Interdisciplinary TPACK: A Case Study Using Variables, I/O, and Loops with Graduate Teacher Candidates in a Methods Course

Paris Kalathas*
kalathap@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

Jennifer Parham-Mocello
parhammj@engr.orst.edu
Oregon State University
Corvallis, Oregon, USA

ABSTRACT

This research paper describes the mathematics teacher knowledge transformation when computer science concepts are integrated with mathematics. During the last decade, schools and educators made a significant effort to introduce Computer Science (CS) and Computational Thinking (CT) in K-12. While this effort produced some positive results in schools where resources and specialized educators are available, there are school districts where creating new classes and/or hiring specialized teachers is not a sustainable option. This creates a need for teachers that are specialized in one subject area to expand their knowledge into CS and CT, and teach these concepts in their classes. This knowledge expansion and integration should result in creating integrated curricula using their subject matter knowledge and the newly acquired CS knowledge. This implies that the teacher should be able to identify the intersection of concepts between the two fields, and leverage their difference and similarities to promote deep learning for their students. The current study explores how the teacher candidates' technological, pedagogical content knowledge (TPACK) in mathematics is affected when mathematical concepts are integrated with CS and CT. This study is a three-day intervention in a graduate level secondary mathematics education methods course, where utilizing Math + CS integrated modules we infuse CS concepts into mathematics. The already known mathematics context reduced the cognitive load for familiarizing the teacher candidates with CS, and the integration of the CS concepts provided a different perspective for thinking about mathematics and teaching. After having the teacher candidates reflecting on the concepts of *Variable* and *Equal Sign (=)*, our results showed that they started noticing the differences and nuances between the two fields, developing this way their interdisciplinary Math + CS TPACK (I-TPACK) knowledge. Furthermore, the study showed that representing multiplications using repetition structures illuminated the internal mechanic of multiplication and its properties. Finally, the experience with the strict grammar and syntax rules of programming illuminated the areas in the teacher candidates mathematics instruction that were ambiguous in the way they teach variables, equations, percentages, and multiplication.

TBA, TBA, TBA, TBA, TBA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Social and professional topics → K-12 education**; **Computational thinking**; **Computer science education**; • **Applied computing → Computer-assisted instruction**.

ACM Reference Format:
Paris Kalathas and Jennifer Parham-Mocello. 2024. Interdisciplinary TPACK: A Case Study Using Variables, I/O, and Loops with Graduate Teacher Candidates in a Methods Course. In *Proceedings of ACM Conference (TBA).* ACM, TBA, TBA, TBA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Digital technology provides to our society the means to create better products, communicate faster and more efficient, access more information, and have better control of our lives [38]. However, the prevalence of digital technologies in our everyday lives has resulted in a high demand of citizens who are fluent in Computational Thinking (CT) and understand concepts related to Computer Science (CS) [40–42]. To meet this demand, countries around the world have tried to find ways to introduce CT and CS in K-12 and implemented policies as a commitment to this goal [12, 36]. One challenge has been to expand the knowledge of teachers specialized in one subject area to CS and CT, so they can integrate CS and CT into their teaching. In this study, we use Math + CS modules to introduce the CS concepts of variable, input/output, and repetition embedded into the mathematical concepts of principle investment and geometrical shapes. To provide a deeper understanding of mathematical and CS concepts we integrated Math + CS, and we use programming as a tool for expressing and visualizing mathematical concepts in the context of CT and CS. We explore how secondary mathematics graduate teacher candidates' technological, pedagogical content knowledge (TPACK/TPCK) is shaped through their experiences with the Math + CS modules in a graduate-level teaching methods class focused on mathematics pedagogy and technology. Specifically, we aimed to answer the following question.

How does graduate teacher candidates' mathematics TPACK knowledge transforms when they interact with Math+CS integrated modules in a secondary mathematics teaching methods course?

2 TEACHER KNOWLEDGE

2.1 Knowledge Domains for Teaching

In 1986 and 1987, Lee S. Shulman defines the Pedagogical Content Knowledge (PCK) for teaching as:

"the blending of content and pedagogy into an understanding of how particular topics, problems, or issues are organized, represented, and

adapted to the diverse interests and abilities of learners, and presented for instruction. Pedagogical content knowledge is the category most likely to distinguish the understanding of the content specialist from that of the pedagogue" [34, 35].

In 2006 after witnessing the integration of technology into instruction, Punya Mishra and Matthew J. Koehler introduce the technological pedagogical content knowledge (TPCK or TPACK) [20, 21, 25, 26]. TPACK complemented PCK by adding technological knowledge as a new knowledge domain. Education researchers examined how the technological knowledge domain got infused and transformed the other knowledge domains. The studies resulted to professional development programs, development of curricula and assessments to support teachers [5, 19, 24, 27, 28]. In Europe, the European Digital Competence Framework (DigCompEdu) is used to pinpoint the digital competences educators need to master to incorporate digital technologies into their subject matter, as well as teach their students how to use those digital technologies [30]. The framework consists of 22 basic competences organized into six areas. Area 1 addresses educators' professional environment and the way educators use digital technology to engage with people in that environment other than teaching. These people may be students, colleagues, parents, and administrators. Area 1 also addresses the engagement in self-reflection and the digital continuous professional development (CPD) of the educators. Areas 2–5 focus on the competences educators need for pedagogical purposes. Areas 6 addresses the transfer of those digital competences to the students through specific pedagogical guidance. The TPACK and DigCompEdu frameworks are useful for developing teachers' knowledge within a content area; however, they are not suitable for X + CS integration. A recent study introduces the CTIntegration framework for integrating computing and CT within a host discipline [17]. The framework is based on the five integration elements: 1. Understanding complex systems, 2. Innovating with computational representations, 3. Designing solutions that leverage computational power and resources, 4. Engaging in collective sensemaking around data, 5. Understanding potential consequences of actions. However, the CTIntegration framework only considers the content and pedagogy knowledge domains, without taking into account technology knowledge domain. While research exists on frameworks for integrating CS and CT into K-12, these studies focus on 1. teaching CS and CT as a standalone course [2, 37], 2. isolate CT as a problem solving process in K-12 classrooms [10, 18, 22], 3. consider CS and CT as parts of the technological knowledge and used as tools for enhancing pedagogical purposes [4, 6, 7, 32], and focus only one some knowledge domains [17].

