Paper ID #19605

Capturing the Computational Thinking of Families with Young Children in Out-of-School Environments

Ms. Hoda Ehsan, Purdue University, West Lafayette (College of Engineering)

Hoda is a Ph.D. student in the School of Engineering Education, Purdue. She received her B.S. in mechanical engineering in Iran, and obtained her M.S. in Childhood Education and New York teaching certification from City College of New York (CUNY-CCNY). She is now a graduate research assistant on STEM+C project. Her research interests include designing informal setting for engineering learning, and promoting engineering thinking in differently abled students in informal and formal settings.

Dr. Monica E Cardella, Purdue University, West Lafayette (College of Engineering)

Monica E. Cardella is the Director of the INSPIRE Institute for Pre-College Engineering Education and is an Associate Professor of Engineering Education at Purdue University.

Capturing the Computational Thinking of Families with Young Children in Out-of-School Environments

Abstract

For the past two decades, researchers and educators have been interested in integrating engineering into K-12 learning experiences. More recently, computational thinking (CT) has gained increased attention in K-12 engineering education. Computational thinking is broader than programming and coding. Some describe computational thinking as crucial to engineering problem solving and critical to engineering habits of mind like systems thinking. However, few studies have explored how computational thinking is exhibited by children, and CT competencies for children have not been consistently defined. Hence developing and implementing effective CT-related activities for children can be difficult. Therefore, exploring what computational thinking looks like for children is critical.

Children can engage in, and learn to engage in computational thinking in both formal and informal settings. In this study, we are interested in exploring what computational thinking might look like in settings that approximate children's everyday experiences. More specifically, in order to investigate what computational thinking looks like when enacted by young children, we are interested in observing children and their family members engaging in open-ended engineering activities that are play-based. To accomplish this, we observed and video-recorded 5-8 year-old children and their families creating different structures together using large foam blocks that are out for free play at a science center. Based on our observations and analysis of the video-recordings, in this paper we report on the computational thinking practices and competencies children and families demonstrated while engaged in engineering play. Our findings can provide information needed to create a framework for promoting computational thinking in young children in informal settings.

Introduction

Computational thinking and engineering education

Many believe that there are overlaps between engineering and computational thinking (NRC, 2011; Wing 2006). From a curricular standpoint, undergraduate engineering students frequently learn programming and coding skills to use tools such as MATLAB in their work. Wing (2006) argues that computational thinking is more than just programming and coding, and is not thinking like computers. She believes computational thinking is the way that "humans, not computers, think (p.35)" which requires conceptualizing problems when solving them. She argues that computational thinking complements and combines engineering thinking and mathematical thinking (Wing, 2006). According to her, computational thinking draws on engineering thinking to solve problems and design systems that interact with humans and the real world. Interacting with the real world requires thinking about design criteria and constraints – such as safety and efficacy. Like engineers, computational thinkers involve in a process of problem-solving (Computer Science Teacher Association & International Society for Technology in Education, 2011).

Promoting problem solving and designing skills is a core focus of both undergraduate engineering education and pre-college engineering education. P-12 engineering education also has the potential to effectively impact student learning of different disciplines, increase technological literacy, and boost student interest towards engineering (National Academy of Engineering, 2009). In line with that, in 2009 the National Academy of Engineering (NAE 2009) released a document emphasizing three critical areas for pre-college engineering education: teaching engineering design, incorporating effective methods such as "computational methods" for developing different content knowledge skills, and promoting engineering habits of mind. Therefore, developing computational thinking is helpful in learning and strengthening pre-college engineering.

Additionally, due to the growth of engineering-related careers, many current K-12 students will end up working in fields that involve computing (Barr & Stephenson, 2011). At the same time, Barr and Stephenson also discuss that today's children will live in a life which is heavily influenced by computing and requires computational thinking. Therefore, today's children should develop computational thinking competences, learn to solve problems computationally and work with computational methods and tools early on. However, embedding computational thinking in pre-college education requires practical approaches based on operational definitions (Barr & Stephenson, 2011) suitable for the age of children. In order to develop suitable computational thinking definitions and take practical approaches, characterizing what computational thinking looks like in children is necessary.

