

COMBINING BLOCK-BASED PROGRAMMING WITH ROBOTICS KITS TO SUPPORT A MIDDLE
SCHOOL COMPUTING CURRICULUM

By

Bernard Yett

Dissertation

Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

May 12, 2023

Nashville, Tennessee

Approved:

Gautam Biswas, Ph.D.

Corey Brady, Ph.D.

Ákos Lédeczi, Ph.D.

Alan Peters, Ph.D.

Xenofon Koutsoukos, Ph.D.

Copyright © 2023 Bernard Hamilton Yett III
All Rights Reserved

ACKNOWLEDGMENTS

I would like to start by thanking my advisor and dissertation chair Dr. Gautam Biswas. Dr. Biswas has advised me since I joined his research group in 2018, aiding in my progression as a researcher and educator. I also want to thank my initial advisor, Dr. Xenofon Koutsoukos, for his assistance at the beginning of my graduate schooling. Thanks too to Dr. Ákos Lédeczi, who often supported me with opportunities to work with his lab on curriculum development and cybersecurity interventions. I extend this thanks to the remaining two members of my committee - Dr. Alan Peters (who guided much of my progress as an educator as well) and Dr. Corey Brady.

Thanks to all of the friends and mentors I have gained while at Vanderbilt. I would first highlight Dr. Kaz Kawamura for his invaluable perspective as an educator while I served as his Teaching Assistant for 3 years. Thanks also to Caitlin Snyder and Dr. Nicole Hutchins for their friendship and direct assistance on both this work and countless other papers and ideas. Dr. Dimitrios Boursinos, Dr. Charlie Hartsell, Tim Krentz, and Ibrahim Ahmed deserve a great deal of appreciation for keeping me sane with sports and other activities through the years. Gordon Stein and I collaborated on virtual robotics environments and had great discussions on teaching and research. Other friends, collaborators, and co-workers include Ningyu, Hamid, Dr. Witulski, Caleb, Dr. Sternberg, Devin, Marian, Ram, Grant, Gayathri, Anabil, Allison, Naveed, Shitanshu, Dr. Karsai, Feiyang, Sami, Dawit, Joyce, Mona, Bikram, Eduardo, Tim D., Grayson, Clayton, Win, Brian, Prof. Beck, and Celestine.

Outside of Vanderbilt, I have had an immensely powerful support system. Thanks to my parents – Mary Kay and Ben - and my in-laws - Bill and Jerome - for their willingness to drive to Nashville from Texas, the lifetime supply of H-E-B coffee, and for everything else you have done for me. My mom in particular had to put up with a lot of background noise from traffic while I double-dipped on walking and talking to her. Thanks to my sister Elizabeth for the constant encouragement, outside perspective and assistance, and for being my first student many years ago. Thanks to my grandparents Rita and Joe for the constant treats, prayers, and insight. Thanks to Brian, Bryce, Dev, David, Jordan, Chase, Andrew, Micah, and others for the conversations and gaming and providing some semblance of work-life balance. Thank you to the cats - Fiona and Lefty - for truly never allowing me to feel alone. Thank you to the entire Scheurich family for beach trips and wedding weekends. There are too many other friends, cousins, aunts, uncles, and more to list but thank you all nonetheless. I love you all.

Thank you to Mr. J.M. and his students for their enthusiasm and willingness to directly contribute to and participate in this work. I would not have been able to do this without all of you. Thank you to Dr. Sabatto for giving me the chance to join the faculty at TSU and gain valuable experience as an educator and person. Thanks to all of my own students for their understanding while I worked on this as well.

In closing, a very special thank you to my amazing wife Amanda. She has been my staunchest supporter through everything, from the many years in graduate school to the many long days and nights finishing this work. She proofreads, she surprises me with brownies, she makes me laugh, and she reminds me that I can actually do this. Thank you for always encouraging me even when I was not making much progress. Thank you for the TV watching, soccer playing, coffee walking, traveling, and everything else we do together. I love you and I thank God for you and all of these great people in my life.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	xi
1 Introduction	1
1.1 Motivation	1
1.2 Approach	2
1.3 Research Challenges	3
1.4 Research Contributions	4
1.5 Organization	5
2 Related Work	7
2.1 K-12 CS Education	7
2.1.1 Foundational Approaches	8
2.1.2 Recent Developments and Future Directions	10
2.1.3 Assessments	12
2.1.4 Curriculum Design	18
2.1.4.1 Robotics	19
2.1.4.2 Cybersecurity	22
2.1.4.3 Networking	27
2.1.4.4 Co-Design	29
2.1.5 Learning Analytics (and Other Educational Analysis Approaches)	30
2.1.5.1 Analysis of Progress of Student Learning	34
2.2 Learning Environments and Educational Robotics	35
2.2.1 Key Features	35
2.2.2 Role of Robotics	38
2.2.3 Scaffolding	40
2.3 Past Implementations	43
2.3.1 Complete Environments	43
2.3.2 Role of Robotics	46
2.3.3 Scaffolding	48
2.4 Critical Summary	49
3 Research Plan	52
3.1 Overview of Research Problem	52
3.2 Research Overview	53
3.3 Research Questions	55
3.4 Summary of Chapter	56
4 Methods	58
4.1 Curriculum Design	58
4.1.1 Module 1 - Introduction to NetsBlox and CT	61
4.1.2 Module 2 - Loops, Variables, and Custom Blocks	62
4.1.3 Module 3 - Conditionals and Lists	62

4.1.4	Module 4 - Messages, Networking Introduction, and RPC's	63
4.1.5	Module 5 - Robot Driving, Networks, and Network Security	64
4.2	Learning Environment	66
4.2.1	The NetsBlox Environment	66
4.2.2	The RoboScape Environment and Physical Robotics Platform	67
4.3	Assessments	69
4.3.1	Pre-Post Assessments	69
4.3.1.1	Survey	69
4.3.1.2	CT	70
4.3.1.3	Networking and Network Security	70
4.3.2	Embedded Assessments	70
4.4	Data Collection and Processing	71
4.4.1	Assessments	71
4.4.2	Surveys	72
4.4.3	NetsBlox Projects	72
4.4.3.1	Project Files	73
4.4.3.2	NetsBlox Logs	73
4.5	Study Demographics and Description	74
4.6	Summary of Chapter	76
5	Results	77
5.1	Research Question 1 - How did student performance, engagement, and attitudes measured using pre-post-assessments and surveys change as a result of our intervention?	77
5.1.1	Pre-Post-Test Analysis of Content Questions	77
5.1.2	What were the pre- to post-survey changes of students?	81
5.2	Learning progression of students in CT, networking, and security-related assessments (RQ2)	86
5.3	Strategy progression (RQ3)	102
5.4	Summary of Chapter	119
6	Discussion	121
7	Conclusions	126
7.1	Contributions	126
7.2	Limitations and Future Work	127
A	Appendix A - List of Publications	130
B	Appendix B - Full Summative Assessments	132
B.1	CT Assessment	132
B.2	Networking + Security Assessment	140
B.3	Post-Survey	144
C	Appendix C - Module Examples	147
C.1	Module 2	147
C.2	Module 3	155
C.3	Module 4	157
D	Appendix D - Additional Results	159

References **164**

LIST OF TABLES

Table	Page
2.1	Tennessee Standards Related to the Curriculum 23
2.2	CSTA K-12 Computer Science Standards Related to the Curriculum 23
2.3	K-12 Computer Science Framework Standards Related to the Curriculum 23
2.4	Characteristic Features of Co-Design[Roschelle et al., 2006] 30
4.1	Reasoning codes, listed in order from highest to lowest reasoning levels. 73
4.2	Selected actions related to either general programming behaviors or specifics of the projects being completed. 75
4.3	Demographic information of the $n = 48$ students who were full participants in the Spring 2022 semester-long study. Students were free to choose multiple options for race, so the total for those classifications will sum to $> 100\%$ 76
5.1	Changes from pre- to post-test - Computational Thinking Overall and Categories. p-scores determined via two-tailed t-test. Negative effect size implies a decrease from pre- to post-test and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d 79
5.2	Significant ($p < 0.05$) changes from pre- to post-test - Computational Thinking Individual Questions. p-scores determined via two-tailed t-test. Negative effect size implies a decrease from pre- to post-test and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d 79
5.3	Changes from pre- to post-test - Networking and Security Overall and Categories. p-scores determined via two-tailed t-test. Negative effect size implies a decrease from pre- to post-test and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d 80
5.4	Significant ($p < 0.05$) changes from pre- to post-test - Networking Questions. p-scores determined via two-tailed t-test. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d 80
5.5	Significant ($p < 0.05$) changes from pre- to post-test - Security Questions. p-scores determined via two-tailed t-test. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d 80
5.6	Summary of abbreviations used in upcoming tables related to survey data. 82
5.7	Changes from pre- to post-survey on a category-by-category level largely corresponding with Table 5.6. "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. p-scores determined via two-tailed t-test. Negative effect size implies a significant decrease from pre- to post-survey and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d 83
5.8	Significant ($p < 0.05$) changes from pre- to post-survey on a question-by-question level. p-scores determined via two-tailed t-test. Negative effect size implies a significant decrease from pre- to post-survey and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d 83
5.9	Spearman correlations are displayed as one measure of the relationship between a student's pre-post-test performance in the main areas of CT, networking, and security. Positive correlations ≥ 0.4 are in bold and blue text. 85
5.10	Number of occurrences of key constructs throughout the various modules of the curriculum. Multiple constructs may occur within the same sub-module. Total number of sub-modules provided for reference 89
5.11	Assessment performance over time broken down by key constructs. Results are presented as "Mean (SD)". "N/A" indicates that the concept was not present during that particular assessment. Scores were normalized to allow for easier comparisons across assessments. 90

