Characterizing Computational Thinking in the Context of Model-Planning Activities

Joseph A. Lyon 1,*, Alejandra J. Magana 2 and Ruth A. Streveler 3

1 Engineering Honors Program, Purdue University, West Lafayette, IN 47909, USA
2 Polytechnic Institute, Purdue University, West Lafayette, IN 47909, USA; admagana@purdue.edu
3 School of Engineering Education, Purdue University, West Lafayette, IN 47909, USA; rastreve@purdue.edu
* Correspondence: lyonj@purdue.edu

Abstract: Computational thinking (CT) is a critical skill needed for STEM professionals and educational interventions that emphasize CT are needed. In engineering, one potential pedagogical tool to build CT is modeling, an essential skill for engineering students where they apply their scientific knowledge to real-world problems involving planning, building, evaluating, and reflecting on created systems to simulate the real world. However, in-depth studies of how modeling is done in the class in relation to CT are limited. We used a case study methodology to evaluate a model-planning activity in a final-year undergraduate engineering classroom to elicit CT practices in students as they planned their modeling approach. Thematic analysis was 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

1. 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 [1,2]. This is partly because it has been challenging to assign departments within universities to teach computational and data science skills to undergraduates [1]. This issue is amplified by instructors who may not feel comfortable teaching these topics and expect students to learn these skills in other courses [3]. The net result is university graduates who cannot think through complex computational and data science problems. In order to address this critical educational gap, we need to more explicitly understand the specifics of students’ thinking as they develop CT skills.

These issues are especially apparent within engineering curricula, which are already perceived to be packed with courses by many engineering faculty [3]. 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 [4–7]. 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 [6]. However, few studies have tracked the emergence of CT through modeling processes in a classroom.

Models allow students to apply mathematical and computational concepts to solve real-world problems, and they have been heavily studied in engineering [8–11]. Often the modeling process is divided into multiple phases, including planning, building, evaluating,
and revising [6,12–14]. 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 [6]. For this work, our research question is: What computational thinking practices emerged when students participated in model-planning activities? The question was important because it filled research gaps around further defining CT, showing explicit examples of CT in the classroom, and making explicit connections between the modeling process and CT.

2. 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.

2.1. Computational Thinking (CT)

In recent years, computational thinking (CT) has been of particular interest to researchers and educators alike [15,16]. CT is essentially an umbrella term for different common thinking practices derived from computational sciences [17]. 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 [15,18–20].

Additionally, studies incorporating CT in the classroom have been surpassed mainly by researchers discussing CT in more general and definitional ways [16,21,22]. When operationalized, it is often in K12 settings rather than the undergraduate level [23,24]. This K12 literature base has covered a host of topics such as the definition of CT [25], how to assess CT [26,27], and how to teach CT [28]. However, this focus on K12 has left a significant gap in the literature for concrete implementation strategies of CT or best practices for developing CT in undergraduate educational settings. Specifically, what exactly does CT look like when it is used by students and what are ways of operationalizing it in the classroom? There are few studies that delve into the specifics of computational thinking, opting for more definitional approaches [16,21,22] and when implemented, studies often use computational thinking to structure course content instead of finding concrete evidence or emergent practices of CT in students’ work [16]. This study addresses the gaps of operationalizing CT and connecting it to classroom practice by focusing on evidence-based emergent CT practices using a defined CT framework [18,20].

2.2. 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 [29]. Students’ additional considerations while planning should be time constraints and meeting other criteria of the given problem [30]. Planning a solution pathway is a cognitive skill that plays a vital role in students’ final solution [31]. 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 [32]. Thus, planning the model before solving it 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 [12]. Within modeling interventions, students
who plan their solution before starting often exhibit behaviors more aligned with subject-matter experts, focusing on their conceptual understanding of the material and avoiding trial-and-error solutions [33].

2.3. 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 [34–37]. MEAs help students learn complex problem-solving, mathematical modeling, and team skills. The activities are built on six core principles [34]: (1) the model-construction principle; (2) the reality principle; (3) the self-assessment principle; (4) the model-documentation principle; (5) the generalizability principle; and (6) the 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 [8,35,38,39]. For these reasons, the MEA framework was used to structure the modeling activities for this study.

3. Methods

We used a specific case and the MEA framework to design the learning intervention [34]. Classroom artifacts were then analyzed using thematic analysis to identify computational thinking themes throughout the assignments.

3.1. Participants and Context

Participants (N = 26) in 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 as male. 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.

3.2. 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 [8]. Students were divided into teams of three or four to plan out how they would model the process. Nearly all groups consisted of four team members, with only a few consisting of three. The students were able to choose their own groups, meaning no background or demographic information was used to create the student groupings. There were 15 total groups in the class. 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 that 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 [40].

3.3. 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 (which were collected from each of the teams at the end of the lab period). One researcher transcribed audio/video data of the group planning process to transform it into a format suitable for thematic analysis. Table 1 shows the data collection and the total participants from the study.