Computational thinking in out-of-school environments

Families can play an important role in children's learning experiences, because children spend most of their time in out-of-school environments (Stevens & Bransford, 2007). These environments include everyday settings like family activities or in designed spaces like museums and science centers. Children are engaged in different activities with their families that may provide them a wealth of learning opportunities. Through these learning opportunities, children deeply engage in learning while interacting with family members (and others), build on their prior knowledge and interest, develop stronger thinking, and finally reflect on their learning experiences through sensemaking conversations with their families (Bell, Lewenstein, Shouse, & Feder, 2009). Therefore, family interactions can support children in learning and developing their thinking, skills and competencies.

While little research has studied learning engineering in out-of-school environments, some evidence from the literature has shown that engineering interest, engineering knowledge and engineering abilities may increase in these environments (Paulsen et al. 2015; Dorie et al. 2014; Kotys-Schwartz, Besterfield-Sacre, & Shuman, 2011). As previously mentioned, computational thinking has a strong connection with engineering and engineering thinking. Hence we believe we may be able to see development of computational thinking in Out-of-School environments.

Purpose of the Study

This study is part of an NSF-funded project (Hynes et al. 2016). One aspect of this project looks at K-2 students' computational thinking competencies in integrated STEM informal experiences. In earlier phases of the project, we conducted research to develop a set of definitions of CT competencies that can be observed when enacted by children (Dasgupta, Rynearson, Purzer, Ehsan, & Cardella, 2017; INSPIRE Definitions, 2017). For this study, we are focusing on seven of these competencies: Abstraction, Algorithms and Procedures, Debugging, Problem Decomposition, Parallelization, Pattern Recognition and Simulation. These competencies were then synthesized into three phases of an iterative computational thinking process consisting of (1) Problem Scoping, (2) Development, and (3) Implementing and Improving. Each of the phases involves multiple CT competencies as noted below:

Problem Scoping:

- Problem Decomposition
- Abstraction
- Pattern Recognition

Development:

- Algorithms & Procedures
- Parallelization
- Abstraction
- Pattern Recognition

Implementing & Improving:

- Troubleshooting/Debugging
- Problem Decomposition
- Simulation, Automation, Evaluation
- Pattern Recognition

For this study, our purpose was to investigate which CT competencies can be observed when children and their families are engaged in an engineering design task at a science center.

Methods

Study Procedure

The study was conducted at a science center in the Midwest. Families with K-2–aged children who visited the science center were invited to participate in the study. They were given an engineering design task, and were asked to build their solution using big foam blocks (see Table 1). The task (see Figure 1) was presented them on signs that served as a proxy for exhibit signage.

We also provided the adults (i.e. parents and other family members) with information about computational thinking by hanging signage at multiple locations in the big foam block exhibit space. The families were given 30 minutes to read the task, discuss it with their children, and develop their solutions. At the end, we interviewed both adults and children about their experiences during this task, and previous experiences that they considered to be similar to the activity.

Table 1. Pictures of Playgroud created by Families.