5.12	Statistics related to each of the presented linear regression plots.	94
5.13	Spearman correlations are displayed as one measure of the relationship between a student's pre-test performance and their performance throughout the remainder of the intervention. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate pre-test component, and each row corresponds to a separate assessment category from later in the intervention.	96
5.14	Spearman correlations are displayed as one measure of the relationship between a student's post-test performance and their performance on other assessments during the intervention. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate post-test component, and each row corresponds to a separate assessment category from earlier in the intervention.	96
5.15	Spearman correlations are displayed as one measure of the relationship between a student's early CT performance and their performance and participation during networking and security modules and assessments. Each row corresponds to a separate CT-related assessment, and each column corresponds to a metric from later in the intervention. This early CT performance is based on the Module 2 and 3 project submission scores. Positive correlations ≥ 0.4 are in bold and blue text.	98
5.16	Spearman correlations are displayed as one measure of the relationship between project performance and student performance on related topics both before and after each respective project. Different projects have different associated concepts. M2, M3, M4 are short for Module 2, Module 3, Module 4. Dashes indicate that scores were not relevant for that particular project. Each column corresponds to a separate project, and each row corresponds to a related metric compiled from assessments before or after that project for "Through" and "After" respectively. Positive correlations ≥ 0.4 are in bold and blue text.	99
5.17	Spearman correlations are displayed as one measure of the relationship between a student's pre-test performance and either attitudes (see Table 5.6) or actions. "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate pre-test component, and each row corresponds to a metric from later in the intervention. Additional survey categories can be found in Appendix D	100
5.18	Spearman correlations are displayed as one measure of the relationship between a student's pre-survey performance (see Table 5.6) and their performance throughout the remainder of the intervention. Only selected survey categories with at least one relevant correlation $\geq 0.4 $ are presented for the sake of space. "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate pre-survey category, and each row corresponds to a metric from later in the intervention. Further correlations are included in Appendix D	100
5.19	Spearman correlations are displayed as one measure of the relationship between a student's pre-survey performance and their attitudes (see Table 5.6) and actions. Only selected survey categories with at least one relevant correlation $\geq 0.4 $ are presented for the sake of space. "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. Each column corresponds to a separate pre-survey category, and each row corresponds to a metric from later in the intervention. Positive correlations ≥ 0.4 are in bold and blue text.	101
5.20	Counts (percentages) for each action category during each of the four evaluated projects.	103
5.21	Transition probabilities of students' NetsBlox actions during the Module 1 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.	105
5.22	Transition probabilities of students' NetsBlox actions during the Module 2 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.	107

5.23	Transition probabilities of students' NetsBlox actions during the Module 3 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.	108
5.24	Transition probabilities of students' NetsBlox actions during the Module 4 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.	110
5.25	Selected DSM Results - Project 1 to Project 2	117
5.26	Selected DSM Results - Project 2 to Project 3	117
5.27	Selected DSM Results - Project 3 to Project 4	117
5.28	For each survey, we present the likelihood of a student's response falling under the appropriate reasoning type. It was possible for one response to indicate more than one type of reasoning, so rows will not sum to 1. Dashes indicate that no responses were coded as that reasoning type for that survey.	118
5.29	p-score (Effect Size) shown for each type of reasoning as we transition from survey to survey throughout the intervention. Negative effect size corresponds with a decrease from the earlier survey to the later survey and is indicated by red text. However, for these reasoning types, a negative effect size would be desirable. Bold rows indicate a moderate ($d \geq 0.5 $) or greater effect size as measured by Cohen's d.	118
5.30	p-score (Effect Size) shown for each type of reasoning as we transition from survey to survey throughout the intervention. Negative effect size corresponds with a decrease from the earlier survey to the later survey and is indicated by red text. Bold rows indicate a moderate ($d \geq 0.5 $) or greater effect size as measured by Cohen's d. "N/A" indicates that the survey combination in question was not applicable for that reasoning type.	118
D.1	Assessment performance over time broken down by key concepts. Results are presented as "Mean (SD) / Max". "N/A" indicates that the concept was not present during that particular assessment. This table contains the pre-test through module 3 - see Table D.2 for the remaining results.	159
D.2	Assessment performance over time broken down by key concepts. Results are presented as "Mean (SD) / Max". "N/A" indicates that the concept was not present during that particular assessment. This table contains module 4 through the post-test - see Table D.1 for the initial results.	160
D.3	Spearman correlations are displayed as one measure of the relationship between a student's grade (7th or 8th), gender (male or female), or race (only listing those with $\geq 16.67\%$ of the student population) and their performance throughout the remainder of the intervention. Positive correlations ≥ 0.4 are in bold and shown in blue text, while negative correlations ≤ 0.4 are in bold and shown in red text. Each column corresponds to a separate grade, gender, or race, and each row corresponds to a metric from the intervention.	160
D.4	Spearman correlations are displayed as one measure of relationship between a student's grade (7th or 8th), gender (male or female), or race (only listing those with $\geq 16.67\%$ of the student population) and their attitudes (see Table 5.6), actions, and attendance. "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. Negative correlations ≤ 0.4 (or in the case of Tardies, a positive correlation that indicated more tardies and is generally considered to be a negative situation) are in bold and shown in red text. Each column corresponds to a separate grade, gender, or race, and each row corresponds to a metric from the intervention.	161
D.5	Spearman correlations are displayed as one measure of the relationship between a student's pre-test performance and their post-test attitudes (see Table 5.6). "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. Each column corresponds to separate component of the pre-test and each row corresponds to a post-survey result.	162

D.6	Spearman correlations are displayed as one measure of the relationship between a student's pre-survey responses (see Table 5.6) and their performance throughout the remainder of the intervention. Only selected survey categories with at least one relevant correlation $\geq 0.4 $ are presented for the sake of space. "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. Positive correlations ≥ 0.4 are in bold and shown in blue text. Each column corresponds to separate component of the pre-survey and each row corresponds to a metric from throughout the intervention.	162
D.7	Spearman correlations are displayed as one measure of the relationship between a student's pre-survey performance and their attitudes (see Table 5.6). "General" refers to attitudes towards school in general, while "Specific" refers to attitudes towards this course or STEM specifically. Each column corresponds to a separate pre-survey category, and each row corresponds to a metric from later in the intervention.	163
D.8	Spearman correlations are displayed as one measure of the relationship between a student's early CT performance and their performance and participation during networking and security modules and assessments. Each column corresponds to a separate CT-related assessment, and each row corresponds to a metric from later in the intervention. This early CT performance is based on the CT components of Module 2 and 3 assessments. Positive correlations ≥ 0.4 are in bold and shown in blue text.	163

LIST OF FIGURES

Figure	Page
2.1 Different LA-related frameworks and models [Elias, 2011]	31
3.1 Overview of Research	53
4.1 Task model categorizing our Action Categories into Solution Assessment, Solution Construction, and Solution Independent	74
5.1 Starting from the top left and progressing in a clockwise order, these plots represent Overall, Networking, Security, and CT performance. Linear regression was applied to student performance on assessments at different points in the curriculum. M1, M2, etc. refer to Module 1, Module 2, etc. If a subject area was not present during assessments for a module, that module was not included in the progression. Increasing shape size corresponds with a larger number of students finishing with the corresponding percentage of points. All scores prior to the post-test were included in the regression to then predict post-test performance, as shown via the black line. Post-test scores are colored red to indicate this separation. Pre- and post-tests were identical and are shown as identical shapes. The classroom final, a distinct summative assessment, is given a distinct shape.	88
5.2 Number of subsections of a module that were related to each concept of interest. M1, M2, etc. refer to Module 1, Module 2, etc.	89
5.3 Performance on a concept during each module as related to maximum possible performance. Total is included for comparison	91
5.4 Starting from the top left and progressing in a clockwise order, these plots represent Variables, Loops, Encryption, and Conditionals performance. Linear regression was applied to student performance on assessments at different points in the curriculum. M1, M2, etc. refer to Module 1, Module 2, etc. If a subject area was not present during assessments for a module, that module was not included in the progression. Increasing shape size corresponds with a larger number of students finishing with the corresponding percentage of points. All scores prior to the post-test were included in the regression to then predict post-test performance, as shown via the black line. Post-test scores are colored red to indicate this separation. Pre- and post-tests were identical and are shown as identical shapes. The classroom final, a distinct summative assessment, is given a distinct shape.	92
5.5 Starting from the top left and progressing in a clockwise order, these plots represent Messaging, Evaluation, Testing and Debugging, and Topology performance. Linear regression was applied to student performance on assessments at different points in the curriculum. M1, M2, etc. refer to Module 1, Module 2, etc. If a subject area was not present during assessments for a module, that module was not included in the progression. Increasing shape size corresponds with a larger number of students finishing with the corresponding percentage of points. All scores prior to the post-test were included in the regression to then predict post-test performance, as shown via the black line. Post-test scores are colored red to indicate this separation. Pre- and post-tests were identical and are shown as identical shapes. The classroom final, a distinct summative assessment, is given a distinct shape.	93
5.6 Markov chain model of students' NetsBlox actions during the Module 1 project.	111
5.7 Markov chain model of students' NetsBlox actions during the Module 2 project.	112
5.8 Markov chain model of students' NetsBlox actions during the Module 3 project.	113
5.9 Markov chain model of students' NetsBlox actions during the Module 4 project.	114
B.1	132
B.2	133
B.3	134

