collection.

9.4.2 Design of the intervention

The intervention was based on the design principles of the model-eliciting activity framework (Diefes-Dux et al., 2004). This included giving students realistic context, opportunities to build and assess a mathematical modeling solution, and having them document the process at each step. The students were broken up into teams and given a realistic scenario in the form of an email to a consulting firm they were working for, which read (Lyon & Magana, 2021):

Hey FOODSCorp team,

We know we have done work with you before and are confident we will get great results again. Last night one of our heaters before the filler went out for our canning sterilization line. The manufacturer of the filler is unfortunately in Germany and this is a specialty part, so we don't expect to get the new heating element for a couple of months. In addition, we reached out to general machining, however, they too cannot generate the new heating element. The line is work about 10k an hour and runs four different products, so anything we can do to get it running again the company is willing to try.

Our hope is within the next two weeks to get the line back up and running by adjusting processing parameters, however, we do not have the capabilities in house to model the process using MATLAB software. The heating element was originally able to heat the food material prior to canning to 200 ^{o}F however, the replacement part we found can only achieve a filling temperature of 180 ^{o}F currently. After the food is canned, it is heated to commercial sterilization and then it is cooled with water. Our micro team has asked per company policy that we achieve a 12-15 log reduction in microbial load prior to production. Our quality team has asked that we maximize our Vitamin B1 and Vitamin C intact in all products. […] Please deliver us an appropriate computational model via MATLAB software that is capable of outputting visuals of the sterilization process for all four food products showing temperature at various points along the radius as a function of time in addition to plots describing micro load and nutrition degradation.

Thanks!

Jenn

They were asked to plan their model as a group and create it individually. We have reported the computational thinking benefits to these portions of the previous activity in the literature (Lyon et al., n.d.; Lyon & Magana, 2021). However, the third portion of the activity asked students to evaluate as a group and reflect individually on the model they had created.

For the evaluation activity, students were paired with students from other planning groups to expose them to ideas they would not have had access to initially in the activity. They were then given approximately twenty minutes to answer and take notes on the following questions:

1. What are different assumptions that you made about the physical properties of the system?
2. Did you use different data?
3. How would these differences impact the model?
4. How did your programming strategies differ?
5. What advantages do you see in how they did their model?
6. What advantages do you see in your own?

When the students had gone twenty minutes with their first pairing, they then did the same thing with another pair of students from other planning groups, giving them even more exposure to different ways of solving the modeling problem.

After the students had done the planning activity, students were asked to individually complete a short reflection report that asked them the following questions:

1. What approaches did other students take with respect to the data that they used (justifications, assumptions, and limitations) and the way they programmed their model? Be as detailed as possible in listing various differences between models. For each difference talk about WHY you think the other group chose to do it the way they did. Be detailed.
2. How did these differ from your own approach? When would your own approach make the most sense? When would different assumptions that other groups made make the most sense?
3. If you were to do this assignment again what are different assumptions you would make and what do you believe to be the optimal solution to the problem?
4. What was the most challenging piece of this assignment? Why do you feel that was the most challenging? How did you overcome this challenge?

These activities forced students to look external to their solution, causing them to reckon that there are multiple ways to solve a problem, which can be a valuable learning experience for students (Kapur, 2008). Students did more than create a modeling solution to the given situation; they had to justify their solution pathway and understand the benefits and drawbacks of different ways of thinking about the given problem.

9.4.3 Data Collection and Analysis

The data used and collected for this study were the student artifacts created while going through the evaluating and reflecting process after the model-building process. It involved data in both audio and written forms and is overviewed in Table 21 below. Students were originally in sixteen teams for the model-planning process, and thus data were used from one individual from each of the teams. Additionally, only four of the sixteen teams were audio/video recorded during the model-evaluation meetings due to research limitations.

Table 21. Overview of data used for this study.

Data Type	Description	Group/Individual	Sample Size
Audio/Video recording of in-class evaluating process	Recordings of students meeting in groups to discuss the different ways that they solved the modeling problem.	Group	4 (4 teams of 4 students each)
Evaluating the model report	Written document of student notes written down individually while meeting with other peers to compare and evaluate modeling solutions.	Individual	16 (one from each group of students)
Reflecting on the model report	Written document of student reflections about the modeling process and their own solution, discussing difficulties, future improvement, and benefits to their modeling process.	Individual	16 (one from each group of students)

Each artifact (written reports and transcribed audio recordings) underwent thematic analysis. Our thematic analysis process follows the six-step process as outlined by Braun and Clarke (2006) as (1) familiarizing yourself with the data, (2) generating initial codes, (3) searching for themes, (4) reviewing themes, (5) defining and naming themes, (6) producing the report. Our deductive categories for coding the data were the computational thinking practices adapted from Curzon et al. (2014) and previously used in our work (Lyon et al., n.d., 2019; Lyon & Magana, 2021). These categories are overviewed in Table 22.

Table 22. Definitions of computational thinking (CT) practices (Curzon et al., 2014; Lyon & Magana, 2021).

CT Practices	Definition
Abstraction	Making problems easier to solve/think about by selectively removing/hiding unnecessary complexity by making assumptions about how the problem should operate or by neglecting or adding components to the problem.
Algorithmic Thinking	Setting up or talking about problems so that they can be solved in a stepwise manner by looking at the procedural steps/aspects of the solution; or by acknowledging the uses and effects of solution structures temporally and procedurally throughout the solution.
Evaluation	Comparing one's own solution against various criteria either given by the problem itself or against criteria brought to the problem by the student themselves. This can also be a statement describing desirable qualities of the final solution in terms of function, display, or other final form characteristics.
Generalization	Reusing previous solutions, methods, or constructs for the current problem; or the use of current solutions, methods, or constructs from the current problem to hypothesize about their use in future contexts.
Decomposition	Thinking about how the pieces of the problem can be used separately, strategically breaking down the problem for easier solution or use, or breaking the problem down into subcomponent structures, pieces, or areas.

After coding the data for the computational thinking practices, the results were searched for subthemes within each practice, which we have called outcomes. Our previous research found many CT outcomes associated with the model-building and planning processes (Lyon et al., n.d.; Lyon & Magana, 2021). In this study, those outcomes were used as a starting point for understanding the similarities and differences between the evaluating/reflecting stages of the modeling process and the building and planning phases. Thus, initial coding categories and definitions of the initial coding categories have previously been used in the literature (Lyon et al., n.d.; Lyon & Magana, 2021)

9.4.4 Trustworthiness Considerations

To ensure the trustworthiness of the results, this study uses data transparency by providing detailed descriptions of the results (Onwuegbuzie & Leech, 2007). In addition to this, the initial codebook has been tested through inter-rater reliability in previous studies by the same author and primary coder (Lyon et al., n.d.; Lyon & Magana, 2021). Finally, triangulation of the results was achieved through the prior study of the codebook and multiple data sources in this study.

9.5 Results

The thematic analysis results revealed that many of the same outcomes were found in model evaluation and reflection as were found in both the model planning and building phases. Table 23 below shows the complete codebook after coding for both the deductive categories and the outcomes. This is an updated version of the table from previous studies to include the new results (Lyon & Magana, 2021).

Table 23. Finalized codebook of CT practices for model evaluation and reflection

Abstraction	**Algorithmic Thinking**	**Evaluation**	**Generalization**	**Decomposition**
· Ranges to values · Multiple to single · Dynamic to static · Geometric relationships · Similar systems · Infinite to finite	· Indication of later use/effect · Stepwise approach · Parallel methods · Conditional logic	· Solution accuracy · Solution complexity · Time efficiency · Design criteria · Solution usability · Solution flexibility · **Ideal Solution** · **Peer Comparison** · Debugging	· Previous coursework or experience · External applications · **Similar problems** · Reuse of code/equations ·	· Organization of larger solution method · Allocation of resources · Ease of use of program

***bold** denotes a newly identified CT outcome

The analysis resulted in three new outcomes that emerged through the model evaluation and reflection process. Two were in the category of evaluation and were students evaluating in comparison directly to their peers and a perceived ideal-world solution. Additionally, an outcome emerged in the generalization category where students talked about how they generalized parts of their solution from similar problems concurrent to the modeling problem.

9.5.1 Abstraction

The category of abstraction, while still a major coded category throughout the model evaluation and reflection phase, the practice of abstraction was a bit less prominent than in previous stages of the modeling activity, such as planning and building the model. As such, no new outcomes emerged from the coding process. However, the evaluation and reflection data observed all the previously seen outcomes. Table 24 below overviews each of the outcomes, the definitions of the outcomes, and an example of each outcome in the evaluation and planning of models.

Table 24. Observed abstraction outcomes during model evaluation and reflection.

Outcome	Definition	Quote
Ranges to values	Simplifying a range or list of values into a single value (e.g., worst-case scenario, average across a range, and picking the highest value) or a single function (assume that *x* follows function *y*).	"I averaged the retort temperature and the filling temperature and used that value of temperature for [equation]."
Multiple to single	Simplifying aspects of the problem that have multiple dimensions or factors and simplifying the dimensions or factors considered in the solution through choice of one of neglect of factors; or creating limits or boundaries to what is considered to be affecting or influencing the considered system.	"C: So I think one of the assumptions that I made was I didn't really even consider the metal can as a layer or boundary layer really. B: Yeah. C: Um. Just because heat would just transfer so fast so I just thought it was like negligible."
Dynamic to static	Making factors or variables constant, uniform, or unchanging that would normally vary in respect to other variables or aspects of the problem.	"I said thermal properties stayed constant even though temperatures changed."
Geometric relationships	Simplifying problems by assuming a geometric characteristic or relationships between variables, constants, or factors within the problem space.	"I assumed additional energy cost was proportional, directly proportional to the percent increase in production time."
Similar systems	Aligning properties of a current unknown system or property of a system with knowns of a system that is well known or familiar.	"I did nacho cheese, and there was a paper published on thermal properties of cheese and there was a liquid cheese in there. So I was like, golden."
Infinite to finite	Making aspects of the problem or system that are infinite or continuous and making them finite or discrete in nature.	"Yeah I just assumed there was like in the retort there's a dump it into a bath of water. It just like opened up and then… C: Cans just… A: Yeah the cans just fall out into a bath of water."

Students did not appear to have any new abstraction behaviors during the evaluation and reflection process but used all of the previously identified outcomes. Much of the abstraction came through students comparing how they simplified their problems to each other or explaining the simplifications they made while planning and building their models. Often, abstraction would eventually lead to evaluative behaviors as students mentioned how they simplified their problem and what it ultimately led to in their model's performance.

9.5.2 Algorithmic Thinking

Algorithmic thinking was another CT practice with no new identified outcomes throughout the model evaluation and reflection stages. However, the thematic analysis found all of the previously identified outcomes in the data set. Table 25 overviews all of the outcomes associated with algorithmic thinking and gives example quotes for each from the students.

Table 25. Observed algorithmic thinking outcomes during model evaluation and reflection.

Outcome	Definition	Quote
Indication of later use/effect	Indicating that information will be useful or was used for a later process or point in the code. Indication of what certain variables or structures will cause or did cause later in the solution or code or explanation of location of code for effect.	"I used it to find my delta r and my number of nodes."
Using a stepwise approach	Giving things a logical order or listing steps to follow in order to solve the problem. This can include listing steps explicitly (e.g., 1, 2, 3…), diagramming, giving things logical order (i.e., we should do *x* first, *y* second, and then *z* third), or thought experiments (i.e., if we do *x*, then *y* will happen, and then *z* will likely follow).	"I had like Ti, like initial time being the, the initial time for all of the material being the filling temperature. And then after that, so like at the surface my initial temperature is the filling temperature and then after that it's the same temperature as the steam. Um. That's how I did it."
Considering parallel methods	Indication that there are parallel, multiple, or redundant solution methods or information given.	"The way we solved our models was very different also, with some people using either for or while loops (or a mixture of both), while others preferred to use predominantly functions."
Conditional logic	Student making an if/else statement or conditional statement based on previous or future conditions (i.e., if I do *x*, *y* will happen)	"While loops should be used when the loop is conditional and For loops when the duration of the loop is known."

One algorithmic thinking outcome that was particularly prominent in this dataset was *considering parallel methods*. This is because as students evaluated their solutions against the solutions of others, there were many times were they encountered redundant or parallel ways of solving the same problem. This led to more of this outcome during the reflection as students discussed what they might do differently next time by considering other ways the problem could be ultimately solved. Another minor difference is that the outcome of *indication of later use/effect* was often more of an explanation of how the code worked. In contrast, this outcome was more propositional in the planning of the model phases. Earlier in the process, students spent more time conjecturing what they would use information for later in the code. The students merely explained how the code worked operationally in the evaluation and reflection stages.

9.5.3 Evaluation

As one would expect, evaluation was the most prominent outcome and the one that was coded the highest number of times throughout the analysis. Additionally, it generated two new codes from the previous two rounds of research, including comparing to an ideal solution and peer comparison.

Table 26. Observed evaluation outcomes during model evaluation and reflection.