To address the need of teaching CS where CS as a standalone course cannot be introduced, and to broaden the participation in CS by showing its relevance and connections with other fields, there is a need for a framework that examines and provides guidance for the integration of CS as a subject matter into other disciplines differentiating it from technology [3]. The integration should not just focus on enhancing current pedagogical methods or superficially acknowledge the connections between the disciplines. To broaden teachers' knowledge the integration should be structured in a way that illuminates the affordances each discipline offers, the differences in their definitions, and what each discipline's limitations are.

2.2 Knowledge for Teaching X + CS - Theoretical Framework

Integrating CS into another discipline, which we refer to as X + CS, requires teachers to understand two different subjects and their relationships. X + CS integrated curricula is more than integrating technology into the instruction of one subject. An interdisciplinary X + CS integrated curricula must highlight features of CS and CT that are common to and offer a different perspective of the X subject matter to create new solutions and ideas (see Fig. 1 right). Whereas, a multidisciplinary approach uses features from CS and features from X to create new solutions and ideas, such as using CS and psychology to create artificial intelligence (see Fig. 1 left), and a cross-disciplinary approach applies features from CS to expand the boundaries of X, like using high-performance computing to better understand genomics (see Fig. 1 middle). An interdisciplinary approach goes beyond multidisciplinary and crossdisciplinary approaches by finding the intersection of features between the disciplines [31], such as the concept of a function existing in both mathematics and CS. In an interdisciplinary approach, curriculum developers and teachers need to explore and understand the CS content and how to integrate CS content knowledge with existing content knowledge to form X + CS interdisciplinary content knowledge [43]. To achieve this, Woods suggests a model of Interdisciplinary Communicative Competence for forming proper communication between disciplines. Woods' model consists of eight components.

(1) Conceptual competence: the ability to identify and gather the appropriate tools and concepts from different disciplines required to solve the problem at hand.
(2) Competence in negotiating meaning: the ability to conceptualize different meanings of similar terms between disciplines and articulate those meanings to their audience.
(3) Competence in interdisciplinary text production: the ability to recognize diversity in literature between disciplines and to make it accessible to their audience.
(4) Knowledge of how their discipline's products and practices shape and can be shaped from perspectives of other disciplines.
(5) Skills of interpreting and relating: the ability to understand and relate products from another discipline, and explain their intersection with their own discipline.
(6) Skills of discovery and interaction: the ability to acquire new knowledge of a different disciplinary culture and use that knowledge in their own interactions.
(7) Attitudes of curiosity and openness: Having the willingness for engaging in deep understanding of other disciplines that spans further of just learning about and using their concepts.
(8) Critical disciplinary awareness: the ability to think critically about practices and products in various disciplines based on explicit and implicit criteria and perspectives.

Interdisciplinary knowledge construction requires concepts and methods from different fields integrated and studied together. Linn et al. presents the knowledge integration (KI) framework with four

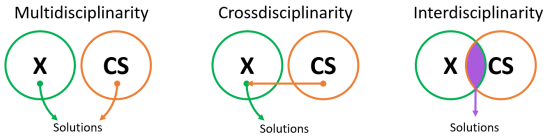

Figure 1: Multidisciplinary, Crossdisciplinary and Interdisciplinary Knowledge

processes for forming interdisciplinary knowledge [23], 1. Eliciting ideas: The use of students' prior knowledge to create relevant connections with the new knowledge in a learning environment, 2. Adding new ideas: Design activities to add ideas for students to explore them and form connections between the new and their existing ideas, 3. Distinguishing ideas: Design activities so students can reflect on the new ideas and distinguish between those that are useful for solving the problem and those which are not, 4. Sorting out ideas: Students have to sort out the connections between shared ideas to develop a coherent understanding of the subjects.

To teach X + CS integrated curricula, teachers have to integrate Technology, Pedagogy, Content, and Contexts knowledge from the subject matters of X and CS, developing their Interdisciplinary - TPACK (I-TPACK) for the X + CS instruction. Taking into account the work of Shulman, Mishra, and Koehler, we present a visualization of where I-TPACK fits within a teacher's knowledge domain (see Fig. 2). As the figure shows, I-TPACK teacher knowledge is a transformed knowledge which occurs due to integration of concepts from multiple knowledge domains. Each domain interacts with other domains in a unique way; however with I-TPACK as a product, we focus on those interactions that are particularly necessary for teaching. The placement of each knowledge domain in the figure is not coincidental. We argue that teachers need to integrate their subject matter X with the CS content knowledge to develop their X + CS content knowledge. Then, using their pedagogical knowledge, they need to find the most appropriate methods for teaching their newly developed integrated content knowledge in order to develop their interdisciplinary PCK. Finally, integrating technology into their teaching methods and taking into consideration the context for teaching X and CS, teachers develop their X + CS I-TPACK by engaging in the following steps.

- *Acknowledge* the relationship between the two disciplines. The teachers should witness and experience the interactions of their field and CS. They need to develop an open mindset and curiosity for exploration how the two fields interact before attempting any effort for content knowledge integration.
- *Clarify* terms, field jargon and develop interdisciplinary definitions. The teachers need to investigate the common terms between the fields and identify differences and similarities. Furthermore, disciplinary jargon terms need to be translated and explained in a way that people from other fields can understand them. Finally, they need to create new, interdisciplinary definitions of terms that capture the breadth on their application in the real world.