Playground 1	Playground 2	Playground 3

Data Sources

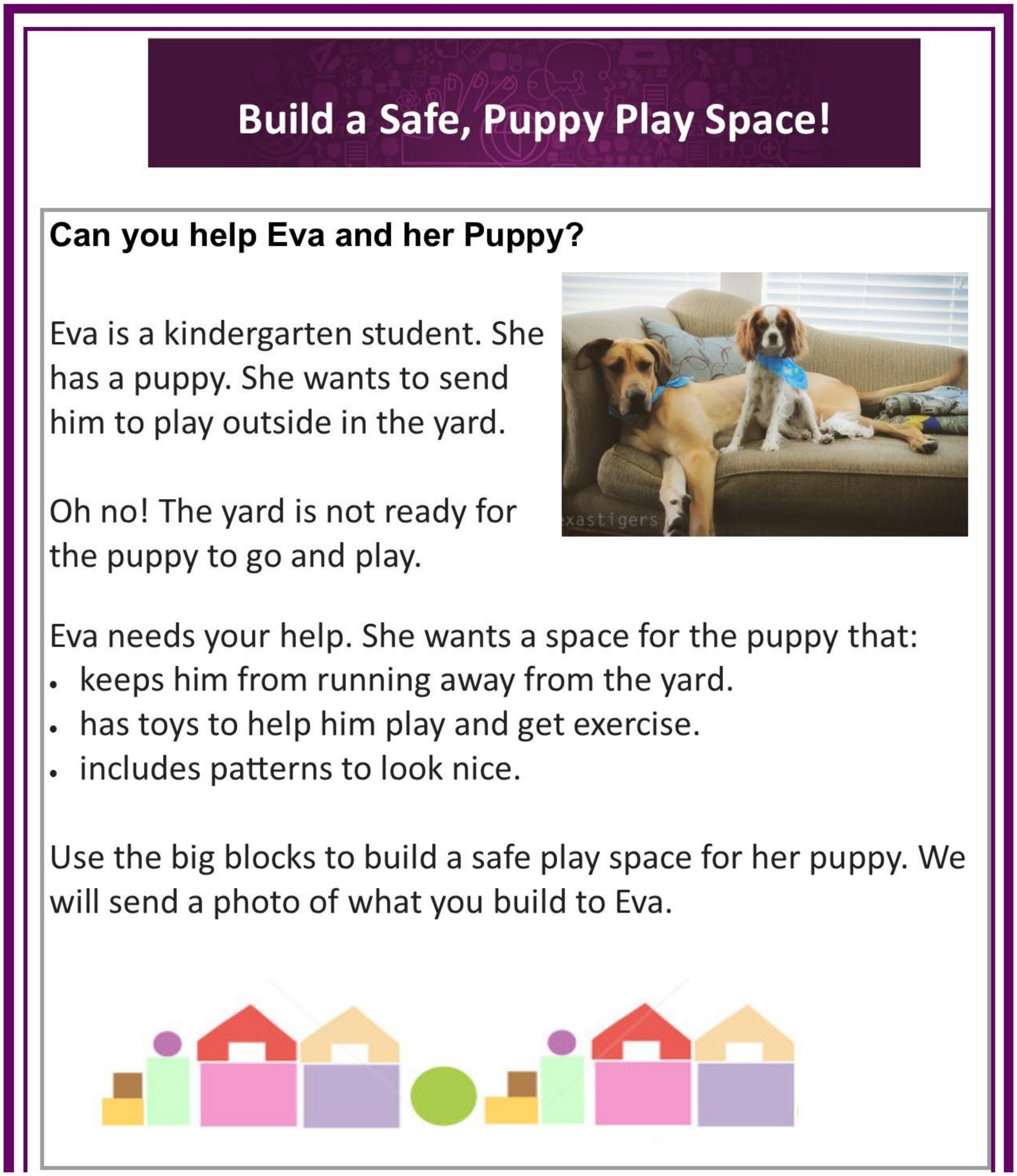
To date, five families have participated in this study. In this paper, we focus on three cases to begin to map out the space of skills and competencies that children and families can engage in, without making claims about how common it is for children or families to engage in these competencies. For this study, data sources include video recordings and field notes collected while the families engaged in the activity, audio-recordings of the interviews with the adults and children and transcripts of the video and audio recordings. .

Data Analysis

To analyze the data, we created a codebook. The codebook was organized based on the three phases of *Problem Scoping*, *Development*, and *Implementing and Improving*. The codebook included the definitions of CT competencies as well as the abbreviation for each competency as a code. In order to analyze the video data, we followed the analytical model suggested by Powell, Francisco, and Maher (2003). The model consists of seven non-linear phases:

1. Viewing attentively the video data
2. Describing the video data
3. Identifying critical events
4. Transcribing
5. Coding
6. Constructing a storyline
7. Composing a narrative.

Figure 1: **The text of the design challenge that was presented to the families.**



Build a Safe, Puppy Play Space!

Can you help Eva and her Puppy?

Eva is a kindergarten student. She has a puppy. She wants to send him to play outside in the yard.

Oh no! The yard is not ready for the puppy to go and play.

Eva needs your help. She wants a space for the puppy that:

- keeps him from running away from the yard.
- has toys to help him play and get exercise.
- includes patterns to look nice.