B.4	135
B.5	136
B.6	136
B.7	137
B.8	138
B.9	139
B.10	140
B.11	141
B.12	142
B.13	143
B.14	144
B.15	145
B.16	146
B.17	146
C.1	147
C.2	147
C.3	148
C.4	148
C.5	148
C.6	This Debugging task was included with Module 2 (Part 1 of 2)	149
C.7	This Debugging task was included with Module 2 (Part 2 of 2)	149
C.8	This is the project task associated with Module 2	150
C.9	This is an example of a strong implementation of the Module 2 project	151
C.10	This is the stage output of a strong implementation of the Module 2 project	152
C.11	This is an example of a weaker implementation of the Module 2 project	153
C.12	This is the stage output of a weaker implementation of the Module 2 project	154
C.13	This is an example student-facing document for Module 3 (Part 1 of 2)	155
C.14	This is an example student-facing document for Module 3 (Part 2 of 2)	156
C.15	This is the project task associated with Module 4	157
C.16	This is an example of a strong implementation of the Module 4 project	157
C.17	This is an example of a weaker implementation of the Module 4 project	158

CHAPTER 1

Introduction

1.1 Motivation

In recent years the K-12 standards for Science, Technology, Engineering, and Math (STEM) and computer science (CS) at the national [Seehorn and Clayborn, 2017; K-12, 2016] and state (for example, Tennessee's [of Education, 2018]) levels have been revamped in order to introduce more comprehensive and advanced topics at various stages of the curriculum. These standards are joined by specific courses such as AP Computer Science Principles (AP CSP) [K-12, 2020] that have a similar focus. Key skills in the areas of computational thinking (CT), networking, and cybersecurity need to be introduced to students starting at a young age to prepare them for high school, college, and eventually for the 21st century workforce [Crumpler and Lewis, 2019].

In particular, it is important to increase awareness of privacy and security issues, especially when it pertains to an individual's own data [Geyamallika and Reddy, 2019]. These skills are also highly applicable across a wide variety of fields, with a focus on information technology management, electronics engineering, computer engineering, and telecommunications [Hoffman et al., 2011] but with a continuously growing list of related areas. Therefore, it is necessary to improve upon existing approaches for increasing awareness of these areas from an early age. Obvious needs include comprehensive curricula that are tailored to address these topics, and assessments that can accurately measure the progress of students. However, there are still questions about how such curricula may be best delivered in K-12 computing and technology classrooms, requiring further study and evaluation to develop successful interventions. The end result is an exciting area for research with multiple potential paths to explore.

Many aspects must be considered when preparing instructional materials covering advanced concepts for K-12 students:

- The curricular material should be age-appropriate and intuitive for those students to work with while maintaining their interest.
- Scaffolding should be provided by creating abstractions that focus on the primary ideas while removing irrelevant detail. Additionally, students should be introduced to a topic area via situated learning experiences.
- Providing physical artifacts that students can interact with may provide opportunities for students to put their learning to practice. One example of this is through educational robotics [Nugent et al., 2016].

- Assessments and other data collection and analysis should be applied to identify student misconceptions and generate insights into their learning behaviors. This will further lead to students overcoming their difficulties and learning the content more effectively.

There are few K-12 teachers who have a strong background and experience in teaching the new CS curricula. This is especially true of our primary areas of interest such as cybersecurity and networking [K-12 Cybersecurity, 2021]. One option to familiarize them with the concepts and practices to be taught and to help them in delivering the curricular material is to use a co-design approach [Wu et al., 2020; Kelly et al., 2019]. Co-designing the curricula with teachers allows for them to tailor materials to meet the needs of themselves and their students. This includes collaborating with teachers as they strive to assess and aid students as they progress through their work. The curriculum includes teaching of advanced concepts, formative assessments that support student learning, and problem solving tasks that help students gain a deeper understanding of curricular material. This provides opportunities for collecting rich data of student learning and problem solving behaviors. Creating appropriate assessments that have both research significance and can be used as grades within a classroom is an important part of this. Additionally, collecting and interpreting multiple sources of data can lead to improved knowledge about how students are working.

1.2 Approach

We fulfilled the needs of researchers, teachers, and students through the development of a comprehensive, hands-on curriculum that is compatible with national and state standards by progressively introducing students to basic computing, networking, and cybersecurity concepts. A good curricular implementation must take prior knowledge - which can influence student perceptions [Walker et al., 2016] - into account in order to ensure the correct level of complexity. Additionally, it can help to ground ideas of cybersecurity and networking within topics students are more familiar with. It would also be futile to jump right into these difficult concepts without scaffolding this introduction to students through unplugged activities or simulations. Following from ideas of situated learning [Anderson et al., 1996], we can focus on supplementing abstract instruction with concrete examples to address these concerns. Finally, assessments have to meet the needs of teachers, students, and researchers. They should indicate components of the intervention in need of refinement from the research perspective, allow teachers to provide feedback to students, and help students to recognize the topics they are struggling with. This includes both traditional assessments at the beginning and end of the intervention, but also formative assessments and collecting and analyzing other data sources such as programming actions. All of these challenges required substantial development time and provide avenues for further research.

It is also necessary to accurately and concretely portray the concepts we expect students to become fa-

miliar with. Cybersecurity and networking in particular are both abstract topic areas, and concepts in these fields can be difficult to properly express to students. Therefore, to supplement the curriculum itself, we implemented a learning environment composed of two primary components. A robotics platform provides students with a tangible method for knowledge application. We believe the hands-on experience with robots reinforced what they have learned in an engaging manner. Additionally, the block-based programming environment allowed for scaffolding of student learning and provided many affordances. Using these tools, abstract concepts were introduced to students within authentic contexts. They were able to see and experience the concepts in an easy-to-understand yet accurate manner. Overall, this environment aimed to support the learning and problem solving skills of the students participating in the intervention.

1.3 Research Challenges

A factor that is often ignored in analysis approaches is the prior knowledge of students. Incorporating the prior knowledge of students along with the myriad of other factors impacting student learning is difficult to accurately accomplish without additional aid. It is unlikely that many students in K-12 have significant experience with the more abstract topics such as networking and cybersecurity. Prior knowledge of more common subject areas such as CT and programming may impact their ability to pick up related topics more quickly. Both pre-tests and formative assessments will help teachers (and students themselves) realize common difficulties and misconceptions encountered by students. The next step is then to provide opportunities for students to learn these concepts before moving on to more advanced concepts. This leads to the research challenge of designing curriculum and assessments to facilitate this process.

While carrying out interventions in CS and STEM with younger students, it is common to assess those students at the beginning of and at the conclusion of the intervention. This is often done through qualitative methods, such as a self-assessment completed by the student with questions focusing on areas such as task value and motivation [Linnenbrink-Garcia et al., 2018]. Also popular are quantitative methods, such as a test related to the specific subject area of the intervention. Other options include grading the programming projects or similar artifacts completed by students, performing analysis on the specific programming actions students take, or evaluating students on the way they describe these actions verbally to themselves or to a partner [Tang et al., 2020]. The problem with these approaches is that there are rarely clear connections between components of the intervention and assessments, leaving researchers to guess at aspects that benefited students versus those that did not. This leads to difficulties in evaluating the effectiveness of the curriculum and in providing feedback to students. Overall, the challenge is the lack of comprehensive assessment and analysis techniques in this networking and cybersecurity context for K-12 students.

1.4 Research Contributions

We first summarize the contributions that will be fulfilled by this dissertation:

- Item 1 - Creation, refinement, and implementation of a curriculum for introducing K-12 students to a variety of CT, networking, and cybersecurity topics. This also included providing tools to teachers such that they could implement the curriculum on their own.
- Item 2 - Extension of previously implemented assessments while contributing new ones to form a novel, comprehensive means of evaluating students' progress through the curriculum.
- Item 3 - Providing additional information to students, teachers, and researchers to ease their burdens during implementation, including insight gained from formative assessments, programming actions, and attitudinal surveys.
- Item 4 - Development of systematic methods to evaluate the effectiveness of our curriculum through learning analytics and results from assessments.
- Item 5 - Improved understanding of the progressions of student learning and strategy usage.

It was necessary to properly build and evaluate the curricular materials to match the needs of the teachers and students. This was accomplished through the co-design process. We hope that this work achieves a step towards understanding the CS learning progression students undertake throughout middle and high school by mapping to the previously mentioned standards. In particular, networking has become a key part of these standards, but the subject had not been incorporated into the classroom or other interventions at the same level as CT and cybersecurity to reflect this importance. By using CT and cybersecurity as supplementary concepts while keeping networking as the primary focus, we hoped to contribute a novel curriculum addressing this perceived weakness.

Furthermore, summative tests and surveys were connected with formative assessments to track student learning and help the teacher provide appropriate feedback. Additionally, a coordinated focus on relating specific aspects of the curriculum to a variety of assessments positively contributed to our understanding of student learning. The evaluation of formative assessments exposed students' problem solving behaviors, allowing teachers and researchers to track students' progress and tailor instruction to support students' needs. The target was curricula with an initial emphasis on basic computing concepts and CT before moving on to more advanced networking and cybersecurity concepts. This allowed us to ensure a reasonable baseline for all students. Meanwhile, the hands-on robotics platform provided a conduit for student learning of those advanced concepts while maintaining student engagement. We introduced the difficult and advanced concepts

of networking and cybersecurity through a hands-on situated set of activities on a robotics platform to provide students with authentic learning activities.