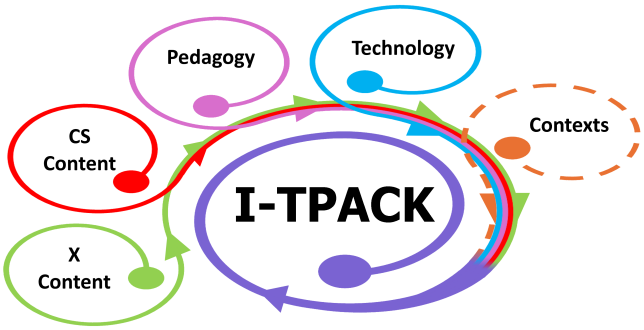

Figure 2: X + Computer Science integrated TPACK (I-TPACK).

- *Reflect* and develop knowledge. The teachers need to sort out the new concepts they just acquire and reflect on them making the necessary connections between them and their subject matter. They need to examine the similarities and differences, and find out the affordances each field offers to each other. They need to not only know what and how a concept is, but also why it is, and in which ways is relevant or not. This is the stage where the raw information becomes the X + CS integrated knowledge.
- *Develop* material for teaching. Using their pedagogical and technological knowledge, teachers now create curricula for teaching their newly developed X + CS integrated content knowledge. They need to identify the best pedagogical approaches and technological tools that will unpack and illuminate the X + CS concepts.
- *Evaluate* and fine-tune. At this stage the teachers teach their material to their students, and based on student assessments and student feedback make the necessary adjustments to fine-tune that material for their class.
- *Expand.* After acquiring their initial interdisciplinary knowledge and interdisciplinary thinking, teachers should expand their knowledge by finding more connections between their subject matter and CS expanding their X + CS curriculum. Furthermore, they could develop their knowledge into fields other than CS, creating this way a diverse teaching curriculum that could be utilized for teaching diverse student audiences.

3 METHODOLOGY

To help teachers form their I-TPACK for teaching X + CS, it is important to create modules that focus on developing all knowledge domains together, rather than just only one knowledge domain independent of the others. With this in mind, we present our method for creating Math + CS modules to develop teachers' I-TPACK.

3.1 Site and Participants

The study took place in a graduate-level secondary mathematics methods course in the college of education on a university campus. The course was an excellent fit for our modules because it integrated a focus on mathematical knowledge for teaching, equitable instructional skills for diverse learners, and the strategic use of

Table 1: Student groups and their CS experience.

Group	Students	CS Experience
GP1	Kim, Shawn	Intro to CS, Intro to CS
GP2	Ginny, Rhoda	Intro to CS, No exp.
GP3	Cassie, Soria	Matlab + Rstudio, No exp.
GP4	Levi, Samantha	No exp., No exp.
GP4/GP5	Chad, Taylor	No exp., No exp.

technology to support student learning with the Common Core State Standards for Mathematics. Methods courses are suggested by the literature as the ideal opportunity for integration of CS and CT [44]. The graduate teacher candidates were divided into three groups of two and one group of three, for the first and third day, and 5 groups of two during the second day. The selection of the groups was made based on their previously noted patterns of participation in the class they intervention took place. Working in groups draws from a situative perspective, suggesting that joint participation in an activity shapes one's understandings, intentions, and goals [14–16].

3.2 Coding Environment

After researching different programming environments for their strengths and drawbacks, we decided that a hybrid environment was best-suited for our modules. Hybrid environments offer both a textual and a block-based language interface; thus offering diverse interaction for teachers and students [1, 39]. In addition, BlockPy offers both block-based and text-based environments simultaneously with automated translation from one environment to the other, providing a representational coordination of the code at the same time [8, 9]. BlockPy uses Python as its programming language; therefore the teachers can use it in classes of different grades, making it a tool with a more broad use than a single modality environment [39]. To familiarize the graduate teacher candidates with the programming environment, we provided workspace tour document with explanations for all the buttons and different areas of the workspace and we gave them time to read it and explore the different areas of the environment Fig. 3. The graduate teacher candidates worked in pairs, shifting roles between coding and directing, at regular intervals across the modules.

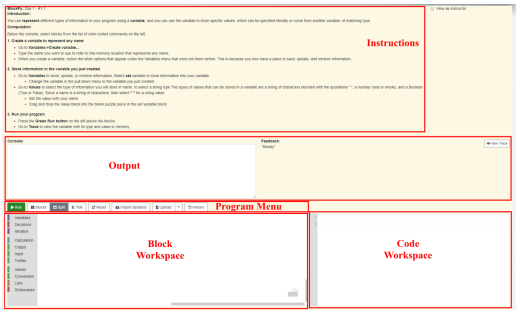

Figure 3: The BlockPy programming environment with instructions

3.3 Concept Selection Process

Since the teacher candidates were going to teach the material they learned from the Math + CS modules in the classroom where they were doing their student teaching, the first step was to identify concepts covered in 7th grade secondary mathematics education. We reviewed the Common Core State Standards for Mathematics [29] and the curricular material covered in the class where the teacher candidates were practicing their teaching. We identified the mathematical concepts of "Ratios and Proportional Relationships", "Expressions and Equations", and "Geometry" and the mathematical practices of "Reason abstractly and quantitatively", "Model with mathematics", "Attend to precision", and "Look for and make use of structure", and we applied the concepts and practices within the context of real-world problems, such as principle investment with simple interest and geometric modeling of polygons, which the standards suggested as age-appropriate for 7th grade [29]. For the CS content, we used the CSTA K-12 standards [11], and we selected the concepts of *Variable*, *Input/Output*, Order of Execution, *Repetition (Loops)*, and *Computer Graphics* from "Algorithms and Programming" and the practice of "Developing and Using Abstractions" to help develop a more nuanced understanding of the mathematical concepts and practices, while also learning about CS (see Table 2). The complete Math + CS modules are in Appendix 8.