Table 1. Data overview for the current study.

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

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

3.4. Data Analysis

We used thematic analysis and followed the methodology as defined by Braun and Clarke [41] (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 2, with the codebook being used and adapted from previous studies [6,20].

Table 2. Practices within computational thinking (CT) and definitions from the literature [6].

<table>
<thead>
<tr>
<th>Computational Thinking Practice</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>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. 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.</td>
</tr>
<tr>
<td>Algorithmic Thinking</td>
<td>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.</td>
</tr>
<tr>
<td>Evaluation</td>
<td>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.</td>
</tr>
<tr>
<td>Generalization</td>
<td>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.</td>
</tr>
<tr>
<td>Decomposition</td>
<td></td>
</tr>
</tbody>
</table>

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 [6] were used, with the entire codebook used as a starting point. 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. Thus, some of the definitions and subthemes are the same or overlap significantly with the original codebook [6].

3.5. 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.

4. Results

The results section is first broken down into a section that overviews the outcomes seen across the dataset. From there, each section’s outcomes one of the CT practices with an in depth look at the outcomes identified.
4.1. Overview of the Results

Many of the CT outcomes and definitions identified were similar or the same as the previous study [6]. However, some key differences to the codebook appear in bold text in Table 3.

**Table 3.** Finalized codebook of CT practices for model-planning.

<table>
<thead>
<tr>
<th>Abstraction</th>
<th>Algorithmic Thinking</th>
<th>Evaluation</th>
<th>Generalization</th>
<th>Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranges to values</td>
<td>Multiple to single</td>
<td>Dynamic to static</td>
<td>Geometric relationships</td>
<td>Similar systems</td>
</tr>
<tr>
<td>Indication of later use/effect</td>
<td>Stepwise approach</td>
<td>Parallel methods</td>
<td>Conditional logic</td>
<td></td>
</tr>
<tr>
<td>Solution accuracy</td>
<td>Solution complexity</td>
<td>Time efficiency</td>
<td>Design criteria</td>
<td>Solution usability</td>
</tr>
<tr>
<td>Previous coursework or experience</td>
<td>External applications</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Organization of larger solution method</td>
<td>Allocation of resources</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* bold denotes a newly identified CT outcome.

As one can see from Table 3, 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 [6] were not identified in the model planning data.

4.2. Abstraction

From previous work, *abstraction* was one of the most commonly identified CT practices in the model-building process [6]. 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 4 below shows the finalized definitions and representative quotes of each CT outcome identified from the model-planning data.

**Table 4.** Definitions and quotes of CT outcomes identified for *abstraction* (letters denote individual deidentified students).

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Definition</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranges to values</td>
<td>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).</td>
<td>“So z-value is the number of degrees that requires that variation. So I feel like we should choose the highest one again.”</td>
</tr>
<tr>
<td>Multiple to single</td>
<td>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.</td>
<td>“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.”</td>
</tr>
<tr>
<td>Dynamic to static</td>
<td>Making factors or variables constant, uniform, or unchanging that would normally vary in respect to other variables or aspects of the problem.</td>
<td>“D: Density, like constant pressure means you have a density that’s not relying on pressure. It’s more relying on the temperature.”</td>
</tr>
<tr>
<td>Geometric relationships</td>
<td>Simplifying problems by assuming a geometric characteristic or relationships between variables, constants, or factors within the problem space.</td>
<td>“D: Yeah so we will have to assume. uh C: Cylindrical. D: Cylindrical yeah. So we’ll get.”</td>
</tr>
<tr>
<td>Similar systems</td>
<td>Aligning properties of a current unknown system or property of a system with knowns of a system that is well known or familiar.</td>
<td>“B: What assumptions will you make to solve the problem? C: Assume, pumpkin pie filling is about like applesauce.”</td>
</tr>
<tr>
<td>Infinite to finite</td>
<td>Making aspects of the problem or system that are infinite or continuous and making them finite or discrete in nature.</td>
<td>“But uh depending on our can dimensions, we could do the whole like infinity, can split by infinity slabs, or just infinite cylinder.”</td>
</tr>
</tbody>
</table>
There were two new codes for abstraction found during the thematic analysis process. One is about aligning properties with similar systems. It 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 is 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 significantly important to the model-building process.

### 4.3. Algorithmic Thinking

Algorithmic thinking was seen throughout the dataset as a common CT practice, with multiple observed outcomes. Table 5 shows the different CT outcomes observed in the data from the practice of algorithmic thinking, with both definitions and representative quotes.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Definition</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indication of later use/effect</td>
<td>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.</td>
<td>“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 $c_p$”</td>
</tr>
<tr>
<td>Using a stepwise approach</td>
<td>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).</td>
<td>“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.”</td>
</tr>
<tr>
<td>Considering parallel methods</td>
<td>Indication that there are parallel or redundant solution methods or information given.</td>
<td>“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 $E_a$.”</td>
</tr>
<tr>
<td>Conditional logic</td>
<td>Student making an if/else statement or conditional statement based on previous or future conditions (i.e., if I do x, y will happen)</td>
<td>“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.”</td>
</tr>
</tbody>
</table>

As shown in Table 5, there are two new subthemes identified. The first, named stepwise approach, is where students outline a solution process step by step by diagramming or 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 1
below shows a diagram submitted as part of a student report that shows this stepwise approach code.