Other contributions were realized by undergoing a co-design process with a teacher familiar with this subject area. The same teacher taught the students who underwent the intervention and participated in the co-design process. This allowed for misconceptions of students to be identified and addressed from the start [Grover and Basu, 2017], while grounding all concepts and curricular materials within the capabilities of the students. Co-design for curriculum planning within the realm of STEM has been theoretically grounded within the work of [Kelly et al., 2019]. They emphasize pedagogical and technological professional development for teachers and a design-thinking framework as the core components. Undergoing such a process in the past has proven to be effective in our areas of interest including CT [Wu et al., 2020] and block-based programming [Grover et al., 2020].

Taking into consideration the concerns raised previously and the contributions to be made, we have formulated and investigated the following questions.

- Research Question 1 - How did student performance, engagement, and attitudes measured using pre-post-assessments and surveys change as a result of our intervention?
- Research Question 2 - What was the learning progression of students in CT, networking, and security-related assessments, as well as key constructs that bridged these broader categories?
- Research Question 3 - What was the strategy progression of students based on their programming actions during projects and their open-ended questionnaire responses?

1.5 Organization

The following are the key sections of the remainder of this dissertation. Chapter 2 presents an extensive literature review on the topics related to this dissertation, including K-12 CS education, the role of robotics within the contexts of CS, cybersecurity, and networking, and scaffolding of abstract or difficult concepts. Also included are assessment methods, designing appropriate curricular materials, co-design, and learning analytics. Later content focuses on theory and applications of learning environments. Chapter 3 outlines the area of research and expands upon the research problems and questions that were addressed through this work. Next, Chapter 4 outlines our approach and methods. This includes new developments and assessments that were needed, the learning environment, and studies that were implemented. It also describes the analytics techniques applied towards answering the research questions. The next chapter - 5 - presents the bulk of the findings. In particular, we describe student performance on different components of the assessment suite, analyze the progression of student learning of key concepts, and highlight changes in programming and

reasoning behaviors. A discussion of the results is then displayed in Chapter 6. Chapter 7 summarizes the contributions of this dissertation and proposes future work to be done by myself or others. Appendices show a list of publications (A) that have already been published, the full pre-post tests and surveys (B), examples of different materials from the curricular Modules (C), and additional results tables (D) to supplement those of Chapter 5. Following this is a compilation of references used throughout the proposal.

CHAPTER 2

Related Work

Based on the problems identified and motivations generated in Chapter 1, three primary areas were selected to review for previous theories, findings, and implementations. First, a general focus on K-12 CS Education, beginning with both foundational approaches (Section 2.1.1 and more recent developments (Section 2.1.2. Following this is a review on assessing student learning (Section 2.1.3). Next, Section 2.1.4 explores past approaches to putting together a generic CS or CT curriculum. This leads into approaches specific to robotics, cybersecurity, and networking in Sections 2.1.4.1, 2.1.4.2, and 2.1.4.3 respectively. The last component surveyed was in the field of learning analytics and other approaches to analyzing educational data 2.1.5.

The second primary area is learning environments, which have been proven to be effective across a wide variety of settings. This includes our primary areas of interest - CT and robotics. The initial focus is on what components make up a learning environment and why such components have been chosen in Section 2.2.1. Next is a discussion on the combination of educational robots with learning environments in Section 2.2.2. The section concludes with recommendations for appropriately scaffolding CS and robotics learning through the use of learning environments in Section 2.2.3.

The final area for review was particular learning environments already created for similar purposes as our approach. Though no brand-new environments were created in this work, the review framed the combination of techniques that were integrated into the finished product. A first set of observations were progressions from one learning environment to the next in Section 2.3.1. After that were specific examples detailing the combination of educational robots with learning environments in Section 2.3.2. Finally, scaffolding implementations related to at least one area of interest are presented in Section 2.3.3. At the end of the chapter, a critical summary of all related work is presented in Section 2.4. This emphasizes the need to address the identified open problems in the field, which further motivates this research.

2.1 K-12 CS Education

CS can be an exciting and engaging subject if taught appropriately and tailored to the students involved. Much of the focus should be on grounding new concepts within areas that students are familiar with and keeping them interested in the field rather than worrying about preparation for university-level courses or future jobs [Armoni and Gal-Ezer, 2014]. A key component of CS is that it is used world-wide; collaboration between students and researchers all over the globe can only further progress the field. In particular, upon examination of a variety of K-12 implementations from around the world [Lockwood and Mooney, 2017],

findings covered many of the tools we are familiar with or are considering incorporating into our approach. At the junior high level, block-based programming and hands-on activities have been the most popular and successful. Block-based programming environments should be further highlighted as a valuable tool for simplifying the initial cognitive load that would be required to learn a syntactically complicated text-based language. They can either be the primary learning environment with an advanced feature set [Broll et al., 2017] or be used as a launching pad to prepare students for those text-based languages that will be necessary to learn at some point in a computer scientist's journey [Dorling and White, 2015]. When including such tools within a CS curriculum, it is necessary to consider multiple factors. These include adapting complexity of the platform and curriculum to account for a broad range of prior knowledge and cognitive development of students [Knobelsdorf and Vahrenhold, 2013] and improving the accessibility of CS as a whole through improved tools for different groups [Das et al., 2020]

A curriculum should involve a progression through CS topics of increasing complexity [Grover, 2017] - this is particularly effective when combined with a comprehensive assessment package for evaluating different aspects of student learning. The assessments can be applied to noticing shortcomings of the intervention as well as supplying feedback to students about misconceptions. Feedback from students can also provide some insight into how they perceive CS and programming [Lakanen and Isomöttönen, 2015]. The primary themes students responded with were syntax and language features, the general nature of programming work, and CT and problem solving. Of further note, more experienced users tended to recognize the importance of algorithmic thinking and logical reasoning, capturing the essence of CS as a skill that has cross-domain applications. Sentance and Csizmadia [Sentance and Csizmadia, 2017] present the results of a survey of over 300 Computing teachers in the UK in 2014. The key recurring themes mentioned by these teachers largely conforms with the highlights of our thought process for curriculum development. These include unplugged activities as a low-stakes means of introducing concepts to students, the value of collaborative learning and scaffolding overly complex programming tasks, grounding CS learning within hands-on activities, and the important role of CT in such a curriculum. Robotics is an oft-applied medium for accomplishing all of the above, either through an unplugged approach for younger students [Rand et al., 2018] or using hardware kits such as LEGO Mindstorms [Lawhead et al., 2002]. Of course, other hardware sources can offer similar benefits depending on the specific curricular constructs [Feaster et al., 2013]. All of these options provide hands-on learning opportunities to students as an engaging means of gaining CS knowledge and confidence.

2.1.1 Foundational Approaches

A review of CS education from 2009 through 2013 was conducted in [Garneli et al., 2015]. The most common approaches discovered from this analysis involved a visual (such as block-based) programming

language, either a game-based or physical tool (such as robotics) as the educational context, and a problem or project-based approach as the instructional method. Though this does largely correspond with our own previous findings and other literature, it is difficult to draw too many conclusions without a measure for the success rate of each approach when compared to some of the less common alternatives. Upon examination of a model curriculum for K-12 CS developed in the early 2000's [Tucker et al., 2003], it can be seen that not all that much has changed in the core knowledge pieces that students should learn. Instead, the main iterations have been in terms of pedagogical techniques, the components of each key overarching category, and at what age students should be introduced to specific concepts. Concerns about actually implementing these curricular ideas in school - and finding teachers with appropriate knowledge and training to successfully present the materials to students - existed at this time and are still a roadblock today. Reinforcing this point using results from a survey of 9693 K-12 US principals published in 2016 [Wang et al., 2016], over 75% of the principals indicated that their schools offered no courses focused on CS with programming.

A six-year project in Georgia [Guzdial et al., 2014] sought to rectify both a general lack of CS education at the K-12 level but also to broaden participation in the field by targeting multiple points along the “pipeline” (the different ages students learn different CS skills). Their approach was largely undergone outside of standard schooling, instead opting for summer camps and after-school or weekend programs. These interventions were found to significantly impact the ability of students to correctly identify and apply loops, conditionals, variables, events, message sending, and code interpretation. The already far-reaching study above was combined with many others to achieve a comprehensive vision of CS education at the K-12 level [Hubwieser et al., 2014]. This work drew many of the same conclusions as the other papers considered here, with an emphasis on improving teacher preparedness, the inclusion of programming for teaching CS, and increasing opportunities for CS education in K-12. An additional consideration of interest raised here was including computational thinking (CT) as a core component of any CS curriculum.

Jeannette Wing developed a list of what exactly goes into CT (and what does not) in [Wing, 2006], which has become widely accepted and applied ever since. Though it can be an integral component of teaching and learning CS, it is a distinct skill as well with applicability across every domain. By including CT within a CS course or intervention, we are able to move beyond a focus on only programming or only computers in order to help students achieve a broader understanding. Barr and Stephenson [Barr and Stephenson, 2011] go beyond the definition laid out by Wing and look at actively integrating CT within K-12 curricula. They highlight ways that the various CT concepts and capabilities can be found in CS, as well as in other STEM subjects and even areas like social studies and language arts. This reinforces the necessity of incorporating CT into K-12 education, as the skills involved translate across domains and lead to more effective and efficient students.