3.4 Three Design Principles for Math+CS Teaching Modules

When creating our Math + CS mathematics teacher education modules, we followed three design principles (see Fig. 4). The first principle was the *Coordination of Complexity* among concepts. To promote understanding and conceptualization of the different concepts while reducing the cognitive load in each exercise, we coordinated the complexity between basic mathematics and CS concepts and practices. Each new computational exercise leveraged existing understanding of the basic mathematical concepts and introduced only one new CS concept at a time, and each new concept was connected to the previous concept(s) to highlight the learning trajectory and make the necessary connections between what was learned and the new concepts being learned. The second principle

Table 2: 7th-Grade Math and CS Concepts and Practices Covered

Real-World Context	
Principle Investment and Simple Interest Geometric Modeling of Polygons	
Mathematics	
Concepts	Practices
Ratios and Proportional Relationships Expressions and Equations Geometry	Reason abstractly and quantitatively Model with mathematics Attend to precision Look for and make use of structure
Computer Science	
Concepts	Practices
Algorithms and Programming Variables Input/Output Order of Execution Looping Graphics	Developing and Using Abstractions

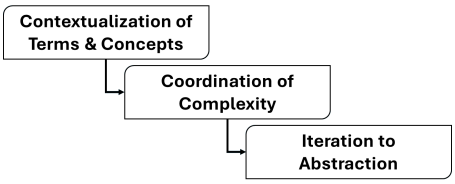

Figure 4: Design Principles for the Math+CS Integrated Modules

was the *Contextualization of Terms and Concepts*. With this principle, we utilized concepts common to Mathematics and CS within the context of simple real-world problems, and we showed how the concepts were similar and different in the two disciplines to better develop interdisciplinary Math + CS knowledge. The third design principle we employed for the module design was *Iteration to Abstraction*. Through this process, we provided opportunities for the learners to iteratively develop solutions that moved from concrete to abstract thinking. To iteratively develop a generalized solution for a specific problem requires a deeper understanding of what each element in the solution represents and forms patterns about how that element is connected with a larger group of elements across disciplines and problems.

3.5 Computational Exercises - Module 1

Module 1 began with an introduction to the problem of "Principal Investment and Simple Interest" and the computational exercises. Additionally, we provided the graduate teacher candidates a BlockPy workspace tour document with explanations for all the buttons and different areas of the workspace environment. Then, we gave the pre-service teachers and graduate teacher candidates computational exercises to explore the CS and Math concepts and practices. After the problem was introduced, the first exercise asked the graduate teacher candidates and undergraduate pre-service teachers to create one variable to store the name of the person investing the money and another variable to store the amount the person was investing. This exercise introduced the concept of variables, the different types of values variables can store, and the functionality of the equal sign as an assignment operator in programming, and in BlockPy, we used the "Trace" feature to highlight these concepts. We also wanted to generate possible errors that can occur when teaching students to program and address why the errors occurred to help build their pedagogical content knowledge for teaching CS. So, we asked graduate teacher candidates and undergraduate pre-service teachers to rearrange the operands provided to the assignment operator and describe what happened when they executed their program. This exercise reinforced the concept of the equal sign being an assignment operator, introduced the importance of precise syntax in programming, and enforced the practice of attending to precision in mathematics.

In the next exercise, we showed how to print variables to the output window, and we asked the graduate teacher candidates and undergraduate pre-service teachers to print their variables along with literal values for messages about what they were printing. Then, we asked the teacher candidates and pre-service teachers what happens when they place their instructions for printing before they set their variables with values using the assignment operator and run their code to illustrate the sequential order of execution and the types of errors students make when programming.

In the third computational exercise, we changed the problem to ask for the user's name and the amount for investment as input and to print the information to the screen. This exercise addressed abstraction and generalizing algorithms from the CS standards and reasoning abstractly and quantitatively from the mathematics standards. In the last exercise of Module 1, we introduced the concept of repetition/looping using the context of calculating simple interest for some number of years. We used a loop to repeatedly add the interest to the investment and discussed the structure of the loop. While introducing the CS concept of looping, this exercise also covered the mathematical practices of thinking abstractly and looking for and making use of structure.

3.6 Computational Exercises - Module 2

In the second module, we reinforced the concepts learned in the first module using a different context. Since the graduate teacher candidates were going to practice teaching the concepts from the modules to 7th grade students, we used the concept of graphics from CS to teach the concept of geometry and the practice of modeling mathematics found in the 7th grade common core mathematics standards. We used the turtle library in the Python programming language to draw polygons. We started by showing the teacher candidates how to draw lines and make the turtle turn. Then, we had the teacher candidates draw a square on the screen. Next, we asked them to generalize their program to draw squares of any size based on input from the user to continue practicing abstraction. After drawing squares, the teacher candidates drew an equilateral triangle to uncover the patterns within the algorithms used for drawing different shapes, such as the relationship between the angle degree and the number of angles in a shape. Using their understanding of the patterns between the algorithms for drawing a square and triangle, the teacher candidates modified their algorithm to draw any regular polygon based on the number of angles and length of the sides entered by the user. This exercise allowed the graduate teacher candidates to continue practicing all the CS concepts learned in first module, while also exploring the concepts of geometry and graphics using the mathematical practice of modeling in mathematics.

3.7 Data Collection & Analysis

To ensure our understanding and the validity of our interpretations, we implemented data triangulation by collecting the groups' written responses to the questions on CANVAS, their responses to the questions through the discussions in their groups using audio and screen recordings, and whole class discussions for every day of the intervention using a whole class audio/video recording. After transferring and converting all the data to file formats accessible from all the researchers, we created transcripts using the audio recordings. One researcher prepared a codebook using deductive coding. The data segmentation was based on the change of the subjects, which most of the times spanned many lines and different speakers. Thus, in addition to the codebook, we included additional instructions

for segmentation. Then together with a second researcher they performed coding on 20% of the data resulting to an 86.6% inter-rater reliability, which we considered high enough to continue with the coding of the rest of the data. All the material related with the study can be found in the Appendix.