Outcome	Definition	Quote
Solution accuracy	Evaluates the solution or the code in regards to the actual or perceived accuracy of the solution method decisions.	"This was done because it is the most accurate way known to calculate these properties."
Solution complexity	Evaluates the solution or code in regards to the amount of work, time, or effort needed to create the solution.	"It would have been faster in the long run to use functions, but initially their code was more difficult to set up."
Time efficiency	Evaluates the solution method based on the amount of time the code or computer takes or wastes to arrive at the final solution.	"So, I'm gonna guess that the process would run a lot quicker if I had actually done it that way with like 1000 nodes or something and with a small time step."
Design Criteria	Evaluates the code or solution against the actual or perceived wishes and requirement of the stakeholders or problem statement.	"Yeah. We want to make ourselves look good to this company. You think we want them to think that we can't conserve their vitamins?"
Solution usability	Evaluates the solution or code in regards to ease of use (i.e. units, output graphs, etc) for themselves or for others.	"I think that this was a good way of portraying the data because it allowed you show both the heating and cooling in the same graph."
Solution flexibility	Evaluates the solution or code based on the ability to easily change or pursue future changes.	"That is a great advantage to their code because this makes revision to code easier"
Ideal Solution	Evaluates the feasibility of the solution or code based on an ideal or "perfect-world" solution.	"So you wanna go here ideally but it's such a short time period that that would be, to get the process that you want, that like it might not be feasible so you might have to go like at a little bit lower temperature."
Peer Comparison	Evaluates the quality of the solution or code by comparing their own solution with other peers or group members.	"What did you guys find to be your best temperature? Not that it'd be the same one as mine cause I did a different product but like about what?"
Debugging	Evaluating the code or solution periodically throughout the creation or solution process to ensure ability to run, reasonableness, or to collect feedback about the solution.	"So I was gonna do 4, and then 4 wasn't working for me, so I ended up just making it 8 because I figured mine was just too slow if there was pumpking pie filling, so it was like really dense and everything."

The two new outcomes are unsurprising in that they were found in the evaluation and reflection phases. Regarding comparing the ideal solution, students were comparing their model to a perfect-world situation where they didn't need to make tradeoffs or concessions in their design solution. This makes sense in that when comparing to other students who made different tradeoffs, one begins to see the limitations of various design decisions. Additionally, the outcome of peer comparison was prominent because this was the first stage of the modeling activity where students had the opportunity to compare and contrast among other groups and their peers more broadly. Students had completed their models, so they could compare model performance if their answers were similar or what each other's output graphs looked like. As such, this outcome became very visible through the analysis.

9.5.4 Generalization

Generalization as a practice was seen at a moderate level throughout the analysis, with all previous generalization outcomes showing up in the analysis. One new outcome emerged from the evaluation and reflection phase of the analysis, which was students talking about similar problems that were used for the solution. Table 27 below overviews the analysis's outcomes and example quotes for each.

Table 27. Observed generalization outcomes during model evaluation and reflection.

Outcome	Definition	Quotes
Drawing from previous experiences	Use of previous coursework or personal experiences to inform the current solution method. This can include previous coursework, professional experiences, or explicitly related concepts from previous courses.	"I did that for my [other course] project. I did 2D model."
Projecting to other or future applications	Reference to other problem spaces, external to the current problem, where the current solution would be useful or not useful based on that nature of the current problem or solution.	"D: It's weird to think, I wonder if you actually pay someone to do these kind of equations and work like, is there someone like [company] literally sits around and does this project over and over again for years. A: I don't know, I feel like there would be software packages out there… C: I was going to say, now that's where you have a software package."
Reuse of code/equations	Reusing or repeating portions of code or equations throughout the solution method.	"Yep. All I did was change one value, the ambient temperature outside, and I pasted the code for the heating."
Similar Problems	Using, comparing, or referencing information from a similar problem that is useful for the solution of the current problem.	"Similar practice example from textbook played role of guide that tell me what temperature profile is supposed to be."

Generalization was challenging to find in the data set and was much less frequently coded for than the previous three practices of abstraction, algorithmic thinking, and evaluation. The new outcome of similar problems mainly resulted from the reflection activity where students reflected on how challenges were overcome during their solution process. Because of this, students talked about similar issues that were used from sources such as the textbook or online videos and resources.

9.5.5 Decomposition

As in the previous two studies, decomposition continued to be one of the least frequent CT practices that were coded through the dataset. All previous outcomes from the first two studies were observed in the data. Table 28 below overviews the different outcomes associated with decomposition and example quotes for each from the dataset.

Table 28. Observed decomposition outcomes during model evaluation and reflection.

Outcome	Definition	Quote
Organization of larger solution method	Decomposing the code or solution method as an organizational tool.	"I used functions because it helped organize my code."
Allocating resources	Breaking up a problem amongst group members in order to reduce complexity of the work required or to increase quality of the solution method	"I guess, would it be easier to just split up 2 and 2. I guess we could just talk it through."
Ease of use of program	Decomposing the code or solution method so that the program or solution is easier to use.	"Functions could make the code much shorter."

Most of the decomposition observed throughout the evaluation and reflection phases of the modeling sequence was centered around breaking up the code in terms of functions. Sometimes to make the code more organized and sometimes to make the code easier to use, but almost always focused on decomposing the code into functions. This continues the trend throughout the entire modeling sequence, where decomposition outcomes have been scarce and difficult to observe in the dataset.

9.6 Discussion

The results of this study both confirm and build upon previous research, which has shown that modeling activities can be used to elicit computational thinking practices (Lyon & Magana, 2021). Specifically, this design used a model-eliciting activity, which has also been used in the literature to study how students use computational thinking (Z. Liu & Xia, 2021; Lyon & Magana, 2021). However, the results presented here focus on and highlight the importance and ability of the later stages of modeling and simulation process, evaluation and reflection, as capable and valuable in eliciting and giving practice to computational thinking within the engineering classroom. This is a useful finding in that not only is modeling a central skill to engineers across disciplines (Gainsburg, 2006; Lyon et al., 2019; Ortega-Alvarez et al., 2018; Ortiz-Rodriguez et al., 2010), but evaluative practices such as validation and verification are essential practices for engineering professionals as well (Komisar et al., 2018; Lo, 2016; Magana, 2017; Magana & de Jong, 2018).

The results indicate that computational thinking was used through the evaluation and reflection process, and specific new outcomes emerged or were emphasized to a greater degree. As expected, evaluation was a much more prominent computational thinking practice throughout the evaluation and reflection stages of the modeling intervention. The evaluation process allowed students to compare their solutions with their peers, which had not been identified as an outcome in previous studies (Lyon et al., n.d.; Lyon & Magana, 2021). This ability for students to compare their solution processes is a direct result and expected outcome of the learning design using principles of productive failure (Kapur & Bielaczyc, 2012). Additionally, in this study, students worked in teams to evaluate their models. This type of team-based MEA allows students to see diverse and unique perspectives different from their own (Moore et al., 2013). This type of comparison between solutions serves as a source of validation or feedback between students on their solutions.

Many studies in engineering education have previously reported the learning gains and benefits of getting feedback on solutions when comparing to worked examples (Moreno et al., 2009; Yan et al., 2015) as well as when getting feedback from peers (Carberry et al., 2016; Conde et al., 2017; Ekoniak et al., 2013). This feedback is even more beneficial when verbal feedback is similar to how the students were comparing solutions in this study's modeling intervention (Carberry et al., 2016). Students giving feedback on other solutions also increases student motivation within the course (Conde et al., 2017). Thus, the results here are both expected and encouraging. Our learning design elicited additional evaluative forms of computational thinking and may likely promote student learning and motivation with the course materials.

Another prominent outcome was students discussing the generalization process from similar problems found from different resources while solving the problem, such as online resources and the course text. This likely emerged from the previous studies on model planning and building because students finally had a fully realized solution that they had built and had the opportunity to understand how similar problems were both similar and valuable in constructing the current solution. This result is encouraging in that it shows a certain level of knowledge transfer from resources the students located in the textbook, online, and elsewhere.

Previous research has shown that project-based learning interventions, like the one in this study, promote knowledge transfer of conceptual, procedural, and factual knowledge (Lou et al., 2010). Additionally, when working in teams, studies have shown giving real-world case study learning design, similar to the prompt given in this study, students can obtain unique and diverse perspectives from their team members (Doukanari et al., 2020). Thus, our learning design supports these findings from the literature. Students who were engaged in group environment working on a real-world problem-based learning intervention showed evidence of knowledge transfer both as generalized from similar problems and the feedback from their peers.

This knowledge transfer may result from the model-based reasoning occurring as students work on their modeling solutions. Model-based reasoning is how people use and update mental models that represent how they understand the world around them. The produced computational model is a result (Lehrer & Schauble, 2003; Nersessian, 2007). As students use and evaluate their model results, they may conflict between the evaluation process results and how the student believes the model should act. Our results indicate that students are using model-based reasoning as they are constantly evaluating their model in terms of accuracy, efficiency, debugging, peer solutions, among many other factors, by comparing their perceptions of how the model should act to what they see in their actual models. This aligns with what has been seen in our previous studies looking at modeling interventions and computational thinking (Lyon et al., n.d.; Lyon & Magana, 2021).

9.6.1 Implications for Teaching and Learning

This work contributes to the broader literature in engineering education research and has implications for teaching and learning, allowing practitioners to bring theory into practice. This study suggests that new learning outcomes emerge when students can collect and give feedback from their peers on their design decisions through an evaluation process. This is a valuable piece of information previously seen in the literature (Carberry et al., 2016; Ekoniak et al., 2013). Yet, this work has shown similar benefits to mutual feedback between students happening concurrently and together. Instructors should look for ways to allow students to compare their solutions with each other and highlight the differences between them. This will enable students to get real-time feedback on their solutions and begin to understand the benefits and drawbacks of different solution strategies.

This study continues to build on previous evidence that implementing the entire modeling and simulation cycle into the classroom, including planning, evaluating (verifying and validating), and reflecting on the model, gives students opportunities to use unique skills that they might not otherwise use in just building models. The engineering curriculum is often crowded, and instructors find it challenging to add new material into the classroom (Magana & Silva Coutinho, 2017). The impulse may be to have students build or use models to save time, yet, this work and others continue to exemplify the unique and valuable insights to be gained by students through the whole modeling and simulation cycle (Czocher, 2018; Magana, 2017; Magana et al., 2019; Shiflet & Shiflet, 2014). Thus, this study implies that merely building or using a model is often only a limited use of the learning activity. Engaging students in the entire cycle may allow them to learn additional skills.

Regarding the implications for the classroom, a relevant finding was the emergence of the generalization outcome of transferring from relevant and similar examples. The use of worked examples has been commonly studied and shown to be useful in helping students understand complex concepts (Moreno et al., 2009; Vieira et al., 2019). The results of this study seem to indicate further that students seem to learn how to solve current problems by looking to similar worked problems and examples, whether that be from a class text or online resources. Instructors should provide access to problems and examples that help students make sense of the current project they are working on.

9.7 Conclusions and limitations

To conclude, evaluation and reflection are critical components to the learning process and have been used across many learning interventions. Evaluation and reflection are often vital components to the modeling and simulation cycle. In our study, students met to compare their modeling solutions and then reflected on the modeling process. By doing so, students could see multiple ways of solving a complex, ill-defined engineering problem. This study indicates that one such benefit of evaluation and reflection during modeling and simulation is that students produce new and unique computational thinking outcomes when utilized within our proposed learning design and build on previous computational thinking outcomes identified.

This study has some limitations given the nature of the methods. The first and primary one is that the results likely lack generalizability across contexts. This study uses design-based research, and the results are tied inseparably to the learning design, the disciplinary context, and the instructional team (Barab & Squire, 2004; DBRC, 2003). Because of this, similar studies will likely find similar results, but these contextual differences could lead to differences in the results. Additionally, demographic information was not collected on the individual participants, limiting generalizability across student backgrounds (Pawley, 2017). Finally, the results are limited by the sample size. Larger sample sizes may provide additional outcomes elicited throughout the evaluation and reflection process. Future work should look to understand how these outcomes change across contexts, disciplines, and learning designs. By doing so, researchers and educators can build on this work, and have an even more holistic and generalizable understanding of the nature of computational thinking within modeling learning environments.

9.8 Acknowledgements

This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under Grant Number DGE-1842166 as well as the National Science Foundation under Grant Number EEC-1449238. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

10. DISCUSSION AND IMPLICATIONS OF THE RESEARCH

10.1 Introduction

To tie together the results of all three studies, this chapter aims to discuss the results in light of three main areas. First, a summary overview of the results from the three studies is given to look at the results in full. Next, the research implications for engineering education are discussed in how model-based reasoning leads to many computational thinking practices and outcomes. Then, the implications on teaching and learning are discussed, considering these results.

10.2 Progression of Computational Thinking through Model-based Reasoning

The results of the three studies provide a lens to uncover how computational thinking is used throughout the modeling and simulation cycle. Furthermore, the findings of the studies examine the role of computation in promoting model-based reasoning within engineering students. A revised figure, initially presented in Chapter III, is given based on the findings. This revised Figure 12 represents the interplay between the modeling and simulation cycle with the model-based reasoning process and different types of models students generated throughout the modeling and simulation cycle.

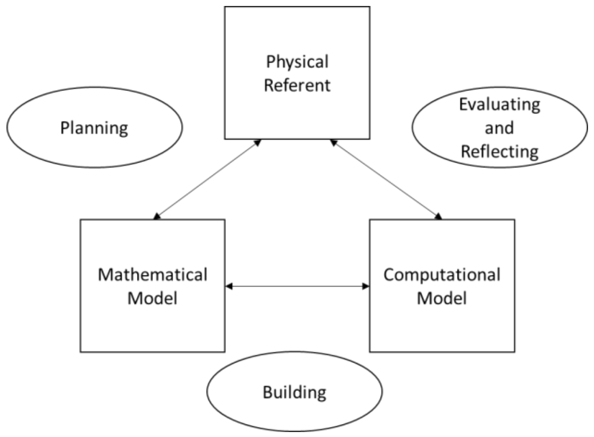

Figure 12. Overview of the computational modeling and simulation process and its alignment with model-based reasoning.