In a review inspired by Wing's thoughts on CT, Grover and Pea [Grover and Pea, 2013] took on the challenge of observing practical implementations as compared to the conceptual thoughts from [Barr and Stephenson, 2011]. In particular, they highlighted tools and environments that had been successful in terms of fostering CT, including familiar options such as Scratch, Alice, and Arduino. They also pave the way to the present day by hypothesizing about large gaps requiring empirical inquiries. These include various factors of contemporary learning sciences such as debugging and scaffolding as well as using computing as a medium for teaching other subjects. Finally, we have an example [Catlin and Woollard, 2014] that also built upon the CT foundation of Wing through an integration with educational robots and the early work of Seymour Papert [Papert, 1980] involving Logo programming and Turtle robots. A recommendation is made for mapping the relationships between CT concepts and specific features of activities. The principles of Educational Robotic Applications (ERA) are also referenced [Catlin and Blamires, 2010] - these are divided into technology (Intelligence, Embodiment, and Interaction), student (Engagement, Sustainable or Lifelong Learning, and Personalisation), and teacher (Pedagogical, Curriculum and Assessment, Equity, and Practical) considerations.

2.1.2 Recent Developments and Future Directions

More recently, Parker and DeLyser [Parker and DeLyser, 2017] presented a detailed overview of the development of a K-12 CS framework [K-12, 2016]. The goal of the framework was to catch up to other disciplines by defining what students should know and what they should be able to do at different age levels. These are specified in terms of five concepts (computing systems, networks and the internet, data and analysis, algorithms and programming, and impacts of computing) as well as seven practices (fostering an inclusive computing culture, collaborating around computing, recognizing and defining computational problems, developing and using abstractions, creating computational artifacts, testing and refining computational artifacts, and communicating about computing). Other aspects of the framework include the need to continually update concepts and practices based on new changes in the CS field, and its purpose of informing those creating standards and curricula. One such set of standards at the national level are presented in [Seehorn and Clayborn, 2017]. By drawing upon the K-12 CS Framework, these standards provide a breakdown of each concept for various grade bands (K-2, 3-5, etc.). These breakdowns combine well-described standards with their related subconcepts (such as cybersecurity, hardware, software, etc.) as well as tying back into the practices and subpractices of the framework.

As another broad example, we turn to the AP Computer Science Principles (AP CSP) framework [K-12, 2020]. This is used as the basis for the AP CSP course that is gaining popularity across the United States for its role in preparing high school students to either continue into further CS study at the university

level or be more successful in other fields thanks to their CS background. Moving on from comprehensive frameworks, Webb et al. [Webb et al., 2017] present important questions in K-12 CS education as we continue to experience technological advances and new additions to the field. These include which specific skills and understanding should be taught, whether those skills and understanding are necessary for all (with some students only learning basic computing and technology literacy as an alternative), whether CS education should be compulsory for a certain age range, and what are appropriate pedagogical approaches considering various factors including age. They also comment on the need to balance assessments with curriculum and pedagogy - too much emphasis on assessment can lead to constraints on the other areas and decrease a student's ability to be creative and explore the learning environment.

One approach to addressing those balancing concerns is to simply perform post-intervention assessments based upon the programming projects themselves, using these results to examine student learning in a more natural way [Grover et al., 2018]. This allows us as teachers and researchers to promote learner agency and take advantage of the affordances of learning environments tailored for the particular age range of the students while still obtaining evidence of whether students improved in the designated subject areas. An open question was the potential and relative values of block-based and text-based programming environments for introducing CS to K-12 students. Initial results [Weintrop and Wilensky, 2017] were promising in either case. Further evidence is needed for understanding differences between options within the categories, the value of hybrid approaches combining block-based and text-based tools, and whether either environment type is significantly more beneficial to assisting struggling students in catching up to their peers.

CT was specifically highlighted in both the K-12 CS Framework [K-12, 2016] and the AP CSP Framework [K-12, 2020], and is therefore worthy of further specific exploration. Creativity in CS and programming was advocated for in [Grover et al., 2018] and is further developed in [Brady et al., 2020]. This implementation emphasizes debugging as a key component of CT within their programming environment and specifically as students created expressive yet computational visual effects. In addition to this example as well as the earlier work by Wing [Wing, 2006], Grover and Pea [Grover and Pea, 2018] provide a focus on CT within a variety of contexts. They emphasize the usage of programming as a strong tool to foster CT development while noting that other approaches can be effective as well. The biggest takeaway was the continued recognition of CT as a distinct and necessary skill set that relates to a wide variety of fields. Within the specific context of programming, a review [Buitrago Flórez et al., 2017] notes the power of this CT and programming combination when introduced to students starting at a young age. Findings supporting the combination include needs for students to be able to extract knowledge from large sources of data, learn complex systems in a scaffolded manner, and intuitively combine multiple layers of abstraction.

To somewhat broaden the scope and examine CT when integrated with other STEM subjects, we turn to

the C2STEM system [Hutchins et al., 2020b]. The system has been tuned to scaffold learning of CT and physics by high school students through an evidence-centered design approach. In addition to the presented results indicating the effectiveness of such a system, a highly transferable framework is laid out to aid in the application of the system to other domains such as CS. CT is normally considered to be a cognitive thinking process. However, many researchers still consider it in terms of the knowledge or skills gained when assessing student growth. An ongoing issue is the lack of consensus on a definition of CT. This has led to confusion on the constructs of assessing CT. A theoretical framework of learning and assessment would be needed in order to disentangle the learning of CT skills from domain-specific knowledge and otherwise capture student CT performance [Tang et al., 2020].

To focus briefly on cybersecurity and related areas, we turn to Cyber.org and the survey results they prepared related to exactly these topics [K-12 Cybersecurity, 2021]. Starting from a basic distinction, very few educators that were surveyed knew “a lot” about cybersecurity education, and they mentioned a relative lack of resources in this area as well. Implementations were often brief insertions into existing courses rather than the focus of a course or extracurricular activity. Only four percent of the elementary school teachers surveyed said their students had access to cybersecurity competitions or opportunities to be exposed to the basics of cybersecurity such as digital literacy. For our particular areas of interest, cryptography was basically never addressed, robotics at 30%, networks and the internet at 39%, and coding/programming at 46% in middle schools. Meanwhile, robotics, coding/programming, and hacking/data security are among the highest areas that middle school teachers believed their students would be interested in. A working group report from ITiCSE [Mountrouidou et al., 2019] emphasized the need for authentic, active learning activities to teach cybersecurity and particularly to achieve a more diverse populace of students interested in the field.

2.1.3 Assessments

A starting place for designing any type of assessment or system of assessments is the book “Rethinking Classroom Assessment with Purpose in Mind” [Earl et al., 2006]. The key approach taken here is to consider assessments with three potential purposes in mind - assessment for learning, assessment as learning, and assessment of learning. Assessment for learning would encompass areas such as formative or embedded assessments - those that occur during the learning progress of students and can provide an opportunity for teachers or researchers to observe and possibly address misconceptions or knowledge growth of students. Assessment as learning could take similar forms in terms of assessments occurring during an intervention, but with different goals in mind. These assessments would be geared towards engaging students in metacognitive processes that lead to increased understanding of the desired subject area as students reflect upon their own learning. Finally, assessment of learning is probably the most common, often accomplished through con-

ceptual or practical pre- and post-intervention examinations. However, even in this well-established realm careful consideration must be made when it comes to designing such assessments to fairly and accurately measure student learning.

Grover presents an approach to assessing students learning CT that relies upon the idea of using multiple complementary measures which capture both cognitive and noncognitive aspects of learning [Grover, 2017]. The idea is transferable across domains as a general plan for comprehensively capturing the process of student learning. An additional suggested assessment type - known as “preparation for future learning” - investigates the transfer of skills and knowledge from one domain to another. To further the implications of this work, as reported in this survey of assessments in the field [Tang et al., 2020], often only one method of evaluation is incorporated within interventions. This indicates that a comprehensive assessment approach that uses multiple methods at a variety of times along with providing information about future learning possibilities could provide much value to researchers, students, and teachers alike.

A fair amount of work has already been done in regards to developing and evaluating the effects of assessments as they relate to CS. Within the intersection of STEM, CT, and CS assessments, Grover examined the ability of students to converse using key ideas of CT and CS before and after a week-long robotics intervention [Grover, 2011]. The primary assessment tool was a pre-post interview attempting to capture the verbal CT expressions of the students. The reasoning behind this process was to observe the benefits of learning this particular branch of academic language. The unique combination of interview and prior experience was used to evaluate students and further the collective definition of what constitutes CT. Further work [Grover and Pea, 2013] reviews the body of research conducted in the field as of publishing. The primary observation related to assessments was that programming projects in one form or another were the primary tool for evaluating a student’s understanding of CT concepts. Papers led by Hutchins [Hutchins et al., 2018; Hutchins et al., 2020b] present an alternative pathway for designing a series of CS - with an emphasis on CT - assessments. The key to this approach is the integration with curriculum design. Assessments only result from the conclusion of the process after establishing the curriculum and identifying the most important components of each step that should be assessed. For these examples, CT was combined with physics as the primary domain before listing the concepts students should learn and be tested on. Pre- and post-tests were also combined with embedded assessments - as we will highlight later, this can be a very effective pairing as part of a comprehensive CS assessment package.

Basu has also contributed several papers within this domain, beginning with a shared paper with Grover [Grover and Basu, 2017]. They listed a set of constructs that students would learn upon participating in their intervention, then created assessments designed to measure the relevant focal knowledge, skills, and abilities (FKSAs) of students based on those constructs. The final set of assessment questions were mapped to one

or more target FKSAs. In the past we have taken a more post hoc approach to this mapping, and we could benefit from such a conscious identification of appropriate concepts and questions similar to this one or those presented by Hutchins [Hutchins et al., 2018; Hutchins et al., 2020b]. Another solo work from Basu [Basu, 2019] presented assessment methods based upon a multi-dimensional rubric used for evaluating block-based programming projects of students. The novelty of this approach was that both front-end project design and back-end evaluation of the complexity of programming constructs used were considered. Student proficiency could then be determined based on the total scores from a variety of categories falling under one of three descriptors - Overall Proficiency, User Experience, and Coding & CS Constructs.