4 RESULTS

4.1 Variables

Working with variables, the teacher candidates notice and reflect on the differences on variable's definition and how they are used in mathematics and CS. They notice that the variables in CS have different types, such as numbers, strings and Boolean values, where in mathematics are mostly numbers. They understand that in CS variables are storage spaces for values (data), thus it makes sense to them that could be of different types of information that are stored. Reflecting on applications of those different types in mathematics, the teacher candidates said that strings can be used for representing units in math equations, and Boolean values could be used in math proofs and inequalities. The teacher candidates state that in the mathematics instruction, precision in algorithms and the meaning of variables are not discussed enough, resulting in students not develop precise understanding about the variables. They state that their students sometimes think of them as fixed things (values) rather than something that changes. In contrast, they say that in CS thinking about variables cannot be avoided because students have to give them a name, a type and a value, enhancing this way deeper understanding and awareness of what they represent in their equations.

4.2 Equal Sign

When they are asked about the equal sign, the teacher candidates state that in mathematics the equal sign represents relationships between mathematical expressions. In contrast, in CS they use "=" to define values for variables. They mention that the block-based environment demonstrates the assignment property of the equal sign very clear, where in mathematics writing "x=4" on paper they would not think of it as 4 been assigned as a value of x. Furthermore, the teacher candidates notice that the equal sign in CS has more structure than in mathematics, meaning that due to syntax constrains in CS the equal sign is asymmetric. In mathematics, the equal sign is symmetric, because of its definition as representing a relationship, thus values and symbols can go on either side of the operator. The teacher candidates engage their pedagogical knowledge discussing how the differences between mathematics and CS on the equal sign definition and use could help eliminate bad student habits which make them assume that whenever an equation includes an equal sign it means "solve".

4.3 Simple Interest, Multiplication, Repetition

The teacher candidates are instructed to construct repetition structures where the products of the multiplications are calculated as repeated additions. Seeing multiplications as repeated additions exposes the underlying mechanics of multiplication, and the role of each number involved in the operation, Fig. 5. The teacher candidates witness how the multiplication process changes based on which number get multiplied as well as how the numbers change

Figure 5: Multiplication exercises in BlockPy.

because of that. They also because familiar with what the required pre-conditions and values are related with variables in CS before a repetition begins. They discus about the names of each parameter in mathematical operations, such as "multiplicand", "multiplier" and "product" stating that they rarely use those names in class with their students. Also, the exercises teach the graduate teacher candidates how to be precise with their equation representations. When it comes to the CS concept of repetition, the exercises illuminate its limitations related to the types of the parameters used as indices being integers, forcing the teacher candidates to use the repetition structures in only one way, which is something different of what they experienced in mathematics.

4.4 Algorithm Development

The teacher candidates reflect on the precision in algorithm development in CS, and that is not discussed enough in their mathematics classes. Figure 6 shows an error debugging exercise which helped the teacher candidates understand that even though all the information are in their code, the code is not executing because it is not in the right order. They identify that this may result in issues with the introduction of CS. They understood that in CS the order of operations is vital for the code execution, so the students will need to change their habits when it comes to thinking about and describing their algorithms and be more precise about which action happens at each step.

4.5 Programming Environment

During the intervention, we pay attention on how the graduate teacher candidates interact with BlockPy. Going through the workshop tour, the majority of the graduate teacher candidates became comfortable using the tool, and they did not hesitate to explore and tinker with the various blocks that were available in the block instruction menu. They state that the puzzle pieces and the colors made everything look more playful and nicer than the confusing lines of code which is what they associate programming with. They say that the split screen is very useful, because they could use the environment they feel comfortable with, thus making the tool suitable for diverse audiences. They state that the language BlockPy uses on the "Set" block to create variables reminds them the "Let"

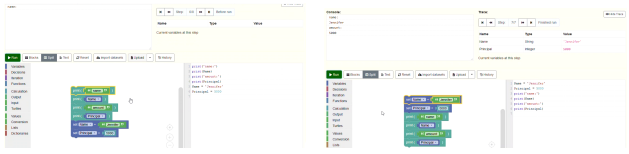

Figure 6: Create and print variables - Error (Left), Create and print variables - Correct (Right)

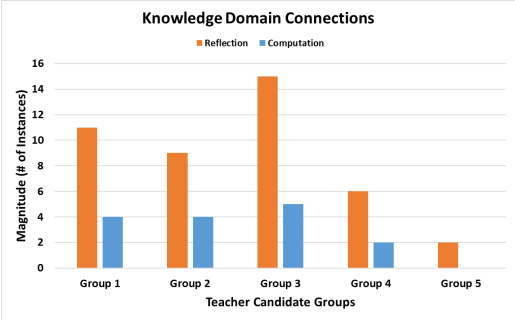

Figure 7: Knowledge domain connections during computation and reflection exercises

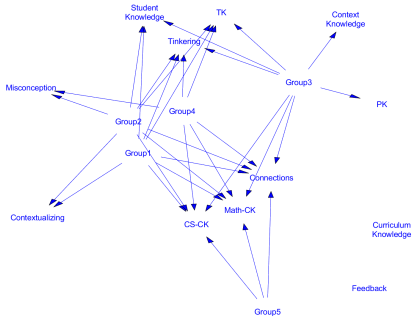

Figure 8: Knowledge domain connections made from each group including connections

statement in mathematics, which is something that they can leverage in their classes. Finally, we observe that the block instruction menu helps the programming novices to find the instructions as well as to the programming experienced teacher candidates since most of their experience is with a different language.

4.6 Knowledge Integration Through Reflection

After analyzing the frequency and the timing our codes appear in the transcripts, it shows that having a way to promote reflection greatly enhances knowledge integration and the I-TPACK development. While going through exercises and developing algorithms may teach and provide programming practice to the teachers, reflection questions make them understand why things are the way they are, in which ways they are related with other concepts in the same discipline or others, and how they can be used to solve problems or create new artifacts. The process of reflection is crucial if we want the new knowledge to stay imprinted on teachers brains so later they can develop their own material and successfully teach CS. Figure 7 shows the number of connection each group makes when they go through a computational exercise and going through the reflection questions. The figure shows that the teacher candidates make more connections during the reflection questions than during the exercises, revealing that to develop Math + CS I-TPACK, is not enough to go through CS material or develop digital artifacts. Teachers need to experience Math + CS curricula and engage in meaningful discussions to examine how the CS concepts are related and used in mathematics. This trend is common for all the groups regardless their previous experience with CS; however, the chart shows that groups 1, 2, and 3, which are the groups with previous CS experience make more connections than groups 4 and 5 which are the groups with no CS experience.