![Diagram](image)

**Figure 1.** 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.

### 4.4. 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 6 outlines the definitions and quotes associated with each observed outcome.

**Table 6.** Definitions and quotes of CT outcomes identified for *evaluation* (letters denote individual deidentified students).

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

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.

4.5. 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 was missing from the original codebook. Table 7 outlines the definitions of the identified outcomes and representative quotes.

Table 7. Definitions and quotes of CT outcomes identified for generalization (letters denote individual deidentified students).

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Definition</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drawing from previous experiences</td>
<td>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.</td>
<td>“C: Oh I remember this equation from [another course], the d equals d₀ times ... B: Oh yeah C: or [another course] I don’t know, d equals d₀ times ten to the t₀ minus t over z.”</td>
</tr>
<tr>
<td>Projecting to other or future applications</td>
<td>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.</td>
<td>“So the model will not account for really hot or really cold days.”</td>
</tr>
</tbody>
</table>

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 affect the future applications that their model could have.

4.6. 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 8 shows the different outcomes, definitions, and quotes identified.

Table 8. Definitions and quotes of CT outcomes identified for decomposition (letters denote individual deidentified students).

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Definition</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization of larger solution method</td>
<td>Decomposing the code or solution method as an organizational tool.</td>
<td>“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.”</td>
</tr>
<tr>
<td>Allocating resources</td>
<td>Breaking up a problem amongst group members in order to reduce complexity of the work required or to increase quality of the solution method</td>
<td>See Figure 2</td>
</tr>
</tbody>
</table>
Table 8. Definitions and quotes of CT outcomes identified for decomposition (letters denote individuated students).

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Definition</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization of larger solution method</td>
<td>Decomposing the code or solution method as an organizational tool.</td>
<td>See Figure 2</td>
</tr>
<tr>
<td>Allocating resources</td>
<td>Breaking up a problem amongst group members in order to reduce complexity of work required or to increase quality of the solution method</td>
<td>“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.”</td>
</tr>
</tbody>
</table>

Figure 2. Example diagram demonstrating organization of larger solution method.

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 2 shows an example of this coded behavior.

Diagrams such as Figures 1 and 2 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.

5. Discussion

The 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 [19], 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 [3,19,42]. 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 [6]. 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 [3] or their lack of self-efficacy with making programming or computing decisions [43], [44]. 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 [10,12,45]. The planning phase is where students are “breaking down the phenomenon under study into small pieces that can be incorporated into an external model” [46] (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 [47]. Yet, the outcome seems to go one step further: students are transferring information not into the same context but to new contexts. This type of knowledge transfer of information to new contexts is indicative of expert-like practice [48,49].

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 [50,51]. 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 [33].

Both new generalization themes seem to be about transferring information from previous situations and future situations indicative of expert-like practice [52,53]. The literature
around adaptive expertise has shown that transferring information into new and novel situations is one way experts are differentiated from novices [48,49,54]. 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.

5.1. 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 [33,55,56], 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 [33]. 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 [57,58].

5.2. Limitations and Future Work

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 transferability 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. This may be a critical limitation because the mental processes underlying CT elements like abstraction may occur at an implicit level and thus may not be articulated easily. 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 in order to track CT across the entire modeling cycle. 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 and thus could improve the design of learning activities designed to elicit CT.

6. Conclusions

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 but is understudied within a naturalistic classroom setting. We studied advanced engineering 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.

**Author Contributions:** Conceptualization, J.A.L. and A.J.M.; methodology, J.A.L. and A.J.M.; validation, J.A.L.; formal analysis, J.A.L.; investigation, J.A.L. and A.J.M.; funding acquisition, J.A.L. and A.J.M.; writing—original draft preparation, J.A.L.; writing—review and editing, J.A.M. and R.A.S.; supervision, A.J.M. and R.A.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This material is based on work supported in part by the National Science Foundation Graduate Research Fellowship Program under Grant Number DGE-1842166 as well as the National Science Foundation under Award Number DBI-2120200. 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.

**Institutional Review Board Statement:** The study was approved by the Institutional Review Board of Purdue University (protocol number 1806020715).

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare that they have no conflict of interest.

**References**

2. Irizarry, R.A. The Role of Academia in Data Science Education. Harv. Data Sci. Rev. 2020, 2. [CrossRef]


47. Dewey, J. Experience and Education; Macmillan: New York, NY, USA, 1938. [CrossRef]