One review and one intervention report provide insight into assessing CS within a robotics perspective. The review [Benitti, 2012] observes what assessments are commonly used as part of conducting a robotic intervention in a classroom, camp, or other setting. Outside of the obvious pre- and post-tests with comparisons made between the two in some way, qualitative results through observations of students either by the researchers or some automated system were sometimes used for additional data points. Finally, Witherspoon et al. [Witherspoon et al., 2017] used a set of analogous exams to assess student performance before and after they participated in a virtual robotics curriculum. The different versions were randomly assigned to students for the pre-test, then one of the other versions was assigned to the same student on the post-test. These tests incorporated questions aimed at evaluating a student's skills in developing algorithms and abstracts, evaluating algorithms, and identifying procedures for constructs such as iterative development. Student progress was also captured at various points during the intervention through evaluation of activity completion within a lesson as well as the use of formative quizzes.

Surveys can be a useful tool for evaluating the perceptions of students on a wide range of topics. One example includes questions designed to measure student attitudes towards engineering and their knowledge about engineering careers [Hirsch et al., 2007]. Martin et al. [Martin et al., 2012] used the results from mathematics-related surveys to predict future intent and disengagement of students in the field by applying multilevel hierarchical regression, finding that self-efficacy, valuing, and enjoyment (among others) were significant predictors. Weintrop and Wilensky [Weintrop and Wilensky, 2017] surveyed students using a 10-point Likert scale to examine their attitudes and perceptions towards CS. Four dimensions in particular were singled out - interest in future CS opportunities, perceived difficulty of CS, self-confidence in CS abilities, and enjoyment of CS. Attitudinal surveys are used to assess student opinions during an Introduction to Computer Security course at the college level [Deshpande et al., 2019]. In particular, students evaluated the impacts of the peer instruction implementation that was used as part of the course. This suggests the value in obtaining student outlook on pedagogical approaches taken - a more enjoyable classroom experience could lead to higher engagement and student learning. Weese and Feldhausen [Weese and Feldhausen, 2017] present an

interesting take on surveys through a self-efficacy instrument that they incorporated with a STEM outreach program for 5th-grade through 9th-grade students. They divide self-efficacy into four components related to problem solving, writing specific computer programs, methods for creating computer programs, and the impact of computer programming. Though their results were mixed, the evaluation approach shows a lot of potential for identifying strengths and weaknesses of a curriculum in terms of self-efficacy.

A final standalone work related to surveys but with many additional applications is presented in [Linnenbrink-Garcia et al., 2018]. One key aspect of the work is the compilation of many different survey components from outside sources along with an actual listing of the questions posed to students, providing a useful resource to myself and others interested in this area. Additionally, the authors present some benefits of a person-oriented approach to learning. They were able to investigate the combination of predictors at the individual level through surveys, resulting in identification of different profiles for different students and how they relate to different outcomes [Bergman et al., 2003]. This approach also has practical applications in the form of modeling motivation combinations. This can lead to closer reflections of interrelations between types of motivations within individuals, providing a new angle to researchers as they look for common combinations and the motivational typologies resulting from those combinations. Multi-variate analysis was applied for this research in order to accurately capture the motivations of students that require several facets to truly understand. This serves as another example of implementing and evaluating surveys in an advanced manner. Results underscored the importance of creating educational contexts that specifically support a student's value placed on school or a particular domain in school, as well as their self-confidence in their learning capabilities [Linnenbrink-Garcia et al., 2018].

Quantitative analysis in the form of student surveys could be supplemented with other methods like interviews or focus groups leading to a better understanding of the in-depth thinking processes of students [Tang et al., 2020]. A gap exists in the form of combining quantitative and qualitative assessments, leaving this investigation into the cognitive process of developing CT skills as an open problem. Another oft-ignored consideration involves performing a reliability and validity analysis of assessments and their results [Tang et al., 2020]. Some examples of reliability and validity evidence produced include putting together a panel of experts to evaluate assessments [Djambong and Freiman, 2016] or confirmatory factor analysis looking for supporting evidence that the assessments covered all relevant CT skills [Araujo et al., 2019].

Pre- and post-tests are a common choice for comparing student performance on related subjects before and after an intervention. Weintrop and Wilensky [Weintrop and Wilensky, 2017] incorporated pre- and post-tests in addition to the survey discussed previously. In this case, they used the tests to examine student results when using block-based or text-based programming, ultimately finding that students achieved higher scores through the block-based approach. In general, this suggests the possibility of combining traditional

assessments with surveys for a more complete picture of student learning. The most often used method for investigating the growth of CT skills involves traditional testing. In particular, that testing is often focused on the specific domain, programming environment, or both that were used throughout the intervention. However, it is unclear whether CT skills can really be regarded as quantifiable mastery of CT knowledge. Though it is easy to define CT in terms of programming concepts, this reliance can lead to a lack of connection or applicability to non-CS subjects [Tang et al., 2020].

One approach to adding a unique quality to the use of pre- and post-tests is through a control group [Ramachandran et al., 2016]. Though the tests themselves are hardly innovative in their relation to the intervention, a control group can provide context when evaluating the intervention that has often been lacking in our research. The different approaches to the intervention relate to a robotic tutor with either adaptive or on-demand (the control condition) feedback. Adaptive feedback was found to be effective for encouraging productive help-seeking behavior through the pre- and post-test results. Within the context of a robotics intervention comparing differences between virtual and physical implementations, general categories for student assessments are suggested [Liu et al., 2013]. These include algorithmic thinking, general programming, the syntax of the programming language used, and functionalities of the robot. However, the fact that half the problems were syntax related could be a negative component, as this does not necessarily reflect the actual knowledge gained by students during the intervention. Finally, Irgens et al. [Irgens et al., 2020] present an assessment implementation for a high school biology unit. One component of the implementation is the pre-post tests to assess the science and CT learning gains of students. They present case studies of a student who improved and a student who regressed on those tests, delving into the intervening lessons of the intervention in an attempt to explain the results.

Embedded assessments - the integration of student progress and performance within instructional materials [Wilson and Sloane, 2000] - have a lot of potential, largely in terms of combinations with pre- and post-tests or other results from throughout an intervention. For example, the preceding assessment implementation for a high school biology unit also includes a series of embedded assessments [Irgens et al., 2020]. As part of the case study process, verbal answers to embedded assessment questions were coded using an automated algorithm and are shown for the positive and negative gain students. This supplementary information helped to explain why these students performed in a particular way - a general trend of strong, well-reasoned answers are clearly visible for the student who engaged with the material and demonstrated learning gains, while the poorer performing student had only scattered answers of a similar caliber. An additional approach presented by Hutchins et al. [Hutchins et al., 2020b] includes embedded formative assessments created using evidence-centered design [Mislevy and Haertel, 2006] as a key component for supporting student learning within an open-ended learning environment. The design process for these assessments included starting with

the target domain before unpacking core ideas and concepts and relevant practices. The next steps were to create integrated domain maps from the unpacking results, to state the learning goals and expected proof of learning, and finally to design the assessment tasks.

Assessment systems can be used as a form of embedded assessments with more emphasis on benefits to the teacher in terms of rendering student performance more easily observable. Wilson and Sloane [Wilson and Sloane, 2000] present the Berkeley Evaluation and Assessment Research system. Intended goals include allowing teachers to assess student performance on key curricular components, to set standards of student performance and track their progress, and to intervene by providing feedback to students. Basawapatna et al. [Basawapatna et al., 2015] bring a similar approach to a specifically CS domain by creating REACT - Real-Time Evaluation and Assessment of Computational Thinking. In this case, the formative/embedded assessments are done throughout student learning using the REACT dashboard to analyze student projects as they work. These outputs were generated at the prompting of the middle school teacher using the system and allowed them to identify struggling students and offer assistance.

An additional option to supplement traditional testing is portfolio assessment [Tang et al., 2020]. This can be used to assess performance by systematically collecting and evaluating various products created by students in order to evaluate CT skills over time. On top of previously mentioned assessments in this area including [Hutchins et al., 2018; Hutchins et al., 2020b; Basu, 2019], Spikol et al. [Spikol et al., 2018] evaluated projects across five areas. These were Level of clarity, Independent thinking, Corresponds with plan, Does it work?, and Quality of solution. Grover, Basu, and Schank [Grover et al., 2018] created rubrics along five dimensions before applying them to projects created in both Scratch and App Inventor. These dimensions were: General factors, Mechanics of design, User experience, Basic coding and constructs, and Advanced coding and constructs.

An approach we have applied in the past which is effective but overly time consuming for the amount of data we will face is evidence-centered design (ECD) rubrics (though the principles of ECD are still relevant). ECD was introduced by Mislevy [Mislevy et al., 2003] as “an approach to constructing educational assessments in terms of evidentiary arguments”. Assessments should be considered in the following way under this framework:

- What are we measuring?
- How do we measure it?
- Where do we measure it?
- How much do we need to measure?

- How does it look?