4.7 Knowledge Domain Engagement Frequency

To examine the frequency each knowledge domain gets engaged we visualize our codes using a pathfinder networks [33] which provide spatial isolation of the less used codes, and bar charts for visualizing the magnitude of each code based on how many times it is used. We construct the pathfinder network and bar chart of all codes for each group as shown in Figures 8 & 9. The Connections code is broad thus we extract the codes from the Connections to examine which knowledge domains are used as stand-alone domains, connected

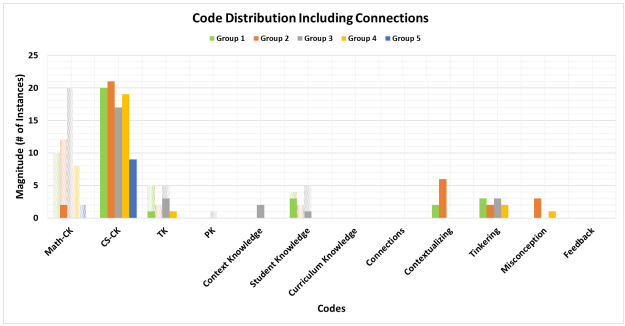

Figure 9: Data analysis code distribution chart including connections

with CS, or not used at all. The added codes are shown with lighter colors on each bar. The graph shows that the Curriculum knowledge domain is not used from the teacher candidates. In addition, the graph shows that the PK domain is used only by group 3, and that group 5 is the only group with limited number of codes. We believe this is because group 5 was only formed for the second day as Table 1 shows, thus some of the code for group 4 are shared between the four teacher candidates in groups 4 and 5. The bar chart shows that the knowledge domains with the highest magnitude are the mathematics content knowledge and the CS content knowledge domains, which is expected since these are the most used knowledge domains in the modules. Furthermore, the figure shows references to the technological, pedagogical, and student knowledge domains as well, which indicate the I-TPACK development.

5 DISCUSSION

In this chapter we examine how the teacher candidates' mathematics TPACK transforms when mathematical concepts are integrated with CS and CT. In the paragraphs below we discuss how the CS content knowledge complements the teacher candidates' existing TPACK knowledge domains. Table 3 demonstrates the ways the Technology, Pedagogy, Mathematical Content knowledge interact with the CS Content knowledge developing the teacher candidates' I-TPACK.

Reflecting on the concepts of *Variable* and *Equal Sign*, the teacher candidates form connections between their mathematics and CS

Table 3: Knowledge Domain Engagement and Integration

Technology	Pedagogy	Mathematics	Computer Science
Variables			
Placeholders			Memory space for different types of data
Values are numbers			Can be either numbers, strings, or Boolean
Students overlook their meaning			Meaning is emphasized due to their names
Students think of them having fixed values			Working on variables in BlockPy makes them more "changeable"
Equal Sign			
Defines relationship			Defines/Assigns a value
Symmetric			Asymmetric
Students' misconception "=" means solve			
Simple Interest, Multiplication, Repetition			
Underlying operations are hidden			"Trace" illuminates the underlying operations
Parameter meaning is hidden			Parameter meaning in the operation is enhanced
Operand names are not used			Enforces clarity on operation definition
Algorithm Development			
Variables are not discussed or defined			Execution order matters
Order of operation can be ambiguous			Defining variables and their values is crucial
Programming Environment			
Coding in text helps experienced teachers			Dual modality promotes diversity in coding styles
Blocks are less confusing than text			Block menu enables exploration
			Working with blocks is easy
			Working with the puzzle pieces is enjoyable
			BlockPy has similar language with Math

knowledge by identifying the differences and the similarities between the two fields. Working with the concepts from two different perspectives causes their definitions and properties to expand. Creating connections between fields is important for improving the teachers' lateral curricular knowledge as it is defined by Shulman in 1986 [34]. Furthermore, connecting knowledge between disciplines develops deeper understanding of the concepts which enriches teaching [13]. The graduate teacher candidates argue that they could adopt ideas from the use of variables and the equal sign in CS to emphasise mathematical concepts that sometimes get overlooked from their students. Through the error handling exercises, the graduate teacher candidates built a strong understanding about the strict nature of programming, and they identify which mathematical properties cannot be transferred in CS. Emphasizing on these differences and reasoning about them helps them develop a deep understanding of those concepts and helps them build their confidence working with the CS concepts.

The graduate teacher candidates engage their pedagogical knowledge identifying its intersection with the CS content knowledge and using that to find ways to help their students develop a broader understanding about mathematical concepts in their classes. They state that utilizing the strict syntax of programming they can eliminate students' misconceptions and enforce formal procedures when going through exercises. Furthermore, the CS concept of repetition can be used when teaching the concepts of *Simple Interest* and *Multiplication* as a pedagogical tool for precise representation of the operations and to examine the underlying mechanism of those operations. Furthermore, they engage in discussions about the way the parameters change during the operation execution and what effect do the properties of multiplication have on that change.

We argue that the selection of the tool based on the programming language and the programming modality plays an important role on how teachers and students communicate and interact with CS concepts developing their I-TPACK knowledge. Even though BlockPy is not as developed as other tools for teaching CS, its unique characteristics, such as the split screen and Python as the underling language are appreciated by the teacher candidates because they promoted diversity on the way they interact with the exercises. In addition, using a programming language that students of higher classes use ensures the relevant of the tool through out the teachers' careers. The trace feature was extensively used by the teacher candidates when working with the concepts of *Simple Interest* and *Multiplication* to understand the underlying operations of the multiplication. The teacher candidates argue that this was very helpful for them to learn because stepping through the algorithm brakes complex operations down to simple ones, easy to understand steps observable by the students.