Each of the double-sided arrows in Figure 12 represents the model-based reasoning process as students move between the various representations of the model. At each point of reasoning, one of the learning phases is to guide that reasoning process. Planning the model is primarily when students move from the physical referent to the mathematical equations and variables needed to model the system mathematically. Students build their computational model based on their planned mathematical model during the building phase. And during the evaluating and reflecting phase, students compare their results to their expectations of the physical referent and world around them. The primary hypothesis of this framework is that computational thinking would emerge during each of these model-based reasoning processes.

The results indicate that there is indeed some sort of relationship between students using model-based reasoning and the subsequent elicited computational thinking outcomes. Figure 13 describes the ideas suggested by the results of the studies.

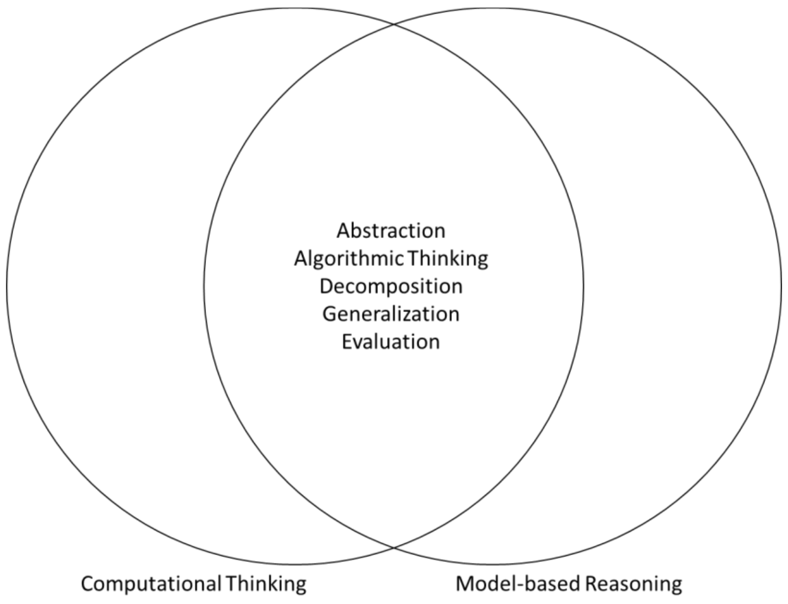

Figure 13. The relationship between computational thinking and model-based reasoning.

While this research doesn't claim that these two entities are one and the same, the results suggest that there is certainly some overlap between the concepts used in model-based reasoning and computational thinking. This is perhaps most highlighted by Nersessian (2007), who gives the following of definition of model-based reasoning in terms of the processes it consists of:

> "**abstraction**: limiting case, generic, idealization, **generalization**; simulation: inferring outcomes or new states via model manipulation (mental or physical); **evaluation**: goodness of fit, explanatory power, implications (empirical, mathematical); and adaptation: constraint satisfaction, coherence, other relevant considerations." (p. 706-707).

As one can see, there is significant overlap between the definitions and codebook provided throughout these studies in terms of CT practices and these definitional processes proposed for model-based reasoning. Nersessian (2007) additionally describes simulation as a key model-based reasoning process, which in our computational thinking coding scheme is highly related to algorithmic thinking. The difference being that mental simulation uses the mind as the simulating agent, whereas the students in these students were using the MATLAB programming environment as the simulating agent.

Certainly, the literature has many aspects to computational thinking that are not necessarily a part of model-based reasoning and vice versa. Thus, the relationship between the two can be thought of as overlapping concepts, but not entirely the same, as shown in Figure 13. Future work may continue to understand the relationship between these two concepts and the exact nature of the interaction. However, the results of these studies indicate that there certainly is a relationship and that when students are engaged in model-based reasoning, they are also engaged in computational thinking.

10.2.1 An Overview throughout the Modeling and Simulation Cycle

The results of the three studies suggest that computational thinking was an emergent pattern throughout the modeling and simulation cycle through the model-based reasoning processes. Figure 14 below shows each of the phases of the modeling and simulation process (planning, building, and evaluating/reflecting) and what percentage of the total coded practices at the categorical level adapted from the literature (Curzon et al., 2014; Selby & Woollard, 2013).

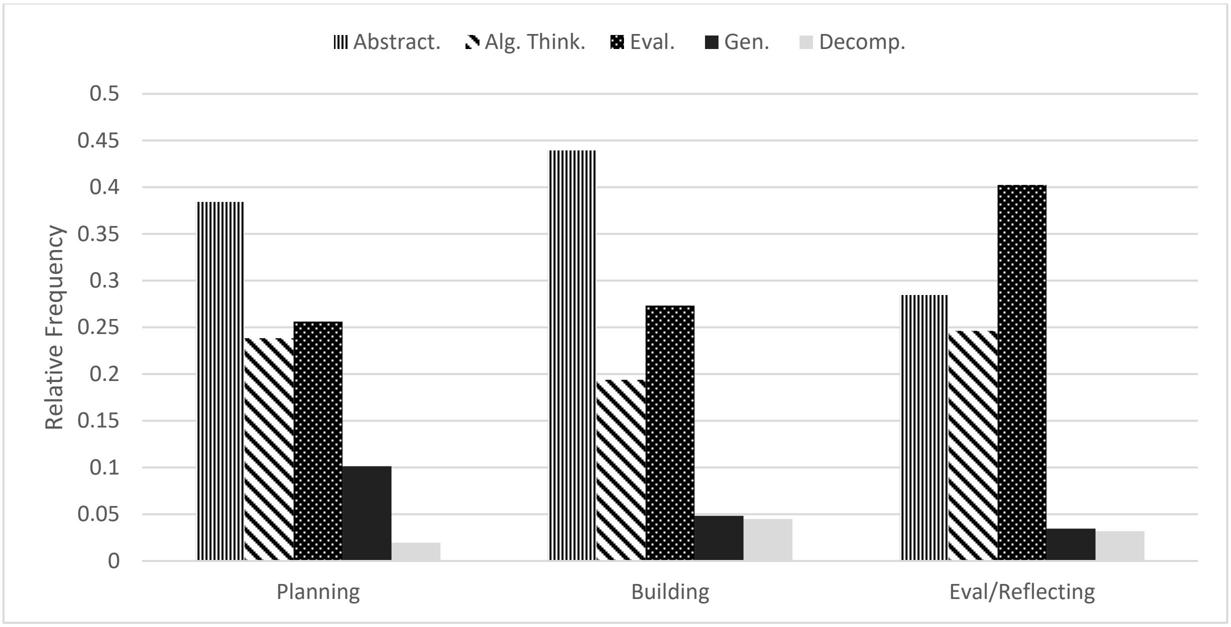

Figure 14. Frequency of categorical codes for each of the three studies.

The results of Figure 14 do not provide an exact representation of how often students used what practices, but they do note a few interesting trends. For example, abstraction was most enacted during the planning and building phases. Alternatively, evaluation was the most enacted practice during evaluation and reflection, an expected development. However, we also see that evaluation remained a prominently enacted practice throughout the entire process, indicating that students continuously evaluate their work in terms of many different factors, a practice common to many expert-learners (Ertmer & Newby, 1996). Generalization also increased during the planning phase, mainly because students had to transfer information from various sources while they thought about how to use that information to create their models.

Table 29. Complete list of computational thinking outcomes across the three studies.

CT Practice	CT Outcomes
Abstraction	· Ranges to values · Multiple to single · Dynamic to static · Geometric relationships · Similar systems · Infinite to finite
Algorithmic Thinking	· Indication of later use/effect · Stepwise approach · Parallel methods · Conditional logic
Evaluation	· Solution accuracy · Solution complexity · Time efficiency · Design criteria · Solution usability · Solution flexibility · Ideal Solution · Peer Comparison · Debugging
Generalization	· Previous coursework or experience · External applications · Similar problems · Reuse of code/equations
Decomposition	· Organization of larger solution method · Allocation of resources · Ease of use of program

The results of the three studies also provide a list of outcomes, listed in Table 29, associated with computational thinking that emerged when engineering students engaged in model-based reasoning through a model-eliciting activity. While this list is likely not exhaustive of all modeling problems and situations, it does give a broad overview of how modeling-based reasoning elicited computational thinking practices. Some categories resulted in more distinct outcomes than others, and this should not be taken to indicate anything about frequency or prominence within the dataset. But instead, it is an indication of how often there were clear differences between the outcomes in the category. For example, evaluation had the most different outcomes, and this was because there were many clear criteria against which students compared their solutions. Yet this says nothing about the frequency of the practice within the dataset.

Previous research has found that computation and computational modeling are good educational tools to study model-based reasoning in student solutions (Crnkovic & Cicchetti, 2017; Magana et al., 2020). This directly aligns with the studies presented here that have seen that computation was a valuable tool in studying model-based reasoning processes. Using computation in the engineering classroom, especially when creating computational models and simulations, promotes the use of multiple types of knowledge while engaging in model-based reasoning processes. These differ throughout the modeling cycle (Magana et al., 2020). Our results show a corresponding idea that the thinking processes elicited through model-based reasoning seem to vary and change across the modeling and simulation cycle.

The connection between computational thinking being emergent from model-eliciting activities has also been studied outside of the results presented here (Carmona et al., 2022; Z. Liu & Xia, 2021). The research continues to find connections between the MEA as a pedagogical tool and computational thinking as an emergent outcome. For example, Liu and Xia (2021) found that the MEA learning design promoted certain computational thinking practices but took a more quantitative approach to measure the computational thinking practices with rubrics. Our approach in these studies is additive to this. It qualitatively shows what these practices look like and identifies how these practices emerge as outcomes within student solutions. Thus, this study goes deeper than measuring computational thinking by characterizing it when enacted in the modeling and simulation cycle context and how it is used throughout the different forms of enacting model-based reasoning processes.

This work also furthers the work done by Curzon et al. (2014) and Selby and Woollard (2013), whose definition of computational thinking these studies operationalized. The studies presented here give greater detail to the definition and take it a step further to provide practical outcomes associated with the proposed definition.

10.3 Implications for Engineering Education Research

The implications of the results for engineering education research from the three studies relate to three primary areas. First, I will look at how the study contributes to the body of research around computational thinking and the implications the results have in furthering that body of research. Next, I will look at the studies' contribution to modeling education literature and how best to utilize these tools in the classroom. Finally, I will discuss the idea of computational model-based reasoning that emerges as students engage in modeling and simulation activities.

10.3.1 Computational Thinking: A Flag in the Ground

The results of this study have multiple implications for literature surrounding computational thinking. The literature around computational thinking has been, for the most part, definitional for many years (Grover & Pea, 2013; Kalelioğlu et al., 2016; Lyon & Magana, 2020b). While the definition continues not to be consensus, there is a significant overlap between many of them (Lyon & Magana, 2020b). The considerable gaps in the literature around computational thinking continue to be how to design pedagogy and activities that allow students to practice and learn computational thinking (Angeli & Giannakos, 2020).

The results of the studies from this dissertation work address this gap in two ways. First, a designed learning intervention is shown to clearly elicit multiple computational thinking practices within the undergraduate engineering curriculum. The designed learning intervention is based on model-eliciting activities, productive failure, and modeling-based learning principles (Diefes-Dux et al., 2004; Kapur & Bielaczyc, 2012; Louca & Zacharia, 2012). Yet, engineering education research should continue to connect the dots on how learning frameworks such as these allow students to practice computational thinking skills through different engineering contexts and learning interventions. Studies continue to emerge with computational thinking used in different pedagogies and learning frameworks such as makerspaces (Y. Yin et al., 2020), design education (Hynes et al., 2016), and project-based learning (Cruz Castro et al., 2021). This study adds and furthers this literature by showing how modeling-based learning, model-eliciting activities, and productive failure can critically influence computational thinking practices.

Second, it gives practical and contextual ways to convert the practices into outcomes. This brings the theory to practice, showing how computational thinking is actualized within student performance on an engineering task. One issue within the computational thinking literature is that it often results in vague descriptions of what was learned, or the concept of computational thinking is diluted to make it easier to measure (Lyon & Magana, 2020b). These studies give a full list of tangible computational thinking outcomes that emerge as a result of engaging in the modeling and simulation process. The connection between modeling and simulation with computational thinking has previously been noted and discussed in the literature (Z. Liu & Xia, 2021; Magana et al., 2013; Sanford & Naidu, 2017; Sengupta et al., 2013; Weintrop et al., 2016). However, the exact nature of this relationship in terms of model-based reasoning and the tangible outcomes that emerge as a result are critical contributions of these studies. Engineering education research should continue to look into the exact nature of this relationship and how these computational thinking outcomes are impacted within varied contexts.

10.3.2 Modeling Education: A Keystone of Engineering Education

Models and modeling are central elements to the educational development of engineering students. They are used across disciplines such as civil (Gainsburg, 2006), mechanical (Leang et al., 2010; Sclarsky et al., 2016), chemical (García-Herreros & Gõmez, 2013; Sclarsky et al., 2016), and electrical (Ortega-Alvarez et al., 2018) to give a few examples. Aside from their primary use of allowing students to connect the physical world around them from the mathematical language to interpret it, models allow engineering students to use deep reasoning processes to make decisions (Lehrer et al., 1994). They also allow instructors to add key elements to the curriculum such as programming education (Magana et al., 2017; Magana & Silva Coutinho, 2017).

The studies presented here contribute to the growing literature that the entire modeling and simulation cycle is key to student learning through modeling, not just building the model (Louca & Zacharia, 2012; Magana, 2017; Shiflet & Shiflet, 2014). Fundamental processes such as model planning, evaluating, and reflecting are needed for students to build additional skills they otherwise would not practice just in building the model, which was corroborated through our results. Many computational thinking practices were used in different and varied ways throughout the entire modeling intervention. This means that critical elements of learning may have been lost in the absence of any given phase of the modeling process.