Answering such questions before, during, and after assessment design ensures that they are serving the appropriate purposes within a curriculum and for all involved. Follow ups to this original work [Mislevy and Riconscente, 2005; Mislevy and Haertel, 2006] explored the different layers of ECD. The process involves gathering information about the domain, expressing needed assessments to match that information in progressively more detailed manners, creating the assessments, and presenting and scoring the assessments. Recent examples of applications of the ECD process to creating and scoring assessments in particular can be found in [Basu, 2019] and [Hutchins et al., 2018].

2.1.4 Curriculum Design

One of the first attempts at compiling and publishing a complete set of curricular guidelines for K-12 CS education was submitted by the ACM K-12 Task Force Curriculum Committee in 2003 [Tucker et al., 2003]. The goals of such a curriculum as put forth by the committee are to introduce fundamental CS concepts to students of all ages, to teach such concepts in an accessible way, to provide advanced materials such that interested students could progress further and increase preparedness for college and a career, and to increase CS knowledge for all students with an emphasis on those from underrepresented populations. Students should start by learning about the foundations of computer science throughout K-8 education, progressing through a wide variety of topics and learning objectives. The AP Computer Science Principles (AP CSP) course in particular has continued to be iterated upon up to the current year [K-12, 2020]. Key practices include computational solution design, algorithms and program development, abstraction in program development, code analysis, computing innovations, and responsible computing. These lead into the Big Ideas of creative development, data, algorithms and programming, computing systems and networks, and impact of computing.

An overview of CT in K-12 education [Grover and Pea, 2013] provides some insight into considerations for computing curriculum design. Such a curriculum should nurture the development of computational competencies in students while properly assessing student learning and providing an opportunity to transfer problem-solving skills from other domains into computing (or vice versa). Armoni and Gal-Ezer [Armoni and Gal-Ezer, 2014] answer questions regarding why CS should be taught, what topics should be covered, when CS should be taught, and to whom. CS at the K-12 level should primarily focus on exposing students to the field and encouraging them onward with engaging activities rather than immediately trying to mold them into professionals. It is also recommended to consider CS as the intersection of math, science, and engineering when deciding what to teach, along with the key abstract learning goals of increasing students' problem-solving competencies and theoretical knowledge base. Finally, an IEEE Computer Society report [Knobelsdorf and Vahrenhold, 2013] recommends a focus on learning objectives over topics in order to allow

for flexibility in teaching methods based on the needs of individual students. This can be difficult for us to observe as researchers, which is where the expertise of teachers through the co-design process can be an excellent resource. An additional note is that these authors suggest that attempting to begin with a curriculum designed for college students and converting it down for the younger students is unlikely to be the most effective. Instead, the curriculum could be designed from the bottom up by considering the abilities of the target students and designing learning objectives based on that analysis.

Weintrop et al. [Weintrop et al., 2016] present a possible alternative to restricting CT to just CS based on some of the flaws of CS at the K-12 level. In particular, students that are uninterested in CS or teachers that do not have the required background should still have access to learning or teaching CT. Feaster et al. [Feaster et al., 2013] present a more specific curricular example that attempts to provide interactive activities centered around teaching computing. Lectures are included in order to present the necessary background information to students. These could be supplemented or supplanted by situated learning, unplugged activities, or similar additions depending on the specific curriculum in use. This is followed by interactive activities, but in a structured way that encourages a steady build up. It begins from a first project that is more of a demonstration before concluding with a group project with minimal outside intervention. An additional concrete example was a CS course developed for middle school students called “Foundations for Advancing Computational Thinking” [Grover et al., 2015]. Somewhat uniquely for this age group, the course is available as a Massive Open Online Course (MOOC). The key features of the curriculum include a balance between cognitive, interpersonal, and intrapersonal elements, an encouragement of transfer and progression from block-based to text-based programming, a specific emphasis on common misconceptions in the field, and a comprehensive assessment suite featuring formative and summative assignments. Results indicated the potential for such a curriculum and platform when it comes to CS learning. There were reported improvements in the algorithmic thinking of students, the students were able to make the transition from Scratch to text-based programming successfully, and they obtained a more advanced understanding of computing. An additional offering related to this course was that a correlation was found between prior computing experience and mathematics as a predictor for learning outcomes. This adds to previous ideas about the possible benefits of related prior knowledge.

2.1.4.1 Robotics

One of the primary reasons for using robots as vessels for teaching CS, problem solving, critical thinking, and collaborative skills to students is the well-documented history of benefits to such approaches. Petre and Price [Petre and Price, 2004] observed and evaluated students participating in robotics competitions. They found that robots can lead to improved understanding of programming and engineering concepts by motivating

students to solve problems and learn topics that they previously considered impossible. Benitti [Benitti, 2012] performed a literature survey of robots in educational settings, finding proven positive results in areas such as computer programming and thinking skills. Nugent et al. [Nugent et al., 2010; Nugent et al., 2009] examined results from both a 40-hour intensive summer camp and a smaller 3-hour intervention drawing from the longer curriculum. The longer intervention was found to lead to increases in student learning across mathematics, geospatial concepts, engineering, and computer programming as compared to a control group not participating. Meanwhile, the shorter intervention had a positive effect on student attitudes and motivation.

A wide variety of considerations must go into designing and implementing a robotics platform and curriculum. McLurkin et al. [McLurkin et al., 2013] identify the value of a platform that is versatile enough to be used across multiple disciplines. They also emphasize the use of a development environment that is tailored to the intended users. A survey of relevant papers [Benitti, 2012] lists many findings related to this area. This includes providing adequate physical space for students to utilize, limiting the number of students per robot to two or three, and providing appropriate problems for students to solve along with specific guidance on the relation between activities with the robot and the concepts they should be learning. Additional valuable components are strong feedback from the learning environment and allowing students time for open-ended exploration. Giroto et al. [Giroto et al., 2014] espouse the need for embodied approaches to student learning made possible by robots as well as requiring students to explicitly reason about commands being sent to the robot. Lessons learned during these interactions could then be implemented within a feedback system to ultimately improve the problem-solving processes of students.

Barth-Cohen et al. [Barth-Cohen et al., 2018] present their findings related to navigating multiple representations. For our purposes, these could be a programming environment, physical or simulated robot, and the instructions for the task. Placing an emphasis on designing the complete system given the needs of students to process information across each representation can prevent an excessive amount of cognitive load on students. Catlin and Woollard [Catlin and Woollard, 2014] suggest caution when designing a robotics curriculum with CT in mind, emphasizing the importance of incorporating these ideas into a subject area that is well-known to students. This leads to a need for special care to be taken when trying to combine CT and difficult, abstract concepts such as cybersecurity and networking. Taking advantage of the robot platform itself is a final key consideration. Afari and Khine [Afari and Khine, 2017] discuss Lego Mindstorms robots in particular as an option that is versatile, appealing to students, and has a comprehensive suite of sensors and other necessary accessories. This can easily be extrapolated to the robots we have used in the past.

We should also consider the various platform components such as programming environments, custom or standard robots, and curriculum or exercise examples that have been previously created. These predecessors

can reinforce the general principles and benefits presented and further highlight important considerations to focus on during platform and curriculum design. One example comes from Touretzky et al. [Touretzky et al., 2013] as they varied the complexity of programming environment used with a robotics platform to encourage student learning of CT constructs such as variables, loops, and conditionals. Students began with a simple Kodu icon-based environment before progressing to Alice and Lego NXT-G implementations. Several exercises using Lego Mindstorms robots are presented in [Lawhead et al., 2002]. The exercises are mapped to concepts such as simple datatypes, booleans, mathematic expressions, functions, and recursion. Specific examples included programs to count the number of times touch sensors (whiskers in our implementation) are touched and play that many beeps, having the robot turn a specific number of degrees, or a maze navigation problem using the touch sensors and appropriate algorithmic thinking. Grover [Grover, 2011] presents a novel approach to incorporating CT beyond “just programming”, resulting in new ideas for CS curricula built around the basic CT language that students are already familiar with from other domains. Another suggestion was a means of evaluating an intervention based on improvements in the students’ abilities to converse about concepts covered in a technical manner.

McLurkin et al. [McLurkin et al., 2013] present a low-cost robot featuring a Python programming environment, a camera tracking system, and a full sensor suite including a gyroscope, accelerometer, wheel encoders, and light sensors. A graphical user interface provides easy access to sensor readings, though the Python shell presented could be overly complex for younger students. Another example involves the creation of Robot Virtual Worlds and a combination of physical and virtual VEX robots [Liu et al., 2013]. Though no comprehensive curriculum is presented, the idea of physical and virtual robots fulfilling similar roles is a powerful one - especially when also considering the similar effectiveness that was discovered for each platform. Rand, Sengupta, and Feil-Seifer [Rand et al., 2018] suggest and implement the use of unplugged, embodied activities as a means of conveying the ideas of robotics, cybersecurity, and programming to younger students in particular without getting bogged down with syntax and other features too advanced for these students. Activities such as these have a lot of utility in terms of keeping students engaged and providing a different look at abstract concepts that could lead to revelations by students. Also included in the work are suggested roles and activities for such an implementation. An additional attractive option is a pedagogical format involving short lectures, hands-on activities, and structured student worksheets [Nugent et al., 2009; Nugent et al., 2010].

In [Nag et al., 2013], robots are actually miniature satellites developed as a test bed through a partnership with the International Space Station. The authors present a competition framework built upon these robots, providing ideas for how to structure a competition to engage the most students possible and ensure that each group feels important to the overall results. Catlin and Woollard [Catlin and Woollard, 2014] use

Turtle educational robots to explicitly focus on CT skills. Their examples range from students observing and describing robot behavior, to programming a robot to mimic the behavior of a dog, to structure design to accomplish a spacecraft rescue mission, and finally to programming Turtle robots to go against their standard operation of only turning on the spot and instead turn via curved paths.