The code analysis reveals that knowledge integration does not happen automatically or on every exercise. In contrast, there are only few instances when it happened during the exercises, showing that reflection questions and discussions are important for engaging domains that may not be engaged during the computation exercises. Knowledge integration requires explicit prompts and questions that promote thinking about the concepts from multiple perspectives, resulting in developing a new, broader perspective.

6 LIMITATIONS

This study explores graduate teacher candidates' interdisciplinary Math + CS TPACK development in the context of a 3-day intervention. The study is limited in sample size and diversity on both the participants and the disciplines selected for combining with CS. In addition, the group formation was made based on previous knowledge of the graduate teacher candidates' behavior. These limitations prevent us from making any generalizable claims; however, we believe the study provides a good starting point for future research in expanding teacher candidates' existing TPACK knowledge in CS within other disciplines.

7 CONCLUSION

In this study we explored how graduate teacher candidates in a teaching methods course engage with Math + CS modules designed to develop their interdisciplinary Math + CS TPACK. Using our three design principles, coordination of complexity, contextualization of terms and concepts, and iteration to abstraction we created modules with computational exercises that integrate CS concepts into mathematics. Our results show that the graduate teacher candidates engaged in meaningful discussions about how the CS concepts and tools can fit in their instruction and what affordances they bring. They created connections between their existing mathematics TPACK knowledge and the newly acquired CS content knowledge forming an interdisciplinary Math + CS TPACK. Although this was a short intervention with a small sample of graduate teacher candidates it revealed that the CS integration in other fields can be achieved, allowing teachers specialised in other fields to teach CS where teachers specialized in CS are not available. A critical role in the X + CS curricula development is the partnerships of CS and education researchers with K-12 educators to examine and

provide deep understanding of the CS concepts, how these concepts intersect their field and develop their X + CS I-TPACK.

8 ACKNOWLEDGMENTS

We will add acknowledgements upon acceptance.

A ADDITIONAL DOCUMENTS

All the documents related to this study can be accessed through the link below.

https://tinyurl.com/mr4h9wa5

REFERENCES

[1] Hussein Alrubaye, Stephanie Ludi, and Mohamed Wiem Mkaouer. 2019. Comparison of Block-Based and Hybrid-Based Environments in Transferring Programming Skills to Text-based Environments. *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering* (2019).

[2] American Institute for Research. 2019. Three Strategies for Preparing Teachers of Computer Science. (2019). Retrieved August 28, 2020 from https://www.air.org/resource/three-strategies-preparing-teachers-computer-science

[3] David Amiel and Cynthia Blitz. 2022. Computer Science Teacher Capacity: The Need for Expanded Understanding. *International Journal of Computer Science Education in Schools* 5 (2022). https://doi.org/10.21585/ijcses.v5i4.151

[4] Charoula Angeli. 2005. Transforming a teacher education method course through technology: Effects on preservice Teachers' technology competency. *Computers & Education* (2005), 383–398. https://doi.org/10.1016/j.compedu.2004.06.002

[5] Charoula Angeli and Ioannis Ioannou. 2015. Developing secondary education computer science teacher's technological pedagogical content knowledge. *European Journal of Educational Sciences* 2 (2015). https://doi.org/10.19044/ejes.v2no2a2

[6] Charoula Angeli and Nicos Valanides. 2005. Preservice teachers as ICT designers: An instructional design model based on an expanded view of pedagogical content knowledge. *Journal of Computer-Assisted Learning* (2005), 292–302. https://doi.org/10.1111/j.1365-2729.2005.00135.x

[7] Charoula Angeli and Nicos Valanides. 2009. Epistemological and methodological issues for the conceptualization, development, and assessment of ICT–TPCK: Advances in technological pedagogical content knowledge (TPCK). *Computers & Education* (2009), 154–168. https://doi.org/10.1016/j.compedu.2008.07.006

[8] Austin Cory Bart, Javier Tibau, Eli Tilevich, Clifford A. Shaffer, and Dennis Kafura. 2017. BlockPy: An Open Access Data-Science Environment for Introductory Programmers. *Computer* 50, 5 (2017), 18–26. https://doi.org/10.1109/MC.2017.132

[9] Austin Cory Bart, Eli Tilevich, Clifford A. Shaffer, and Dennis Kafura. 2015. BlockPy: From Interest to Usefulness in a Block-based, Educational Environment. *IEEE Blocks and Beyond Workshop* (10 2015). https://doi.org/10.1109/BLOCKS.2015.7369009

[10] Volkan Kukul Filiz Kalelioğlu, Yasemin Gülbarah. 2016. A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing* 4 (2016), 583–596.

[11] National Governors Association Center for Best Practices and Council of Chief State School Officers (NGA). 2010. Common core state standards for mathematics. *Common Core State Standards* (2010). Retrieved April 8, 2020 from https://corestandards.org/wp-content/uploads/2023/09/Math_Standards1.pdf

[12] K–12 Computer Science Framework. 2016. . Retrieved April 8, 2020 from http://www.k12cs.org

[13] Cory Gleasman and ChanMin Kim. 2020. Pre-Service Teacher's Use of Block-Based Programming and Computational Thinking to Teach Elementary Mathematics. *Digital Experiences in Mathematics Education* 6 (01 2020). https://doi.org/10.1007/s40751-019-00056-1

[14] James G. Greeno. 2006. Learning in Activity. In R. K. Sawyer (Ed.). *American Psychologist* (2006), 79–96.