Use the big blocks to build a safe play space for her puppy. We will send a photo of what you build to Eva.

Findings

Analyzing our video data provides us with insights of computational thinking competencies that may be observed when families of 5-8 year-old children engage in an engineering design activity. In this section, we first provide general examples of what we have seen occurring among all three families. Then a narrative of one family is included to provide a more complete description of what computational thinking can look like in a series of family interactions.

General Examples

The table below is organized as three phases of a computational thinking approach to problem solving mentioned above. The CT competencies associated with each phase, their definitions and examples are presented in Table 2. Table 2 consists of pictures of the structures that the families created. From the table, we see that the three families engaged in activities that mapped to all seven computational thinking competencies.

Table 1. CT Problem solving phases

Problem Scoping		
CT Competency	Definition	General Example(s)
Problem Decomposition	Breaking down data, processes or problems into smaller and more manageable components to solve a problem.	Identifying the sub-components of the task by asking questions like: Where to build? How? What to do? Who should do what? What we need?
Pattern Recognition	Observing patterns, trends and regularities in data.	Asking questions about/talking about what the playground might include based on real life examples of playgrounds.
Abstraction	Identify and utilize the structure of concepts/main ideas.	Talking about the main parts of the playgrounds by considering similarities across real-life examples (e.g. all playgrounds have fences and something to play with).
Development		
CT Competency	Definition	General Example(s)
Algorithms and procedures	Following, identifying, using, and creating an ordered set of instructions. (ie, through selection, iteration and recursion)	Selecting appropriate blocks in order to build parts of the playground. Building the structure block by block using adult's directions.
Pattern Recognition	Observing patterns, trends and regularities in data.	Selecting blocks and putting them together based on their experiences of what has and has not worked well.
Abstraction	Identify and utilize the	Identifying the main characteristics

	structure of concepts/main ideas.	of parts of the playground and selecting blocks to build something similar (e.g. using a circular block as a wheel toy).
Parallelization	Simultaneously processing smaller tasks to more efficiently reach a goal.	Dividing the work of building two parts of the playground simultaneously (e.g. one builds the window on one wall, the other builds the next wall).
Implementing and Improving		
CT Competency	Definition	General Example(s)
Debugging	Identifying and addressing problems that inhibit progress toward task completion	Noticing a problem in the structure and trying to solve it (e.g. re-structuring the wall so that the dog may jump over it)
Pattern Recognition	Observing patterns, trends and regularities in data.	Finding the cause of the problem and solving it using what they have experienced before while playing.
Problem Decomposition	Breaking down data, processes or problems into smaller and more manageable components to solve a problem.	Breaking down the problem and its causes in order to solve the problem.
Simulation	Developing a model or a representation to imitate natural and artificial processes.	Playing with/trying out the playground they made to imitate how a pet might use the playground.

Computational Thinking Enacted by One Family

In Table 3, we present a narrative of the experience of one family as they worked on the activity. The narrative allows us to see when the different computational thinking competencies were enacted, the sequence of the family's process, and a concretized version of computational thinking during a family design activity. The family consisted of a mother and her children John (7-years old) and Daniel (4-years old).

Table 2. Examples of Families Engaging in CT.

Narrative	Computational Thinking
The mother begins by reading the task statement aloud to her children and describing the criteria to her children. She then starts the conversation below: Mother: so, what is your plan John? John: I'm gonna build a fence and then toys. Mother: what the fence is gonna look like?	The family begin to break down the problem in a way that helps them define the scope of the problem better. In particular, John identifies the fence and the toys as the two major components of the task (**Problem Scoping-Problem Decomposition)**. In addition, we see that John is able to imagine