Yet, one of the hurdles to implementation is that the engineering curriculum has very little space or time for modeling activities (Magana & Silva Coutinho, 2017). The results of these studies indicate that modeling education can promote computational thinking and aligns with the literature around the types of additional learning it promotes for students, such as programming knowledge (Caballero et al., 2012; Magana et al., 2016) and reflective practice (Jaiswal et al., 2021; Magana et al., 2019). Engineering education research should continue to look for learning benefits from integrating the entire modeling and simulation cycle into the classroom to be thoroughly and more frequently integrated.

One way to continue exploring how to integrate modeling and simulation into the curriculum is to use design-based research methods to further theory and contribute to practice simultaneously (Barab & Squire, 2004; DBRC, 2003). This study has contributed both to the theory around model-based reasoning and computational thinking and delivered a disciplinary modeling unit to integrate into a final-year capstone engineering course. Engineering education research should continue to look for ways to use design-based research to test and contribute learning designs in the classroom to understand how students learn, and instructors should teach in realistic contextualized environments.

10.3.3 The Emergence of Computational Model-based Reasoning

Finally, as it relates to the implications for engineering education research, the union of computational thinking and model-based reasoning emerged from engaging in modeling and simulation activities in the engineering classroom. The studies showed that as students engaged in the model-based reasoning process of converting their mental models to various forms of mathematical and computational models, diverse computational thinking practices emerged. The literature has previously noted this relationship between computation, modeling, and model-based reasoning (Crnkovic & Cicchetti, 2017; Develaki, 2017; H. P. Liu et al., 2017). The nature of the relationship is that computational modeling and simulation activities allow students to engage in model-based reasoning by testing their understanding of reality, which consists of their mental models, and to get feedback on the results. Computational modeling allows students to test their knowledge of complex phenomena quickly and in diverse ways.

Because of this, computational model-based reasoning extends from the literature around model-based reasoning in that both the created computational model and mental model of the student work together to update their understanding of the world around them. Develaki (2017) wrote that computer simulations build model-based reasoning "by facilitating the construction of numerous alternative models and immediate checking their underlying hypotheses" (p. 1023). In essence, the computational model can build and test complex situations that the students could never do on their own. The results of our study support the idea that model-based reasoning occurs through modeling and simulation activities and elicits computational thinking practices.

Engineering education researchers should continue to understand the deeply rooted benefits of modeling and simulation and how they can lead to this type of computational model-based reasoning process.

10.4 Implications for Teaching and Learning

However, the implications are not only for the research laboratory but also for the practitioner. The results from the studies have implications for teaching and learning, looking specifically at what the results imply about the engineering classroom. First, the implications for teaching are reviewed, looking foremost at the role of the teacher in the process of model-based reasoning, modeling, and computational thinking. Next, the implications for learning and how students learn computational thinking skills are reviewed.

10.4.1 Implications for Teaching

There are multiple implications for teaching and how modeling pedagogy can be set up to promote and elicit computational thinking from engineering students. First, the results indicate that allowing students to explore the problem space before the instruction is beneficial to modeling and simulation. This is especially true in upper-division courses such as the one presented here, where students need to build on previous information already given to them in the curriculum. The students needed to generalize from previous courses and experiences instead of having recent instruction on the modeling problem to rely on, forcing them to connect the current context with multiple previous contexts and knowledge. This is precisely what has been seen in the literature when productive failure is used in pedagogical design (Kapur & Bielaczyc, 2012).

Another implication for teaching for computational thinking in the context of modeling and simulation is to allow students to explore a problem space that is big enough for multiple solutions. Many of the abstraction outcomes that emerged from the students resulted from students needing to make decisions based on the complexity and real-world aspects of the problem. If the problem had been straightforward, students would not have needed to decide how to abstract the problem from the complex real-world context. The benefits of necessary complexity through ill-structured problems for learning are well-documented (Jonassen & Cho, 2011; Kapur, 2014; Shin et al., 2003) and setting modeling problems within a real-world context (Diefes-Dux et al., 2004;

Magana et al., 2016). Our findings align with these claims and show that they force students to abstract aspects of the problem from the real world to simplify the solution's complexities.

This leads to the final implication for teaching: students need to be given the ability to compare their solutions to other solutions. Multiple studies have shown that allowing students to compare solution methods in mathematics is beneficial in developing procedural and conceptual knowledge and will enable them to maintain more flexibility within their solution approach (Elisha, 2013; Rittle-Johnson & Star, 2007). The results of these studies indicate that there were computational thinking benefits during a mathematical modeling activity as well. It allowed students to compare how they set up the problem and the simplifications they made to abstract the problem from its real-world context in terms of accuracy and complexity, among others. This is similar to the findings of Elisha (2013), who wrote that "comparing contrasting solution methods […] helps students to differentiate essential features of a problem," (p. 22). In the same way, when students are comparing their modeling solutions, they can compare what essential simplifications were and what were not.

10.4.2 Implications for Learning

In addition to the implications for teaching, there are also implications from the studies for how students learn during the modeling process. The first is that students are constantly evaluating their solution approach, methods, and performance. This constant self-evaluation that students do on their work and performance is a crucial tenet of self-regulation (Urban & Urban, 2019) and calibration of work (Osterhage et al., 2019). While it is not surprising that by the time students are at their capstone experience, they are generally good at regulating their solution strategies, it is undoubtedly an encouraging finding that this work shows that students are constantly engaging in this process. This work finds that students focus on many more items than just accuracy when evaluating their solutions. Important metrics such as time efficiency, complexity, usability, and flexibility also seem significant to students.

The work also suggests that having teammates are a core part of learning in ill-structured modeling activities. This is not a new finding as much of the modeling literature discusses and demonstrates the benefits of working in teams within a modeling context (Diefes-Dux et al., 2004, 2006; Moore et al., 2013). Literature has shown that when working in teams together, groups can internally fix misconceptions among the group members (Moore et al., 2013). This work furthers that by giving some insight into the mechanism in how this occurs. Students were able to see different modeling assumptions, different coding strategies, and talk about the benefits and drawbacks of each as they worked through both the planning and evaluating stages of the modeling activity. This allowed students to give continuous feedback to each other as they planned their models and then as they evaluated what they had done. Through this process, students were able to help each other move towards correct ways of thinking about the problem.

Instructors should strive to scaffold the learning activity to lead students towards desired learning outcomes when giving ill-structured activities. The literature has shown time and time again that building in proper scaffolding into the learning activity allows students to learn more effectively as well as guide an activity toward the desired outcomes of the instructional team (Hislop, 2006; Hislop & Ellis, 2009; Lanier et al., 2016; Melero et al., 2012). Specifically, the learning design presented in these studies gave multiple forms of scaffolding to support student practices, such as defined report templates to structure student writing and thinking and using intermediate deliverables throughout the entire process rather than turning in the whole assignment at one time. The studies suggest that these scaffolding techniques help structure student learning in complex, ill-structured capstone projects, which aligns with previous research (Hislop, 2006).

10.5 Summary

This chapter has overviewed a discussion of the overall results of the three studies and given implications of the results in full. An overview of how computational thinking was used across the modeling and simulation sequence was given. The connection between model-based reasoning processes and computational thinking was made explicit to the theoretical framing of the entire set of studies. Next, the implications for engineering education were overviewed. The results generated tangible outcomes of what computational thinking looks like and a further demonstration that modeling is a crucial need for all disciplines of engineering students. The implications for teaching and learning were then overviewed. First, there was a discussion of the pedagogical implications of the studies and the ways instructors can promote computational thinking through concepts such as delayed instruction, creating ill-structured problems, and allowing students to compare their solutions. The implications to learning were then discussed, looking at students' self-evaluation and the benefits it can bring, as well as the learning that occurs when students can correct each other in their ways of thinking.

11. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

11.1 Future Research Directions

There are multiple essential and exciting research directions to build off the results presented in these three studies. As such, numerous research questions naturally result from these lines of inquiry such as:

- How can computational thinking practices, such as abstraction and generalization, be measured in the context of the engineering classroom?
- How much do students' computational thinking skills increase from engaging in modeling and simulation activities in the engineering classroom?
- How do computational thinking outcomes change when students engage in other engineering activities such as engineering design?
- In what ways do pedagogical design decisions impact the resulting computational thinking outcomes?
- What different learning types occur when students use computational thinking during modeling and simulation activities?
- In what ways do students benefit and struggle when engaging in model-based reasoning during a modeling and simulation task?
- How are computational thinking practices elicited through other types of modeling outside of computational modeling, including black-box approaches?
- How are computational thinking practices elicited through modeling at the K12 level, and in what ways is it different than undergraduates?
- How can instructors build computational thinking assessments for modeling and simulation activities once they have implemented them?
- How does the modeling and simulation context contribute to the elicited computational thinking?
- How can computational thinking activities be designed to broaden participation and close representation gaps in engineering and computing fields?

While there are many different directions one could go, here are two lines of inquiry I would like to discuss specifically. The first is looking at other essential engineering tools and processes and how they may elicit and utilize computational thinking in practice. Modeling is far from the only critical engineering skill; others include the design process (Crismond & Adams, 2012) and complex problem-solving (Schefer-Wenzl & Miladinovic, 2019). Understanding how these engineering tools elicit computational thinking outcomes would help to build a holistic picture of how computational thinking is used and utilized by engineering students and professionals as they go about all aspects of their work. Building out this list of practices would allow educators to fully understand how computational thinking is used by engineers and will enable them to target these practices throughout the engineering curriculum.

Similarly, future research should continue to look for how pedagogical design decisions impact the emerging computational thinking outcomes. While the studies did not have a control condition to compare, it is very likely that design decisions from a pedagogical standpoint strongly influenced the computational thinking outcomes that were elicited as a result. Because of this, varied pedagogical practice for learning with models would likely result in additional computational thinking outcomes. While this study used the MEA to structure the learning design, other modeling-based interventions exist within the literature (Lyon & Magana, 2020a). Research should continue to form the link between modeling, pedagogical design, and computational thinking.

11.2 Limitations

The study has multiple limitations that should be addressed. First is the limited generalizability of the study to new contexts and populations. This study sits contextually within the specifically designed learning intervention, the specific population of undergraduate engineering students in biological engineering, and the capstone course it was all situated within. While other studies may see similar results in their contexts, there will likely be differences due to the contextual differences. The hope for these differences would be that they would continue to build upon this list of computational thinking outcomes that emerge both during modeling activities and other critical engineering skills.

Similarly, the lack of demographic information on the participants limits the studies' ability to make claims across or about specific populations. Engineering has a long history of significant gaps in diversity in terms of race and sex. Engineering education research should work to make these gaps visible to readers and practitioners (Pawley, 2017). Because of the lack of demographic information on the students, this limits the generalizability of the results of the study.

Finally, the study results are limited by one crucial factor: to what degree students discuss their thinking practices. This has long been seen as a limitation to think-aloud interviews and methods like them that rely on the students to discuss or write down their thinking process or respond to a learning environment, which might be pretty difficult for the student to do (Leighton, 2021). This study is limited by what the students can realize they are doing and put into words or writing. Further studies could likely identify additional elicited outcomes that the students are unaware of if different research methods were utilized.

11.3 Conclusions

To conclude, computational thinking is a concept that is rapidly spreading throughout the STEM education literature and is seen as increasingly essential to develop in students across disciplines and industries (Acevedo-Borrega et al., 2022; Dolgopolovas & Dagienė, 2021; Li et al., 2020; Lyon & Magana, 2020b; Magana & Silva Coutinho, 2017). Additionally, modeling and simulation are critical engineering skills that engineers of all disciplines use (Gainsburg, 2006; Igual et al., 2018; Leang et al., 2010; Lyon & Magana, 2020a). These studies show that computational thinking is practiced frequently and in various ways throughout the modeling and simulation cycle.

Additionally, the studies presented here also move the discussion around computational thinking from theory to practice, showing that pedagogy used to elicit computational thinking has practical outcomes that educators can observe in their students' work. The studies also indicate that educators can make design decisions to elicit various computational thinking outcomes within their pedagogical practice. One such learning design connected to learning theory with computational modeling is given in these studies that can and should be used and modified to implement modeling and elicit computational thinking.

Finally, these studies utilized a design-based research approach to contribute to theory and deliver a learning activity to further real-world instructional practice. The results imply that engineering educators can implement computational modeling activities into their classrooms to allow students to engage in model-based reasoning, which will enable them to practice and build their computational thinking skills. And because the study uses design-based research, these results were obtained in an actual classroom in partnership with an engineering instructor. Because of this, the results are tested and derived from a natural learning environment, and the results of the learning design contribute to and form the basis for current and future instructional practice in the undergraduate engineering curriculum.

REFERENCES

Acevedo-Borrega, J., Valverde-Berrocoso, J., & Garrido-Arroyo, M. del C. (2022). Computational Thinking and Educational Technology: A Scoping Review of the Literature. *Education Sciences*, *12*(1), 39. https://doi.org/10.3390/educsci12010039

Alabi, O., Magana, A. J., & Garcia, R. E. (2015). Gibbs computational simulation as a teaching tool for students' understanding of thermodynamics of materials concepts. *Journal of Materials Education*, *37*(5–6), 239–260.

Anderson, T., & Shattuck, J. (2009). Design-based Research: A decade of progress in education research. *Educational Researcher*, *41*(1), 16–25.