[15] James G. Greeno. 2015. Commentary: Some Prospects for Connecting Concepts and Methods of Individual Cognition and of Situativity. *Educational Psychologist* 50 (07 2015), 248–251. https://doi.org/10.1080/00461520.2015.1077708

[16] James G. Greeno and Middle School Mathematics Through Applications Project Group. 1998. The situativity of knowing, learning, and research. *American Psychologist* 53 (01 1998), 5–26. https://doi.org/10.1037//0003-066X.53.1.5

[17] Shuchi Grover. 2021. 'CTintegration': A Conceptual Framework Guiding Design and Analysis of Integration of Computing and Computational Thinking into School Subjects. *EdArXiv* (2021). https://doi.org/10.35542/osf.io/eg8n5

[18] Timothy Hurt, Eric Greenwald, Sara Allan, Matthew A. Cannady, Ari Krakiwski, Lauren Brodsky, Melissa A. Collins, Ryan Montgomery, and Rena Dorth. 2023. The computational thinking for science (CT-S) framework: operationalizing CT-S for K–12 science education researchers and educators. *International Journal of STEM Education* 10 (2023). https://doi.org/10.1186/s40594-022-00391-7

[19] Ioannis Ioannou and Charoula Angeli. 2015. *Technological Pedagogical Content Knowledge as a Framework for Integrating Educational Technology in the Teaching of Computer Science.* Springer US, Boston, MA, 225–237. https://doi.org/10.1007/978-1-4899-8080-9_11

[20] Matthew J. Koehler and Punya Mishra. 2009. What Is Technological Pedagogical Content Knowledge? *Contemporary Issues in Technology and Teacher Education* 9 (2009), 60–70.

[21] Matthew J. Koehler, Punya Mishra, and William Cain. 2013. What is Technological Pedagogical Content Knowledge (TPACK)? *Journal of Education* (10 2013).

[22] Brian W. Lane, Terrie M. Galanti, and X. L. Rozas. 2023. Teacher Re-novicing on the Path to Integrating Computational Thinking in High School Physics Instruction. *Journal for STEM Education Research* (05 2023). https://doi.org/10.1007/s41979-023-00100-1

[23] Marcia C. Linn, James D. Slotta, Hiroki Terashima, Elisa Stone, and Jacquie Madhok. 2010. Designing Science Instruction using the Web-based Inquiry Science Environment (WISE). *Asia-Pacific Forum on Science Learning and Teaching* 11 (01 2010).

[24] Elena Macrides and Charoula Angeli. 2018. Investigating TPCK through music focusing on affect. *The International Journal of Information and Learning Technology* 35 (2018), 181–198.

[25] Punya Mishra. 2019. Considering Contextual Knowledge: The TPACK Diagram Gets an Upgrade. *Journal of Digital Learning in Teacher Education* (04 2019).

[26] Punya Mishra and Matthew J. Koehler. 2006. Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. *Teachers College Record* 108 (06 2006), 1017–1054.

[27] M. Niess, R. Ronau, K. Shafer, S. Driskell, S. Harper, C. Johnston, C. Browning, S. Özgün Koca, and G. Kersaint. 2009. Mathematics teacher TPACK standards and development model. *Contemporary Issues in Technology and Teacher Education* 9 (2009), 4–24.

[28] Margaret L. Niess, Emily H. van Zee, and Henry Gillow-Wiles. 2010. Knowledge growth in teaching mathematics science with spreadsheets: Moving PCK to TPACK through online professional development. *Journal of Digital Learning in Teacher Education* 27 (2010).

[29] Oregon Department of Education. 2010. 2010 Oregon Mathematics Standards. *Common Core State Standards* (2010). https://www.oregon.gov/ode/educator-resources/standards/mathematics/Documents/ccssmath.pdf

[30] Christine Redecker. 2017. European Framework for the Digital Competence of Educators. *Publications Office of the European Union* (2017). https://doi.org/10.2760/159770

[31] Jan Roddy. 2021. The Impacts of Integrating Interdisciplinary, Introductory Computer Science in High School Courses. *Graduate Student Theses, Dissertations, & Professional Papers* (2021).

[32] Milad M. Saad, Aziz M. Barbar, and Suzanne A.R. Abourjeili. 2012. Introduction of TPACK-XL: A Transformative View of ICT-TPCK for Building Pre-Service Teacher Knowledge Base. *Turkish Journal of Teacher Education* (12 2012), 41–60.

[33] Roger W. Schvaneveldt, Francis T. Durso, and Donald W. Dearholt. 1989. NETWORK STRUCTURES IN PROXIMITY DAT A. *THE PSYCHOLOGY OF LEARNING AND MOTIVATION* 24 (1989).

[34] Lee S. Shulman. 1986. Those who understand: Knowledge Growth in Teaching. *Educational Researcher* 15 (02 1986), 4–14.

[35] Lee S. Shulman. 1987. Knowledge and Teaching: Foundations of the new reform. *Harvard Educational Review* 57 (02 1987).

[36] CSTA K-12 Computer Science Standards. 2017. . Retrieved April 8, 2020 from https://www.csteachers.org/page/standards

[37] CSTA K-12 Computer Science Standards. 2017. CSTA K-12 Computer Science Standards. *https://www.csteachers.org/page/standards* (2017).

[38] Suherlan Suherlan. 2023. Digital Technology Transformation in Enhancing Public Participation in Democratic Processes. *Technology and Society Perspectives* 1 (3 2023), 10–17. https://doi.org/10.61100/tacit.v1i1.34

[39] David Weintrop and Uri Wilensky. 2019. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education* 142 (2019). https://doi.org/10.1016/j.compedu.2019.103646

[40] Jeannette M. Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[41] Jeannette M. Wing. 2008. Computational thinking and thinking about computing. *Philosophical Translations of The Royal Society A* 366 (2008), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

[42] Jeannette M. Wing. 2010. Computational Thinking: What and Why? *thelink.* (2010).

[43] Charlotte Woods. 2007. Researching and developing interdisciplinary teaching: towards a conceptual framework for classroom communication. *Higher Education* 54 (2007), 853–866. https://doi.org/10.1007/s10734-006-9027-3

[44] Aman Yadav, Chris Stephenson, and Hai Hong. 2017. Computational thinking for teacher education. *Commun. ACM* 60, 4 (April 2017), 55–62.