John ; A large rectangle or may be a circle. Maybe, I can use this [pointing to a circular block on the top). Mother: Okay where is gonna be? John : Over there. Mother:It is gonna be like this? Well, where we are going to start? John: Right here.	the playground in a yard and focus on features that are common for playgrounds (**Problem Scoping-Pattern Recognition & Abstraction)**. He then focuses on important details of the fence and then based on the fences he might have seen in the real life, he realizes that the fence should have a rectangular or circular shape (**Development-Pattern Recognition & Abstraction).**
Then, the children look for some blocks that they might need by pointing out to and/or bringing the blocks. When John confirms the selection of the blocks, the mother asks the children where they want to place the base of the fence. Then, John tells the mother and Daniel where to place the rest of the blocks.	Children build the fence by the ordered set of instruction that John provides. The ordered set of instruction applies to selecting the blocks and then building the fence. This is an example of **Development- Algorithm**.
John builds the fence, using the same rectangular blocks on top of each other. He uses a cylindrical block as the connector, and connect the blocks. Then, he uses the connectors for other parts as well.	John identifies a pattern of the connector holding the blocks together, so he continues using them for building the fence. This is an example of **Development-Pattern Recognition**.
During building the structure, the mother suggest to build a door or a gate in a corner of the fence, but John responses that they can also build a wall.	Both John and his mother makes connection to the yards in real life by mentioning its possible components like gate, fence or a wall. This is an example of **Problem Scoping- Abstraction.**
In the process of making the fence, Daniel puts several blocks on the top of each which make that fence taller than others. John disagrees and tells his sibling that "it should be the same size. Look at this. You should take one, and then another one."	The child knows that based on criteria, they should create a pattern, and building one of the fence taller than others does not make a pattern (**Implementing and Improving-Pattern Recognition)**. Then, he intends to solve this problem (**debugging**) by taking out two blocks.
John builds stairs using few small rectangular blocks. Mother: what is this for? John: umm, I don't know, but it is stairs. Mother: is it a toy or something that the puppy can play with? John: I don't know. It is just stairs. The mother tries to convince him that the stars would not work, because the puppy can jump over it and get out of the fence. Daniel acts out like a puppy and jump on the stairs and then over the fence.	The child builds a structure and calls it stairs because he know stairs look like that. This is an example of **Development-Abstraction**, and **Implementing and Improving-simulation**. The mother tries to encourage them to do **Implementing and Improving - Debugging** by showing the problem, and the second child **Implementing and Improving-Simulate** how the problem would look like.

Towards the end, the mother suggests to build a gate or a door. She describes the door as “it opens and we can let him [the puppy] in.” Daniel nods and gets two blocks which look like hinges and a connector, and builds the gate.	He was able to identify the concept of a door being open and closed, focus on the important information of the concept, and utilize that information in his structure by creating something a hinge. This is an example of **Development-Abstraction**.

Discussion

The examples we have seen in our data suggest that children are capable of engaging in computational thinking when they working on engineering tasks with their families. The process of computational thinking they demonstrated is iterative not linear. We have seen examples of all seven of the competencies we focused on in the data we collected. Below, we provide further discussion of pattern recognition, abstraction, and debugging. We now focus on these three competencies because pattern recognition, abstraction, and debugging were repeated in more than one phases. Problem decomposition occurred quite the same in both phases of problem scoping and improving and implementing. Both abstraction and pattern recognition were seen slightly differently in different phases.

In our study, **Pattern recognition** happened in two different ways. In the problem scoping phase, it was embedded into abstraction. This will be discussed in the next paragraph. Pattern recognition in both development and implementing and improving phases occurred when children used their experiences of failure and success in working with blocks to develop or improve the structure. They recognized which blocks went on top of each other better and were stable and look the way they want to.

In our dataset, **Abstraction** happened in both problem scoping and development phases. Abstraction in both phases can be observed through pattern recognition, but in different levels. This is consistence with Bennett and Müllar’s interpretations of abstraction (Bennett & Müller, 2010). They argue that through abstraction, children are first able to identify features based on the overall appearance of similar items like the patterns they see in similar objects. An example for the task of this study would be that based on what children saw in most of the real-life playgrounds (patterns), they realized they should include a fence/wall and gates around the playground and toys or play equipment in the middle. Planning to build a similar playground was abstraction. Then, they recognized independent features of an object and later they considered more features of the objects. This is what happened in the development phase. In the development phase, children focused on the main details of the part they decide to build (e.g. a gate). They recognized the main details from the similarities they see of that part in the real-life examples (patterns). Then based on those details, they selected and used the blocks to build a structure very similar to the real-life one.