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, *105*. https://doi.org/10.1016/j.chb.2019.106185

Bandura, A. (1998). Self-efficacy. *Encyclopedia of Human Behavior*, *4*(1994), 1–65. https://doi.org/10.1002/9780470479216.corpsy0836

Barab, S., & Squire, K. (2004). Design-based research : Putting a stake in the ground. *Journal of the Learning Sciences*, *13*(1), 1–14. https://doi.org/10.1207/s15327809jls1301_1

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning and Leading with Technology*, *38*(6), 20–23.

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is involved and What is the Role of the Computer Science Education Community? *ACM Inroads*, *2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Baxter, P., & Jack, S. (2008). The Qualitative Report Qualitative Case Study Methodology: Study Design and Implementation for Novice Researchers. *The Qualitative Report*, *13*(4), 544–559. https://doi.org/citeulike-article-id:6670384

Bohle Carbonell, K., Stalmeijer, R. E., Könings, K. D., Segers, M., & van Merriënboer, J. J. G. (2014). How experts deal with novel situations: A review of adaptive expertise. *Educational Research Review*, *12*, 14–29. https://doi.org/10.1016/j.edurev.2014.03.001

Borrego, M., & Henderson, C. (2014). Increasing the use of evidence-based teaching in STEM higher education: A comparison of eight change strategies. *Journal of Engineering Education*, *103*(2), 220–252. https://doi.org/10.1002/jee.20040

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Braun, V., Clarke, V., Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology Using thematic analysis in psychology. *Qualitative Research in Psychology*, *0887*(2006).

Brophy, S. P., Magana, A. J., & Strachan, A. (2013). Lectures and simulation laboratories to improve learners' conceptual understanding. *Advances in Engineering Education*, *3*(3), 1–27.

Buckley, B. C. (2012). Model-Based Learning. *Encyclopedia of the Sciences of Learning*, 2300–2303. https://doi.org/10.1007/978-1-4419-1428-6_589

Busch, T. (1995). Gender Differences in Self-Efficacy and Attitudes toward Computers. *Journal of Educational Computing Research*, *12*(2), 147–158. https://doi.org/10.2190/h7e1-xmm7-gu9b-3hwr

Caballero, M. D., Kohlmyer, M. A., & Schatz, M. F. (2012). Implementing and assessing computational modeling in introductory mechanics. *Physical Review Special Topics - Physics Education Research*, *8*(2), 1–15. https://doi.org/10.1103/PhysRevSTPER.8.020106

Campbell, K., Overeem, I., & Berlin, M. (2013). Taking it to the streets: The case for modeling in the geosciences undergraduate curriculum. *Computers and Geosciences*, *53*, 123–128. https://doi.org/10.1016/j.cageo.2011.09.006

Carberry, A. R., Brunhaver, S. R., Csavina, K. R., & McKenna, A. F. (2016). Comparison of written versus verbal peer feedback for design projects. *International Journal of Engineering Education*, *32*(3), 1458–1471.

Carey, C. C., & Gougis, R. D. (2017). Simulation Modeling of Lakes in Undergraduate and Graduate Classrooms Increases Comprehension of Climate Change Concepts and Experience with Computational Tools. *Journal of Science Education and Technology*, *26*(1), 1–11. https://doi.org/10.1007/s10956-016-9644-2

Carmona, G., Galarza-Tohen, B., & Martinez-Medina, G. (2022). *Exploring Interactions Between Computational and Critical Thinking in Model-Eliciting Activities Through Epistemic Network Analysis*. Springer International Publishing. https://doi.org/10.1007/978-3-030-93859-8_23

Case, J. M., & Light, G. (2011). Emerging Methodologies in Engineering Education Research. *Journal of Engineering Education*, *100*(1), 186–210. https://doi.org/10.1002/j.2168-9830.2011.tb00008.x

Chenail, R. J. (2011). Interviewing the investigator: Strategies for addressing instrumentation and researcher bias concerns in qualitative research. *Qualitative Report*, *16*(1), 255–262.

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design Experiments in Educational Research. *Educational Researcher*, *32*(1), 9–13. https://doi.org/10.3102/0013189X032001009

CollegeBoard. (2013). *The College Board AP ® Computer Science Principles Draft Curriculum Framework.* https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxjc3ByaW5jaXBsZXNwaWxvdGlpfGd4OjNkZWM5ZTY4ODQ4NzZlOWE

Conde, M. A., Sánchez-González, L., Matellán-Olivera, V., & Rodríguez-Lera, F. J. (2017). Application of peer review techniques in engineering education. *International Journal of Engineering Education*, *33*(2), 918–926.

Confrey, J. (2006). The evolution of design studies as methodology. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 135–152). Cambridge University Press. https://doi.org/10.1017/CBO9780511816833.010

Craik, K. (1943). *The nature of explanation* (C. U. Press (ed.)).

Crismond, D. P., & Adams, R. S. (2012). The informed design teaching and learning matrix. *Journal of Engineering Education*, *101*(4), 738–797. https://doi.org/10.1002/j.2168-9830.2012.tb01127.x

Crnkovic, G. D., & Cicchetti, A. (2017). *Computational Aspects of Model-Based Reasoning* (Issue January 2018). https://doi.org/10.1007/978-3-319-30526-4

Cruz Castro, L. M., Magana, A. J., Douglas, K. A., & Boutin, M. (2021). Analyzing Students' Computational Thinking Practices in a First-Year Engineering Course. *IEEE Access*, *9*, 33041–33050. https://doi.org/10.1109/ACCESS.2021.3061277

Curzon, P., Dorling, M., Selby, C., & Woollard, J. (2014). Developing computational thinking in the classroom: a framework. *Computing at School*, *June*.

Czocher, J. A. (2018). How does validating activity contribute to the modeling process? *Educational Studies in Mathematics*, *99*(2), 137–159. https://doi.org/10.1007/s10649-018-9833-4

Davis, R., & Hamscher, W. (1988). *Model-based reasoning: Troubleshooting*.

DBRC. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, *32*(1), 5–8. https://doi.org/10.3102/0013189X032001005

De Koning, K., Bredeweg, B., Breuker, J., & Wielinga, B. (2000). Model-based reasoning about learner behaviour. *Artificial Intelligence*, *117*(2), 173–229. https://doi.org/10.1016/S0004-3702(99)00106-X

Delany, D. (2008). Advanced concept mapping: Developing adaptive expertise. In A. J. Canas, P. Reiska, M. K. Ahlberg, & J. D. Novak (Eds.), *Concept Mapping - Connecting Educators: Proceedings of the Third International Conference on Concept Mapping: Vol. 3. Posters* (pp. 32–35). Institute for Human & Machine Cognition. http://cmc.ihmc.us/cmc2008Proceedings/cmc2008 - Vol 3.pdf

Dennett, D. C. (2000). Making tools for thinking. In D. Sperber (Ed.), *Metarepresentations: A multidisciplinary perspective* (pp. 17–29). Oxford University Press.

Develaki, M. (2017). Using Computer Simulations for Promoting Model-based Reasoning: Epistemological and Educational Dimensions. *Science and Education*, *26*(7–9), 1001–1027. https://doi.org/10.1007/s11191-017-9944-9

Dewey, J. (1938). *Experience and education*. https://doi.org/10.1007/s13398-014-0173-7.2

Diefes-Dux, H. A., Hjalmarson, M. A., & Zawojewski, J. S. (2013). Student Team Solutions to an Open-Ended Mathematical Modeling Problem: Gaining Insights for Educational Improvement. *Journal of Engineering Education*, *102*(1), 179–216. https://doi.org/10.1002/jee.20002

Diefes-Dux, H. A., Hjalmarson, M., Zawojewski, J. S., & Bowman, K. (2006). Quantifying aluminum crystal size part 1: The model-eliciting activity. *Journal of STEM Education: Innovations and Research*, *7*(1–2), 51–63.

Diefes-Dux, H. A., Moore, T., Zawojewski, J., Imbrie, P. K., & Follman, D. (2004). A framework for posing open-ended engineering problems: Model-eliciting activities. *Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference*. https://doi.org/10.1109/FIE.2004.1408556

Diefes-Dux, H. A., Zawojewski, J. S., Hjalmarson, M. A., & Cardella, M. E. (2012). A framework for analyzing feedback in a formative assessment system for mathematical modeling problems. *Journal of Engineering Education*, *101*(2), 375–406.

Dolgopolovas, V., & Dagienė, V. (2021). Computational thinking: Enhancing STEAM and engineering education, from theory to practice. *Computer Applications in Engineering Education*, *29*(1), 5–11. https://doi.org/10.1002/cae.22382

Doukanari, E., Ktoridou, D., & Epaminonda, E. (2020). Multidisciplinary and multicultural knowledge transfer and sharing in higher education teamworking. *IEEE Global Engineering Education Conference, EDUCON*, *2020-April*, 1836–1843. https://doi.org/10.1109/EDUCON45650.2020.9125401

Driscoll, D. L. (2011). Introduction to primary research: Observations, surveys, and interviews. In C. Lowe & P. Zemliansky (Eds.), *Writing Spaces: Readings on Writing* (Vol. 2, pp. 153–174). Parlor. https://doi.org/10.1109/SPEEDAM.2006.1649864

Dvir, D., Raz, T., & Shenhar, A. J. (2003). An empirical analysis of the relationship between project planning and project success. *International Journal of Project Management*, *21*(2), 89–95. https://doi.org/10.1016/S0263-7863(02)00012-1

Ehsan, H., Rehmat, A. P., & Cardella, M. E. (2021). Computational thinking embedded in engineering design: capturing computational thinking of children in an informal engineering design activity. *International Journal of Technology and Design Education*, *31*(3), 441–464. https://doi.org/10.1007/s10798-020-09562-5

Ekoniak, M., Scanlon, M. J., & Mohammadi-Aragh, M. J. (2013). Improving student writing through multiple peer feedback. *Proceedings - Frontiers in Education Conference, FIE*, 626–628. https://doi.org/10.1109/FIE.2013.6684901

Elisha, Z. H. (2013). The benefits of comparing solution methods in solving equations. *IOSR Journal of Research & Method in Education (IOSRJRME)*, *3*(1), 18–23. https://doi.org/10.9790/7388-0311823

Ellingson, L. (2009). *Engaging Crystallization in Qualitative Research: An introduction*. SAGE Publications.

Elo, S., Kaarlainen, M., Kanste, O., Polkki, T., Utriainen, K., & Kyngas, H. (2014). Qualitative Content Analysis: A focus on trustworthiness. *SAGE Open*, 1–10. https://doi.org/10.1177/2158244014522633

Elo, S., & Kyngäs, H. (2008). The qualitative content analysis process. *Journal of Advanced Nursing*. https://doi.org/10.1111/j.1365-2648.2007.04569.x

Ertmer, P. A., & Newby, T. J. (1993). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly*, *6*(6), 50–72.

Ertmer, P. A., & Newby, T. J. (1996). The expert learner: Strategic, self-regulated, and reflective. *Instructional Science*, *24*(1), 1–24. https://doi.org/10.1007/BF00156001

Ertmer, P. A., & Newby, T. J. (2015). *The expert learner : Strategic , self-regulated , and reflective . Instructional Science , 26 , 1-26. DECEMBER 1995*, 1–24. https://doi.org/10.1007/BF00156001

Evia, C., Sharp, M. R., & Perez-Quinones, M. A. (2015). Teaching Structured Authoring and DITA Through Rhetorical and Computational Thinking. *IEEE Transactions on Professional Communication*, *58*(3), 328–343. https://doi.org/10.1109/TPC.2016.2516639

Faraday, M. (2004). *Experimental researches in electricity*. ((Reprint), Vol. 136). Dover Publications. https://doi.org/10.1098/rstl.1846.0001

Fennell, H. W., Lyon, J. A., Magana, A. ., Rebello, S., Rebello, C., & Piedrahita, Y. (2019). Designing hybrid physics labs: combining simulation and experiment for teaching computational thinking in first-year engineering. *Proceedings of the 49th IEEE-ERM Frontiers in Education Conference (FIE)*.

Finzer, W. (2013). The Data Science Education Dilemma. *Technology Innovations in Statistics Education*, *7*(2). http://escholarship.org/uc/item/6jv107c7

Flanigan, A. E., Peteranetz, M. S., Shell, D. F., & Soh, L. K. (2017). Implicit intelligence beliefs of computer science students: Exploring change across the semester. *Contemporary Educational Psychology*, *48*, 179–196. https://doi.org/10.1016/j.cedpsych.2016.10.003

Froyd, J. E., Wankat, P. C., & Smith, K. A. (2012). Five major shifts in 100 years of engineering education. *Proceedings of the IEEE*, *100*(SPL CONTENT), 1344–1360.

Gainsburg, J. (2006). The mathematical modeling of structural engineers. *Mathematical Thinking and Learning*, *8*(1), 3–36. https://doi.org/10.1207/s15327833mtl0801_2

García-Herreros, P., & Gõmez, J. M. (2013). Modeling and optimization of a crude distillation unit: A case study for undergraduate students. *Computer Applications in Engineering Education*, *21*(2), 276–286. https://doi.org/10.1002/cae.20469

Gentner, D., & Gentner, D. R. (1983). Flowing waters or teeming crowds: mental models of electricity. In D. Gentner & L. Stevens (Eds.), *Mental Models*. Erlbaum.

Ginsburg, H. P., & Opper, S. (1988). *Piaget's Theory of Intellectual Development* (3rd ed.). Prentice Hall.

Google. (2015). *Computational thinking concepts guide*. https://docs.google.com/document/d/1i0wg-BMG3TdwsShAyH_0Z1xpFnpVcMvpYJceHGWex_c/edit

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Guba, E. G. E., & Lincoln, Y. S. Y. (1994). Competing Paradigms in Qualitative Research. In *Handbook of qualitative research* (pp. 105–117). https://doi.org/http://www.uncg.edu/hdf/facultystaff/Tudge/Guba%20&%20Lincoln%201994.pdf

Hamilton, E., Lesh, R., Lester, F., & Brilleslyper, M. (2008). Model-Eliciting Activities (MEAs) as a Bridge Between Engineering Education Research and Mathematics Department of Mathematical Sciences. *Advances in Engineering Education*, *1*(2), 1–25.

Harrison, A. G., & Treagust, D. F. (2000). A typology of school science models. *International Journal of Science Education*, *22*(9), 1011–1026. https://doi.org/10.1080/095006900416884

Hatano, G., & Inagaki, K. (1984). Two courses of expertise. *Research and Clinical Center for Child Development Annual Report*, *6*, 27–36.

Hesse, M. (2000). Models and analogies. In *A companion to the philosophy of science* (pp. 299–307). Blackwell. https://doi.org/10.1111/b.9780631230205.2001.00047.x

Hislop, G. W. (2006). Scaffolding student work in capstone design courses. *Proceedings - Frontiers in Education Conference, FIE*, 18–21. https://doi.org/10.1109/FIE.2006.322630

Hislop, G. W., & Ellis, H. J. C. (2009). Using scaffolding to improve written communication of software engineering students. *ITNG 2009 - 6th International Conference on Information Technology: New Generations*, 707–712. https://doi.org/10.1109/ITNG.2009.31

Hjalmarson, M., Diefes-Dux, H. A., Bowman, K., & Zawojewski, J. S. (2006). Quantifying aluminum crystal size part 2: The model-development sequence. *Journal of STEM Education: Innovations and Research*, *7*(1–2), 64–73.

Hmelo-Silver, C. E., & Pfeffer, M. G. (2004). Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions. *Cognitive Science*, *28*(1), 127–138. https://doi.org/10.1016/S0364-0213(03)00065-X

Hunt, R. J. (1986). Percent Agreement, Pearson's Correlation, and Kappa as Measures of Inter-examiner Reliability. *Journal of Dental Research*, *65*(2), 128–130. https://doi.org/10.1177/00220345860650020701

Hutchison, M. A., Follman, D. K., Sumpter, M., & Bodner, G. M. (2006). Factors influencing the self-efficacy beliefs of first-year engineering students. *Journal of Engineering Education*, *95*(1), 39–47. https://doi.org/10.1002/j.2168-9830.2006.tb00876.x

Hynes, M. M., Moore, T. J., Cardella, M. E., Tank, K. M., Purzer, S., Menekse, M., & Brophy, S. P. (2016). Inspiring computational thinking in young children’s engineering design activities (Fundamental). *ASEE Annual Conference and Exposition, Conference Proceedings*, *2016-June*. https://doi.org/10.18260/p.25732

Ifenthaler, D., & Seel, N. M. (2011). A longitudinal perspective on inductive reasoning tasks. Illuminating the probability of change. *Learning and Instruction*, *21*(4), 538–549. https://doi.org/10.1016/j.learninstruc.2010.08.004

Ifenthaler, D., & Seel, N. M. (2013). Model-based reasoning. *Computers & Education, 64*(1), 131–142. https://doi.org/10.1016/j.compedu.2012.11.014

Igual, R., Plaza, I., Marcuello, J. J., & Arcega, F. (2018). A survey on modeling and simulation practices for teaching power harmonics. *Computer Applications in Engineering Education*, *August 2017*, 804–823. https://doi.org/10.1002/cae.21911

Ilic, U., Haseski, H. I., & Tugtekin, U. (2018). Publication Trends Over 10 Years of Computational Thinking Research. *Contemporary Educational Technology,* *9*(2), 131–153. https://doi.org/10.30935/cet.414798

Inhelder, B., & Piaget, J. (1958). *The growth of logical thinking from childhood to adolescence*. Basic Books.

Irizarry, R. A. (2020). The Role of Academia in Data Science Education. *Harvard Data Science Review*, 1–8. https://doi.org/10.1162/99608f92.dd363929

Jaiswal, A., Lyon, J. A., Zhang, Y., & Magana, A. J. (2021). Supporting student reflective practices through modelling-based learning assignments. *European Journal of Engineering Education*, *0*(0), 1–20. https://doi.org/10.1080/03043797.2021.1952164

Jeon, Y., & Kim, T. (2017). The effects of the computational thinking-based programming class on the computer learning attitude of non-major students in the teacher training college. *Journal of Theoretical and Applied Information Technology*, *95*(17), 4330–4339.

Joffe, H, & Yardley, L. (2004). Content and thematic analysis. In *Research methods for clinical and health psychology* (pp. 56–68). Sage.

Joffe, Helene. (2012). Thematic Analysis. In D. Harper & A. Thompson (Eds.), *Qualitative Research Methods in Mental Health and Psychotherapy: A guide for student practitioners* (pp. 209–223). Wiley-Blackwell. https://doi.org/10.1002/9781119973249.ch15

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and conciousness.* Harvard University Press.

Johnson-Laird, P. N., Girotto, V., Legrenzi, P., Legrenzi, M. S., & Caverni, J. P. (1999). Naive probability: A mental model theory of extensional reasoning. *Psychological Review*, *106*(1), 62–88. https://doi.org/10.1037/0033-295X.106.1.62

Johri, A., & Olds, B. M. (2011). Situated Engineering Learning: Bridging Engineering Education Research and the Learning Sciences. *Journal of Engineering Education*, *100*(1), 151–185. https://doi.org/10.1002/j.2168-9830.2011.tb00007.x

Jonassen, D. H. (1991). Evaluating Constructivistic Learning. *Educational Technology, 31*(9), 28–33.

Jonassen, D. H. (2009). Externally modeling mental models. In L. Moller, J. Bond Huett, & D. M. Harvey (Eds.), *Learning and Instructional Technologies for the 21st Century* (pp. 49–74). Springer. https://doi.org/10.1007/978-0-387-09667-4_4

Jonassen, D. H., Strobel, J., & Gottdenker, J. (2005). Model building for conceptual change. *Interactive Learning Environments*, *13*(1–2), 15–37. https://doi.org/10.1080/10494820500173292

Jung, H., Diefes-Dux, H. A., Horvath, A. K., Rodgers, K. J., & Cardella, M. E. (2015). Characteristics of feedback that influence student confidence and performance during mathematical modeling. *International Journal of Engineering Education*, *31*(1), 42–57.

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic J . Modern Computing*, *4*(3), 583–596.

Kapur, M. (2008). Productive failure. *Cognition and Instruction*, *26*(3), 379–424. https://doi.org/10.1080/07370000802212669

Kapur, M. (2010). Productive failure in mathematical problem solving. *Instructional Science, 38*(6), 523–550. https://doi.org/10.1007/s

Kapur, M. (2011). A further study of productive failure in mathematical problem solving: unpacking the design components. *Instructional Science, 39*(4), 561–579. https://doi.org/10.1007/sll251-009-9093-x

Kapur, M., & Bielaczyc, K. (2012). Designing for Productive Failure. *Journal of the Learning Sciences*, *21*(1), 45–83. https://doi.org/10.1080/10508406.2011.591717

Kelly, A. E. (2004). Design research in education: Yes, but is it methodological? *The Journal of the Learning Sciences*, *13*(1), 115–128. https://doi.org/10.1207/s15327809jls1301_6

Kolodner, J. (1992). An introduction to case-based reasoning. *Artificial Intelligence*, *6,* 3–34. https://doi.org/10.1136/bmj.4.5576.398

Komisar, V., Flood, A., Walji, N., Foster, J., & Irish, R. (2018). Teaching Credible Validation and Verification Methods To a Large, Multidisciplinary First-Year Engineering Design Class. *Proceedings of the Canadian Engineering Education Association (CEEA),* 1–8. https://doi.org/10.24908/pceea.v0i0.10515

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005

Koton, P. A. (1985). Empirical and Model-based Reasoning in Expert Systems. *IJCAI, 85*, 297–299.

Krampe, D., & Lusti, M. (1997). Case-based reasoning for information system design. *International Conference on Case-Based Reasoning, 1266,* 63–73. https://doi.org/10.1007/3-540-63233-6_479

Lanier, A. S., Khandha, A., Rooney, S. I., Santare, M. H., Higginson, J., & Buckley, J. (2016). Improving scientific writing capability in an undergraduate population using a fading paradigm scaffolding approach. *ASEE Annual Conference and Exposition, Conference Proceedings*, *2016-June*(2004). https://doi.org/10.18260/p.25626

Lawanto, O. (2010). Students' Metacognition During an Engineering Design Project. *Performance Improvement Quarterly*, *23*(2), 117–136. https://doi.org/10.1002/piq

Leang, K. K., Zou, Q., & Pannozzo, G. (2010). Teaching modules on modeling and control of piezoactuators for system dynamics, controls, and mechatronics courses. *IEEE Transactions on Education*, *53*(3), 372–383.

Lee, O., Quinn, H., & Valdés, G. (2013). Science and Language for English Language Learners in Relation to Next Generation Science Standards and with Implications for Common Core State Standards for English Language Arts and Mathematics. *Educational Researcher*, *42*(4), 223–233. https://doi.org/10.3102/0013189X13480524

Lehrer, R., Horvath, J., & Schauble, L. (1994). Developing Model-Based Reasoning. *Interactive Learning Environments*, *4*(3), 218–232. https://doi.org/10.1080/1049482940040304

Lehrer, R., & Schauble, L. (2003). Origins and evolution of model-based reasoning in mathematics and science. In R. A. Lesh & H. M. Doerr (Eds.), *Beyond constructivism: Models and modeling perspectives on mathematical problem solving, learning, and teaching* (pp. 59–70). Lawrence Erlbaum Associates.

Leighton, J. P. (2021). Rethinking Think-Alouds: The Often-Problematic Collection of Response Process Data. *Applied Measurement in Education*, *34*(1), 61–74. https://doi.org/10.1080/08957347.2020.1835911

Lesh, R. A., & Zawojewski, J. (2007). Problem solving and modeling. In F. K. Lester, Jr. (Ed.), *Second handbook of research on mathematics teaching and learning* (pp. 763–804). Information Age.

Lesh, R., & Harel, G. (2003). Problem Solving, Modeling, and Local Conceptual Development. *Mathematical Thinking and Learning*, *5*(2–3), 157–189. https://doi.org/10.1080/10986065.2003.9679998

Lesh, R., Hoover, M., Hole, B., Kelly, A., & Post, T. (2000). Principles for developing thought-revealing activities for students and teachers. In A. Kelly & R. Lesh (Eds.), *The handbook of research design in mathematics and science education* (pp. 591–646). Lawrence Erlbaum Associates. https://doi.org/10.4324/9781410602725.ch21

Leutner, D., Leopold, C., & Sumfleth, E. (2009). Cognitive load and science text comprehension: Effects of drawing and mentally imagining text content. *Computers in Human Behavior*, *25*(2), 284–289. https://doi.org/10.1016/j.chb.2008.12.010

Lew, M. D. N., & Schmidt, H. G. (2011). Self-reflection and academic performance: Is there a relationship? *Advances in Health Sciences Education*, *16*(4), 529–545. https://doi.org/10.1007/s10459-011-9298-z

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020). Computational Thinking Is More about Thinking than Computing. *Journal for STEM Education Research*, *3*(1), 1–18. https://doi.org/10.1007/s41979-020-00030-2

Liu, H. P., Perera, S. M., & Klein, J. W. (2017). Using Model-based Learning to Promote Computational Thinking Education. In *Educational Communications and Technology: Issues and Innovations* (pp. 153–172). Springer.

Liu, Z., & Xia, J. (2021). Enhancing computational thinking in undergraduate engineering courses using model-eliciting activities. *Computer Applications in Engineering Education*, *29*(1), 102–113. https://doi.org/10.1002/cae.22357

Lo, S. H. R. (2016). Verification and validation as a key driver in modern engineering education. *IRA International Journal of Education and Multidisciplinary Studies (ISSN 2455–2526)*, *4*(1), 158–166. https://doi.org/10.21013/jems.v4.n1.p18

Lou, S. J., Shih, R. C., Tseng, K. H., Diez, C. R., & Tsai, H. Y. (2010). How to promote knowledge transfer through a problem-based learning internet platform for vocational high school students. *European Journal of Engineering Education*, *35*(5), 539–551. https://doi.org/10.1080/03043797.2010.489938

Louca, L. T., & Zacharia, Z. C. (2012). Modeling-based learning in science education: Cognitive, metacognitive, social, material and epistemological contributions. *Educational Review*, *64*(4), 471–492. https://doi.org/10.1080/00131911.2011.628748

Louca, L. T., & Zacharia, Z. C. (2019). Towards an Epistemology of Modeling-Based Learning in Early Science Education. In *Towards a Competence-Based View on Models and Modeling in Science Education.* (pp. 237–256). Springer, Cham. https://doi.org/https://doi.org/10.1007/978-3-030-30255-9_14

Lucangeli, D., Tressoldi, P. E., & Cendron, M. (1998). Cognitive and Metacognitive Abilities Involved in the Solution of Mathematical Word Problems: Validation of a Comprehensive Model. *Contemporary Educational Psychology*, *23*(3), 257–275. https://doi.org/10.1006/ceps.1997.0962

Lyon, J. A., Fennell, H. W., & Magana, A. J. (2020). Characterizing students ' arguments and explanations of a discipline-based computational modeling activity. *Computer Applications in Engineering Education*, 1–16. https://doi.org/10.1002/cae.22256

Lyon, J. A., Jaiswal, A., & Magana, A. J. (2020). The Use of MATLAB Live as a Technology-enabled Learning Environment for Computational Modeling Activities within a Capstone Engineering Course. *ASEE Annual Conference and Exposition, Conference Proceedings*.