We also noticed that in all the examples seen in the implementing and improving phase, pattern recognition, problem decomposition and simulation were all a sub-set of **debugging**. Through debugging, children noticed the cause of the problem in their structure, and then they enacted the three other computational thinking competencies to solve the problem.

Conclusion

Our findings provide evidence that 5-8 year-old children are capable of enacting computational thinking competencies when interacting with an adult in solving engineering tasks. All seven of the CT competencies of abstraction, algorithms and procedures, debugging, problem decomposition, parallelization, pattern recognition and simulation were observed happening in this study. We expect that the findings are not limited to this task and study, and can be seen occurring in other engineering design tasks. Because we were able to capture examples of computational thinking in our study, we believe that further studies of computational thinking amongst 5-8 year-old children is a productive direction for future research as well as a productive direction for interventions aimed at promoting CT competencies in children. In addition, further research on how parents' and other adults' interactions with children promotes computational thinking in children should be conducted.

Acknowledgements

This material presented in this symposium is based upon work supported by the National Science Foundation Division of Research on Leaning under Grant No. DRL 1543175. 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. We would also like to acknowledge the contribution of the STEM+C research team at INSPIRE, Purdue University, Imagination Station as well as all the families who participated in this study.

References

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.

Bell, P., Lewenstein, B., Shouse, A. W., & Feder, M. A. (2009). *Learning science in informal environments: People, places, and pursuits*: National Academies Press.

Bennett, J., & Müller, U. (2010). The development of flexibility and abstraction in preschool children. *Merrill-Palmer Quarterly, 56*(4), 455-473.

Computer Science Teacher Association (CSTA), & International Society for Technology in Education (ISTE). (2011). Computational Thinking Teacher Resources (Second ed.).

Dorie, B. L., Cardella, M.E., Svarovsky, G. (2014, June). Capturing the Design Thinking of Young Children Interacting with a Parent. In *Proceedings of the American Society for Engineering Education Annual Conference & Exposition*, Indianapolis, IN.

Dasgupta, A., Rynearson, A., Purzer, S., Ehsan, H., & Cardella, M. (2017, June). Computational thinking in Kindergarten: Evidence from student artifacts. In *Proceedings of the American Society for Engineering Education Annual Conference & Exposition*, Columbus, OH.

Hynes, M. M.,Moore, T., Cardella, M., Purzer, Tank, K., Meneske, M., & Brophy, S. (2016, June). Inspiring Computational Thinking in Young Children's Engineering Design Activities (Fundamental). In *proceeding of the 2016 American Society for Engineering Education Annual Conference & Exposition*, New Orleans, LA.

Kotys-Schwartz, D., Besterfield-Sacre, M., & Shuman, L. (2011, October). Informal learning in engineering education: Where we are—Where we need to go. *In Frontiers in Education Conference (FIE)*

National Research Council. (2011). *Committee for the Workshops on Computational Thinking: Report of a workshop of pedagogical aspects of computational thinking*. Washington, D.C.

National Academy of Engineering and National Research Council, 2009, *Engineering in K- 12 Education: Understanding the Status and Improving the Prospects*.Washington, D.C.: The National Academies Press.

Paulsen, Christine A., Monica E Cardella, Tamecia R Jones and Marisa Wolsky "Informal Pathways to Engineering: Interim Findings from a Longitudinal Study," Proceedings of the American Society for Engineering Education Annual Conference & Exposition, Seattle, WA, June 2015.

Powell, A. B., Francisco, J. M., & Maher, C. A. (2003). An analytical model for studying the development of learners' mathematical ideas and reasoning using videotape data. The journal of mathematical behavior, 22(4), 405-435.

Purdue University's INSPIRE Research Institute for Pre-College Engineering (2017). *Computational Thinking Competencies: INSPIRE Definitions*. Unpublished resource.

Stevens, R., & Bransford, J. (2007). The LIFE Center's lifelong and lifewide diagram. Learning in and out of school in diverse environments: Life-long, life-wide, life-deep. Seattle, WA: University of Washington Center for Multicultural Education.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.