Lyon, J. A., & Magana, A. J. (2020a). A Review of Mathematical Modeling in Engineering Education. *International Journal of Engineering Education*, *36*(1), 101–116.

Lyon, J. A., & Magana, A. J. (2020b). Computational thinking in higher education : A review of the literature. *Computer Applications in Engineering Education*, 1–16. https://doi.org/10.1002/cae.22295

Lyon, J. A., & Magana, A. J. (2021). The use of engineering model-building activities to elicit computational thinking : A design-based research study. *Journal of Engineering Education*, 1–23. https://doi.org/10.1002/jee.20372

Lyon, J. A., Magana, A. J., & Okos, M. (2019). WIP: Designing modeling-based learning experiences within a capstone engineering course. *ASEE Annual Conference and Exposition*.

Lyon, J. A., Magana, A. J., Streveler, R. A., & Perera, V. (n.d.). *Investigating Student Approaches to Model Planning Activities with Computational Thinking*. *[in prep]*.

Magana, A. J. (2017). Modeling and simulation in engineering education: A learning progression. *Journal of Professional Issues in Engineering Education and Practice*, *143*(4). https://doi.org/10.1061/(ASCE)EI.1943-5541.0000338

Magana, A. J., Brophy, S. P., & Bodner, G. M. (2012). Student views of engineering professors technological pedagogical content knowledge for integrating computational simulation tools in nanoscale. *International Journal of Enginnering Education*, *28*(5), 1033–1045.

Magana, A. J., & de Jong, T. (2018). Modeling and simulation practices in engineering education. *Computer Applications in Engineering Education*, *26*(4), 731–738. https://doi.org/10.1002/cae.21980

Magana, A. J., Falk, M. L., & Reese Jr., M. J. (2013). Introducing Discipline-Based Computing in Undergraduate Engineering Education. *ACM Transactions on Computing Education*, *13*(4), 16:1-16:22. https://doi.org/10.1145/2534971

Magana, A. J., Falk, M. L., Vieira, C., & Reese, M. J. (2016). A case study of undergraduate engineering students' computational literacy and self-beliefs about computing in the context of authentic practices. *Computers in Human Behavior*, *61*, 427–442. https://doi.org/10.1016/j.chb.2016.03.025

Magana, A. J., Falk, M. L., Vieira, C., Reese, M. J., Alabi, O., & Patinet, S. (2017). Affordances and challenges of computational tools for supporting modeling and simulation practices. *Computer Applications in Engineering Education*, *25*(3), 352–375. https://doi.org/10.1002/cae.21804

Magana, A. J., Fennell, H. W., Vieira, C., & Falk, M. L. (2019). Characterizing the interplay of cognitive and metacognitive knowledge in computational modeling and simulation practices. *Journal of Engineering Education*, *108*(2), 276–303. https://doi.org/10.1002/jee.20264

Magana, A. J., & Silva Coutinho, G. (2017). Modeling and simulation practices for a computational thinking-enabled engineering workforce. *Computer Applications in Engineering Education*, *25*(1), 62–78. https://doi.org/10.1002/cae.21779

Magana, A. J., Vieira, C., Fennell, H. W., Roy, A., & Falk, M. L. (2020). Undergraduate Engineering Students' Types and Quality of Knowledge Used in Synthetic Modeling. *Cognition and Instruction*, 1–35. https://doi.org/10.1080/07370008.2020.1792912

Malyn-Smith, J., & Lee, I. (2012). Application of the Occupational Analysis of Computational Thinking-Enabled STEM Professionals as a Program Assessment Tool. *Journal of Computational Science Education*, *3*(1), 2–10. https://doi.org/10.22369/issn.2153-4136/3/1/1

Mansbach, R., Ferguson, A., Kilian, K., Krogstad, J., Leal, C., Schleife, A., Trinkle, D. R., & Herman, G. L. (2016). Reforming an undergraduate materials science curriculum with computational models. *Journal of Materials Education*, *38*(3–4), 161–174.

Markauskaite, L., Freebody, P., & Irwin, J. (2011). Bridging and blending disciplines of inquiry: Doing science and changing practice and policy. In L. Markauskaite, P. Freebody, & J. Irwin (Eds.), *Methodological choice and design: Scholarship, policy and practice in social and educational research* (pp. 3–16). Springer. https://doi.org/10.1007/978-90-481-8933-5_1

Mayer, R. E. (2009). *Multimedia Learning*. Cambridge University Press.

McKenna, A. F. (2007). An investigation of adaptive expertise and transfer of design process knowledge. *Journal of Mechanical Design*, *129*(7), 730–734. https://doi.org/10.1115/1.2722316

McKenna, A. F., & Carberry, A. R. (2012). Characterizing the role of modeling in innovation. *International Journal of Engineering Education*, *28*(2), 263–269.

Melero, J., Hernández-Leo, D., & Blat, J. (2012). A review of constructivist learning methods with supporting tooling in ict higher education: Defining different types of scaffolding. *Journal of Universal Computer Science*, *18*(16), 2334–2360. https://doi.org/10.3217/jucs-018-16-2334

Miskovic, M., & Hoop, K. (2006). Action research meets critical pedagogy: Theory, practice, and reflection. *Qualitative Inquiry*, *12*(2), 269–291. https://doi.org/10.1177/1077800405284367

Moore, T. J., Guzey, S. S., Roehrig, G. H., Stohlmann, M. S., Park, M. S., Kim, Y. R., Callender, H. L., & Teo, H. J. (2015). Changes in Faculty Members' Instructional Beliefs while Implementing Model-Eliciting Activities. *Journal of Engineering Education*, *104*(3), 279–302.

Moore, T. J., Miller, R. L., Lesh, R. A., Stohlmann, M. S., & Kim, Y. R. (2013). Modeling in engineering: The role of representational fluency in students' conceptual understanding. *Journal of Engineering Education*, *102*(1), 141–178. https://doi.org/10.1002/jee.20004

Moravcsik, A. (2014). Transparency: The revolution in qualitative research. *PS - Political Science and Politics, 47*(1), 48–53. https://doi.org/10.1017/S1049096513001789

Moreno, R., Reisslein, M., & Ozogul, G. (2009). Optimizing worked-example instruction in electrical engineering: The Role of fading and feedback during problem-solving practice. *Journal of Engineering Education*, *98*(1), 83–92. https://doi.org/10.1002/j.2168-9830.2009.tb01007.x

National Research Council. (2011). *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. National Academies Press. https://doi.org/10.17226/13170

National Research Council. (2012). *Discipline-Based Education Research: Understanding and Improving Learning in Undergraduate Science and Engineering*. https://doi.org/10.17226/13362

National Science Board. (2018). *Science and Engineering Indicators*. https://www.nsf.gov/statistics/2018/nsb20181/report

Nersessian, N. J. (1999). Model-based reasoning in conceptual change. In L. Magnani, N. J. Nersessian, & P. Thagard (Eds.), *Model-based reasoning in scientific discovery* (pp. 5–22). Springer. https://doi.org/10.1287/inte.19.3.58

Nersessian, N. J. (2002). The cognitive basis of model-based reasoning in science. In P. Carruthers, S. Stich, & M. Siegal (Eds.), *The cognitive basis of science* (pp. 133–153). Cambridge University Press. https://doi.org/10.1017/CBO9780511613517.008

Nersessian, N. J. (2007). Model-Based Reasoning in Distributed Cognitive Systems. *Philosophy of Science, 73*(5), 699–709. https://doi.org/10.1086/518771

Neuendorf, K. A. (2019). Content analysis and thematic analysis. In *Research methods for applied psychologists: Design, analysis and reporting* (pp. 211–223). Routledge. https://doi.org/10.4324/9781315517971

Nowell, L. S., Norris, J. M., White, D. E., & Moules, N. J. (2017). Thematic analysis: Striving to meet the trustworthiness criteria. *International Journal of Qualitative Methods*, *16*(1), 1–13. https://doi.org/10.1177/1609406917733847

Onwuegbuzie, A. J., & Leech, N. L. (2007). Validity and qualitative research: An oxymoron? *Quality and Quantity, 41*(2), 233–249. https://doi.org/10.1007/s11135-006-9000-3

Ortega-Alvarez, J. D., Sanchez, W., & Magana, A. J. (2018). Exploring Undergraduate Students' Computational Modeling Abilities and Conceptual Understanding of Electric Circuits. *IEEE Transactions on Education*, *61*(3), 204–213. https://doi.org/10.1109/TE.2018.2822245

Ortiz-Rodriguez, E., Vazquez-Arenas, J., & Ricardez-Sandoval, L. A. (2010). An Undergraduate Course in Modeling and Simulation of Multiphysics Systems. *Chemical Engineering Education*, *44*(4), 299–305. https://doi.org/10.1115/1.2031269

Osterhage, J. L., Usher, E. L., Douin, T. A., & Bailey, W. M. (2019). Opportunities for self-evaluation increase student calibration in an introductory biology course. *CBE Life Sciences Education*, *18*(2), 1–10. https://doi.org/10.1187/cbe.18-10-0202

Papert, S. (1980). *MINDSTORMS: Children, Computers, and Powerful Ideas* (2nd ed.). HarperCollins.

Pawley, A. L. (2017). Shifting the "Default": The Case for Making Diversity the Expected Condition for Engineering Education and Making Whiteness and Maleness Visible. *Journal of Engineering Education*, *106*(4), 531–533. https://doi.org/10.1002/jee.20181

Peteranetz, M. S., Flanigan, A. E., Shell, D. F., & Soh, L. K. (2017). Computational Creativity Exercises: An Avenue for Promoting Learning in Computer Science. *IEEE Transactions on Education, 60*(4), 305–313. https://doi.org/10.1109/TE.2017.2705152

Pirnay-Dummer, P., Ifenthaler, D., & Seel, N. M. (2012). Designing model-based learning environments to support mental models for learning. In D. H. Jonassen & S. Land (Eds.), *Theoretical Foundations of Learning Environments* (2nd ed., pp. 66–94). Taylor & Francis.

President's Information Technology Advisory Committee. (2005). *Computational science: Ensuring America's competitiveness*. https://www.nitrd.gov/pitac/reports/20050609

Quillin, K., & Thomas, S. (2015). Drawing-to-learn: A framework for using drawings to promote model-based reasoning in biology. *CBE Life Sciences Education*, *14*(1), 1–16. https://doi.org/10.1187/cbe.14-08-0128

Raghavan, K., & Glaser, R. (1995). Model–based analysis and reasoning in science: The MARS curriculum. *Science Education*, *79*(1), 37–61. https://doi.org/10.1002/sce.3730790104

Rehmat, A. P., Ehsan, H., & Cardella, M. E. (2020). Instructional strategies to promote computational thinking for young learners. *Journal of Digital Learning in Teacher Education*, *36*(1), 46–62. https://doi.org/10.1080/21532974.2019.1693942

Reimann, P. (2011). Design-based research. In L. Markauskaite, P. Freebody, & J. Irwin (Eds.), *Methodological choice and design: Scholarship, policy and practice in social and educational research* (pp. 37–50). Springer. https://doi.org/10.1007/978-90-481-8933-5_3

Reynante, B. M., Selbach-Allen, M. E., & Pimentel, D. R. (2020). Exploring the Promises and Perils of Integrated STEM Through Disciplinary Practices and Epistemologies. *Science and Education*. https://doi.org/10.1007/s11191-020-00121-x

Rich, S. H., & Venkatasubramanian, V. (1987). Model-based reasoning in diagnostic expert systems for chemical process plants. *Computers and Chemical Engineering*, *11*(2), 111–122. https://doi.org/10.1016/0098-1354(87)80012-1

Rittle-Johnson, B., & Star, J. R. (2007). Does Comparing Solution Methods Facilitate Conceptual and Procedural Knowledge? An Experimental Study on Learning to Solve Equations. *Journal of Educational Psychology*, *99*(3), 561–574. https://doi.org/10.1037/0022-0663.99.3.561

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*(1). https://doi.org/10.1186/s41239-017-0080-z

Sale, J. E. M., & Brazil, K. (2002). Revisint the Quanitative-Qualitative debate: Implications for mixed-methods research. *Quality & Quantity*, *36*, 43–53. https://doi.org/10.1023/A:1014301607592

Sandoval, W. (2014). Conjecture Mapping: An Approach to Systematic Educational Design Research. *Journal of the Learning Sciences*, *23*(1), 18–36. https://doi.org/10.1080/10508406.2013.778204

Sanford, J. F., & Naidu, J. T. (2017). Mathematical modeling and computational thinking. *Contemporary Issues in Education Research*, *10*(2), 159–168. https://doi.org/10.19030/cier.v10i2.9925

Sawyer, R. K., & Greeno, J. (2009). Situativity and learning. In P. Robbins & M. Aydede (Eds.), *The Cambridge Handbook of Situated Cognition* (pp. 347–367). Cambridge University Press. https://doi.org/10.1017/CBO9780511816826

Schefer-Wenzl, S., & Miladinovic, I. (2019). Developing Complex Problem-Solving Skills: An Engineering Perspective. *International Journal of Advanced Corporate Learning (IJAC)*, *12*(3), 82. https://doi.org/10.3991/ijac.v12i3.11067

Schunk, D. H. (2003). Self-efficacy for reading and writing: Influence of modeling, goal setting, and self-evaluation. *Reading and Writing Quarterly*, *19*(2), 159–172. https://doi.org/10.1080/10573560308219

Schunk, D. H., & Ertmer, P. A. (2000). Self-Regulation and Academic Learning: Self-Efficacy Enahancing Interventions. In *Handbook of Self-Regulation* (pp. 631–649). Academic Press.

Schwartz, D. L., & Bransford, J. D. (1998). A time for telling. *Cognition and Instruction*, *16*(4), 475–522.

Sclarsky, E., Kadlowec, J., & Vernengo, A. J. (2016). Modeling stress relaxation of crosslinked polymer networks for biomaterials applications: A distance learning module. *Education for Chemical Engineers*, *17*, 14–20. https://doi.org/10.1016/j.ece.2016.05.003

Seel, N. M. (2003). Model-centered learning and instruction. *Technology, Instruction, Cognition and Learning*, *1*(1), 59–85.

Selby, C. C. (2015). Relationships: Computational Thinking, Pedagogy of Programming, and Bloom's Taxonomy. *Proceedings of the Workshop in Primary and Secondary Computing Education*, 80–87. https://doi.org/10.1145/2818314.2818315

Selby, C. C., & Woollard, J. (2013). Computational Thinking : The Developing Definition. *Proceedings of the 2014 Conference of the Special Interest Group on Computer Science Education (SIGCSE)*, 5–8.

Self, B. P., Miller, R. L., Kean, A., Moore, T. J., Ogletree, T., & Schreiber, F. (2008). Important student misconceptions in mechanics and thermal science: Identification using model-eliciting activities. *Proceedings of Frontiers in Education Conference*. https://doi.org/10.1109/FIE.2008.4720595

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351–380. https://doi.org/10.1007/s10639-012-9240-x

Shiflet, A. B., & Shiflet, G. W. (2006). *Introduction to Computational Science*. Princeton University Press. https://doi.org/10.1017/CBO9781107415324.004

Shiflet, A. B., & Shiflet, G. W. (2014). *Introduction to computational science: Modeling and simulation for the sciences*. Princton University.

Shin, N., Jonassen, D. H., & McGee, S. (2003). Predictors of well-structured and ill-structured problem solving in an astronomy simulation. *Journal of Research in Science Teaching*, *40*(1), 6–33. https://doi.org/10.1002/tea.10058

Skuse, B. (2019). The third pillar. *Physics World*, *32*(3), 40–43. https://doi.org/10.1088/2058-7058/32/3/33

Stroulia, E., Shankar, M., Goel, A. K., & Penberthy, L. (1992). A Model-Based Approach to Blame-Assignment in Design. *Second International Conference on AI in Design*, 1–15.

Taylor, I., Barker, M., & Jones, A. (2003). Promoting mental model building in astronomy education. *International Journal of Science Education*, *25*(10), 1025–1225. https://doi.org/10.1080/0950069022000017270a

Tellis, W. M. (1997). Application of a Case Study Methodology. *The Qualitative Report*, *3*(3), 1–19.

The Royal Society. (2012). *Shut down or restart? The way forward for computing in UK schools* (Issue January).

Tillman, D. (2013). *Implications of problem based learning (PBL) in elementary schools upon the K-12 engineering education pipeline BT - 120th ASEE Annual Conference and Exposition, June 23, 2013 - June 26, 2013*.

United States Office of the Press Secretary. (2016). *Fact Sheet: President Obama Announces Computer Science for All Initiative*. https://obamawhitehouse.archives.gov/the-press-office/2016/01/30/fact-sheet-president-obama-announces-computer-science-all-initiative-0

Urban, K., & Urban, M. (2019). Improving the Accuracy of the Self-Evaluation During on-Screen Self-Regulated Learning Through Calibration Feedback. *INTED2019 Proceedings*, *1*(March), 9002–9007. https://doi.org/10.21125/inted.2019.2239

Vaismoradi, M., Turunen, H., & Bondas, T. (2013). Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study. In *Nursing and Health Sciences*. https://doi.org/10.1111/nhs.12048

Vandierendonck, A. (2002). Evidence for Mental-model-based Reasoning: A Comparison of Reasoning with Time and Space Concepts. *Thinking & Reasoning*, *2*(4), 249–272. https://doi.org/10.1080/135467896394438

Vieira, C., Magana, A. J., Falk, M. L., & Garcia, R. E. (2017). Writing in-code comments to self-explain in computational science and engineering education. *ACM Transactions on Computing Education*, *17*(4), 1–21. https://doi.org/10.1145/3058751

Vieira, C., Magana, A. J., García, R. E., Jana, A., & Krafcik, M. (2018). Integrating computational science tools into a thermodynamics course. *Journal of Science Education and Technology*. https://doi.org/10.1007/s10956-017-9726-9

Vieira, C., Magana, A. J., Roy, A., & Falk, M. L. (2019). Student Explanations in the Context of Computational Science and Engineering Education. *Cognition and Instruction*, *37*(2), 201–231. https://doi.org/10.1080/07370008.2018.1539738

Vieira, C., Magana, A. J., Roy, A., Falk, M. L., & Reese, M. J. (2016). Exploring undergraduate students' computational literacy in the context of problem solving. *Computers in Education Journal*, *7*(1), 100–112.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes* (M. Cole, V. John-Steiner, S. Scribner, & E. Souberman (eds.)). Harvard University Press.

Wang, F., & Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research & Development*, *53*(4), 5–23. https://doi.org/10.1007/BF02504682

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2011). Research notebook: Computational thinking—What and why? *The Link Magazine*, June 23, 2015. http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Yan, J., Vieira, C., & Magana, A. (2015). Exploring Design Characteristics of Worked Examples to Support Programming and Algorithm Design. *The Journal of Computational Science Education*, *6*(1), 2–15. https://doi.org/10.22369/issn.2153-4136/6/1/1

Yin, R. K. (1981). The case study crisis : Some answers. *Administrative Science Quarterly*, *26*(1), 58–65.

Yin, R. K. (1994). *Case Study Reserach - Design and Methods* (2nd ed.). Sage.

Yin, R. K. (2009). *Case study research: Design & methods* (4th ed.). Sage.

Yin, Y., Hadad, R., Tang, X., & Lin, Q. (2020). Improving and Assessing Computational Thinking in Maker Activities: the Integration with Physics and Engineering Learning (Journal of Science Education and Technology, (2020), 29, 2, (189-214), 10.1007/s10956-019-09794-8). *Journal of Science Education and Technology*, *29*(2), 215. https://doi.org/10.1007/s10956-020-09822-y

Yuen, T. T., & Robbins, K. A. (2014). A Qualitative Study of Students' Computational Thinking Skills in a Data-Driven Computing Class. *ACM Transactions on Computing Education*, *14*(4), 1–19. https://doi.org/10.1145/2676660

Zawojewski, J. S., Diefes-Dux, H. A., & Bowman, K. J. (2008). Models and modeling in engineering education: Designing experiences for all students. In *Models and Modeling in Engineering Education*. Sense. https://doi.org/10.1163/9789087904043

APPENDIX A. PLANNING THE MODEL TEMPLATE

Name:

In-class activity 1 (Individual working w/ group): Prepping the Model (due end of class 9/6)		
What properties are needed/not needed for the model? For each property of the food, give reasoning as to why.		
Food Property:	Why Is/Is Not Used:	
For each property where a range or raw data is given, how do you intend to address this in the final model? For each property listed please give reasoning as to why you are choosing your current strategy.		
Food Property:	How will you use or address? Why?	
Are there any properties of the food that are needed that are not given in the problem statement? If any, please justify why it is needed and what source you will obtain it from.		
Food property Needed:	Why is it needed?	Why did you use the source you did?
What assumptions will you make to solve the problem? For each assumption explain why you made the assumption and what it may limit about your model.		
Assumption(s):	Why did you choose?	What will this limit?
The mathematical equations necessary to solve the problem. For each equation please explain why it was chosen and any assumptions your model will make about the equations. Please list all equations necessary. Feel free to use the course textbook or online materials.		
Equation Needed:	Why is this needed?	What assumptions does this equation make?
What computational technique will you use to solve the system? Explain why this technique was chosen, what the benefits are, and what the limitations are. (For example, implicit finite difference, explicit finite difference, finite element method, Crank-Nicolson method, Monte Carlo method, etc).		
Computational technique chosen: What are the benefits of this technique?		

What are the limitations of this technique? Why did you choose this technique over alternatives?
Please show how you intend to combine the properties, equations, assumptions, and numerical techniques in your final solution (provide a "roadmap" of how you believe you will solve the problem). Feel free to include diagrams, drawings, or maps.

APPENDIX B. BUILDING THE MODEL TEMPLATE

Name:

Take home assignment (Individual): Building the model (Due beginning of class 9/27)		
Please outline and describe how your model works in terms of computational structures. For each structure, please explain why the programming technique or process used was chosen. Include as much detail as possible, doing this for each computational structure within your model (groups of variables, loops, conditional statements, sets of equations etc).		
Computational Structure: Ex. Nested for loop in lines 15-25 of function_x. Ex. Define thermal variables in line 26-33.	How does it work? Why was it programmed this way? Ex. The nested for loop indexes through both rows and columns to move through both time and space. It was used because there was a set end point, thus more efficient than a while loop that exits upon an unknown number of iterations. This is useful given it is unknown how long heat will take to transfer. Ex. These statements define how heat is moving through the material. These variables are necessary to be defined previous to the equations in line 45 as they are used there. Variable X does….and interacts with Variable Y in this way…	
Please describe any assumptions made during the modeling process and why those may have been good or appropriate assumptions?		
Assumption(s):	Why did you make this assumption?	How does this impact how your model works?
What process parameters are you using for each of the food materials?		
i. **What microorganism is your program targeting? All of them? Only one?** **Why?**		

APPENDIX C. EVALUATING THE MODEL TEMPLATE

Name:

In class activity (Individual with group): Evaluate your model (due end of class 9/20)
Questions to discuss during group rotation meetings. For these meetings focus on HOW and WHY you solved and programmed the problem the way you did.
1. What are different assumptions that you made about the physical properties of the system? Did you use different data? How would these differences impact the model?
Notes:
2. Do a line-by-line comparison with the other individuals programming files. How did your programming strategies differ? What advantages do you see in how they did their model? What advantages do you see in your own?
Notes:

APPENDIX D. REFLECTING ON THE MODEL TEMPLATE

Name:

Take-home assignment (Individual): Reflect on your model (due beginning of class 9/25)		
What approaches did other students take with respect to the data that they used (justifications, assumptions, and limitations) and the way they programmed their model? Be as detailed as possible in listing various differences between models. For each difference talk about WHY you think the other group chose to do it the way they did. Be detailed.		
How did these differ from your own approach? When would your own approach make the most sense? When would different assumptions that other groups made make the most sense?		
Differences I saw:	What approach makes the most sense:	Why the approach makes the most sense:
If you were to do this assignment again what are different assumptions you would make and what do you believe to be the optimal solution to the problem?		
Things I would do differently:	Why I would do them differently:	
What was the most challenging piece of this assignment?		
Why do you feel that was the most challenging?		
How did you overcome this challenge?		

APPENDIX E. PROGRAMMING FILE TEMPLATE

```
%Name ___________________

%---------------------------------------------------------------------
%USE THIS TEMPLATE FOR EACH SCRIPT/FUNCTION THAT YOU WRITE
%For each block of code (5-10 lines) in the model provide the following:
%       HOW: How does this block of code work?
%       WHAT: What does the block of code represent?
%       WHY: Why is this block of code included in the model?
%For more details, an example of good commenting and bad commenting has
%been uploaded to Blackboard.
% --------------------------------------------------------------------

%---------------Thermal/Physical Properties of the System -----------------
%Here place any/all thermal, physical, or kinetic properties of the system.
%For each block make sure to answer the HOW/WHAT/WHY questions above.

%-------------------Numerical/Analytical Calculations ---------------------
%Here place any calculations, mathematical structures, looping structures,
%etc that are necessary for the modeling activity. For block of code or
%structure make sure to answer the HOW/WHAT/WHY questions above.

%------------------------- Graphical/Numerical Output --------------------
%Any output parameters should go here such as display of graphs, matrices,
%values, variables, or tables. For each block of code make sure to answer the
%HOW/WHAT/WHY questions above.
```

APPENDIX F. FREQUENCY OF COMPUTATIONAL THINKING OUTCOMES ACROSS ANALYZED ARTIFACTS

CT Practice	CT Outcome	**Frequency of artifacts** Building (n=15)	Planning (n=19)	Evaluating/Reflecting (n=34)
Abstraction	· Ranges to values	11	19	21
	· Multiple to single	13	17	27
	· Dynamic to static	11	14	20
	· Geometric relationships	NA	5	9
	· Similar systems	NA	4	11
	· Infinite to finite	10	2	13
Algorithmic Thinking	· Indication of later use/effect	13	19	29
	· Stepwise approach	NA	17	6
	· Parallel methods	NA	4	28
	· Conditional logic	3	3	5
Evaluation	· Solution accuracy	12	17	30
	· Solution complexity	10	12	26
	· Time efficiency	7	11	8
	· Design criteria	3	10	10
	· Solution usability	6	2	16
	· Solution flexibility	3	1	13
	· Ideal Solution	NA	NA	2
	· Peer Comparison	NA	NA	16
	· Debugging	5	0	15
Generalization	· Previous coursework or experience	NA	8	5
	· External applications	NA	6	4
	· Similar problems	NA	NA	5
	· Reuse of code/equations	7	0	5
Decomposition	· Organization of larger solution method	3	9	3
	· Allocation of resources	NA	1	1
	· Ease of use of program	8	0	17