Burleson et al. [Burleson et al., 2017] present a dual approach incorporating learning environments with robotic tangibles. The platform features embodied instruction, where students are required to use their whole bodies to interact with the robots. Another feature is in programming the robots themselves, as they respond to commands provided to the students through fiducial markers (essentially QR or bar codes). Combining several key features of our intervention, researchers in [Chattopadhyay et al., 2020] present a curriculum for introducing college and pre-college learners to cybersecurity concepts and principles, networking topics, and educational robotics. They created step-by-step procedures for hands-on robot hacking procedures and mapped lessons to the IT2017 Curricular Knowledge Domains. They also compared results from their creation to those obtained after adapting a traditional cybersecurity curriculum via Khan Academy. Though aimed at an older audience, all of this provides support for our decisions and guidance on approaches to take.

2.1.4.2 Cybersecurity

The need for an increased emphasis on cybersecurity education has never been higher. Despite best efforts, there are still vulnerabilities to be addressed in hardware, software, and within networking protocols. Additionally, with each new technological innovation comes security risks - social media, cloud computing, smartphones, and autonomous vehicles are some examples that can be potentially attacked or hacked and cause devastating consequences as a result [Jang-Jaccard and Nepal, 2014]. At the university level, this has been addressed by a detailed set of curriculum guidelines listing the necessary knowledge areas to be covered as well as breaking those knowledge areas down into units, topics, and learning outcomes [Curricula, 2017]. At the K-12 level, seven standards within the full CSTA K-12 Computer Science Standards [Seehorn and Clayborn, 2017] are classified as falling under the subconcept of cybersecurity - passwords, real-world cybersecurity problems, physical and digital security measures, encryption, determining recommended security measures given a scenario, tradeoffs between cybersecurity implementations, and comparing means of protecting against unauthorized access. The AP Computer Science Principles Curriculum Framework [K-12, 2020] does not specifically mention cybersecurity, but does have an entire section devoted to the Impact of Computing and many “Essential Knowledge” components are related to safe internet practices and the dangers of cyber attacks.

To prepare for other needs of the curriculum and related materials, a comprehensive list of standards specific to cybersecurity and networking that could be covered in our work was created. We started with

Table 2.1 Tennessee Standards Related to the Curriculum

Identifier	Description
DC.4	Recognize and describe the potential risks and dangers associated with various forms of online communications (e.g., cell phones, social media, digital photos.)
ISA.8	Describe the rationale for various security measures when using technology.
CCP.7	Construct protocols that can be used to share information between people or devices.
CCP.8	Compare the relative strengths and weaknesses of unique protocols considering security, speed, and reliability.
CCP.9	Create models of networks that include packets and domain name server (DNS).
CCP.10	Identify steps to ensure security measures are in place to safeguard online information.
CCP.11	Create ciphers to encrypt data that can be transferred between users.
CCP.12	Explain how encryption can be used to safeguard data that is sent across a network.
CCP.33	Demonstrate an understanding of digital security.

Table 2.2 CSTA K-12 Computer Science Standards Related to the Curriculum

Identifier	Description
2-NI-04	Model the role of protocols in transmitting data across networks and the Internet.
2-NI-05	Explain how physical and digital security measures protect electronic information.
2-NI-06	Apply multiple methods of encryption to model the secure transmission of information.

the Tennessee standards [of Education, 2018] as shown in Table 2.1. With current development focusing on Tennessee schools, these were the primary considerations for our work. Still, related ideas from other frameworks are applicable here, including those from the CSTA K-12 Computer Science Standards [Seehorn and Clayborn, 2017] as shown in Table 2.2. This was also further extended with a few examples from the K-12 Computer Science Framework [K-12, 2016] as shown in Table 2.3. With this framing, we will look at how teaching cybersecurity has been implemented and assessed.

Mirkovic et al. [Mirkovic et al., 2015] present a means of evaluating interventions via a five-step approach before applying the approach to three cybersecurity interventions. The interventions were intended to address

Table 2.3 K-12 Computer Science Framework Standards Related to the Curriculum

Identifier	Description
6–8.Networks and the Internet.Network Communication and Organization	Computers send and receive information based on a set of rules called protocols. Protocols define how messages between computers are structured and sent. Considerations of security, speed, and reliability are used to determine the best path to send and receive data.
6–8.Networks and the Internet.Cybersecurity	The information sent and received across networks can be protected from unauthorized access and modification in a variety of ways, such as encryption to maintain its confidentiality and restricted access to maintain its integrity. Security measures to safeguard online information proactively address the threat of breaches to personal and private data.

a variety of open problems related to cybersecurity, including:

- Students had only superficial knowledge of cybersecurity
- Students lacked practical skills in cybersecurity
- Students were not interested in cybersecurity
- Teachers lacked the requisite practical skills to convey cybersecurity information to students
- Underrepresented populations have even less access to the materials required, leading to lower interest
- Students considered cybersecurity to be a complex field
- Students did not recognize the importance of cybersecurity

Areas of intended growth following such an intervention could be sorted into the broad categories of Skills, Interest and awareness, Learning, and Impact. Suggested methods for measuring Skills growth include the use of security scenarios, embedded assessments, analysis of programming projects, and self-assessments. Self-assessments can also be applied to the Interest and awareness category. Knowledge features an overlap with Skills in the use of scenarios and some concept quizzes, while a survey could be used to measure Impact. The three presented interventions were not able to cover all of these evaluations - it appears to be quite infeasible to do so in most settings. However, the suggestions are sound and some of the overhead can be covered by borrowing from previous implementations and collaborating with other experts in the field. In addition to creating new content for sharing, spearheading such a collaboration would ease the burdens upon all involved and lead to improvements in the quality and quantity of cybersecurity interventions [Mirkovic et al., 2015].

An additional method for addressing these open problems is through competition-based learning. Namin, Aguirre-Munoz, and Jones [Namin et al., 2016] present such a framework that harnesses the benefits of collaboration to increase the cybersecurity knowledge and interest of students as they face-off in realistic scenarios. This work was specifically intended to address perceived shortcomings related to the behavioral, social, and ethnographical aspects of cybersecurity as well as a missing focus on the educational perspective from previously designed competitions. To that end, TracerFIRE [Anderson et al., 2015] was created and centered around the teaching of complex problem-solving skills as related to cybersecurity. It was intended for undergraduate and graduate students already in the field who would also be assisted by existing cybersecurity experts, so there is not a direct translation to the middle school or even high school level. The incorporation of open-source tools to supplement our platform as well as the focus on realistic and competitive scenarios

to ground student learning within that context are still sound concepts for inclusion at any age level. Responses in this particular study were encouraging within the areas of self-reported cyber skill knowledge and confidence, each of which significantly increased over time.

Work done in [Mishra et al., 2017] presents a first effort at incorporating cybersecurity within existing AP CSP courses in New York. They took the already developed course and identified areas where cybersecurity first principles [Payne et al., 2016] could be inserted. One example of this is within the networking lessons of AP CSP. This is of particular interest to us considering our research directions, though suggested additions were also prepared for each other module of the curriculum.

One of the more common approaches to teaching cybersecurity in the past has been through game-based learning techniques. An example implementation occurred at Purdue University Northwest (PNW) [Jin et al., 2018; Jin et al., 2017] as they held GenCyber [Ladabouche and LaFountain, 2016] camps for high school students. The GenCyber program [Ladabouche and LaFountain, 2016] aims to address many of the concerns raised by Mirkovic et al. as previously listed [Mirkovic et al., 2015]. The games at this particular GenCyber implementation were designed to both mimic popular video games for engagement purposes - a common theme in educational games [Olano et al., 2014] - while also incorporating relevant cybersecurity topics such as secure online behavior, cybersecurity first principles, social engineering (the art of manipulating others into revealing confidential information), and cyber defense techniques. A post-survey was administered to assess the self-efficacy, motivation, and perceived knowledge of students. The average rating for each question was greater than four on a scale from 1 to 5. However, without a pre-survey for comparison or other quantitative or qualitative measures as reported by the authors, it is difficult to draw further conclusions.

Another example is SecurityEmpire, a multiplayer game developed by Olano et al. at the Cyber Defense Lab at the University of Maryland, Baltimore County (UMBC) [Olano et al., 2014]. The game required users to practice information assurance (IA) while avoiding other security breaches and building up a green energy company. The primary purpose of this implementation was to address two perceived threats to cyber safety - the tendency of users to act without considering the consequences and a lack of awareness of IA concepts. It focused on high-level user behaviors over technical programming and skill development, but was more complex than other implementations such as Control-Alt-Hack [Denning et al., 2013] that simply used cybersecurity vocabulary as a game component. Once again, a post-survey was used to measure how the sample population of high school students responded to the game. The average was around 4.0 out of 5.0 for the majority of positive statements related to engagement and the game itself. However, a pre-survey for comparison on certain statements is missing or unreported, and nothing specifically related to cybersecurity was reported. It may not have been possible to include such aspects given the time constraints of the intervention, but SecurityEmpire and the PNW games both seem to be lacking the assessments necessary

to determine if game-based learning was successful when used to teach cybersecurity to students.

There have also been efforts to build upon AP CSP with the inclusion of cybersecurity content in a CS curriculum [Javidi and Sheybani, 2018]. The emphasis was on co-design and collaboration with teachers in order to reach a larger set of students and to create activities that impact cybersecurity awareness, education, and training in equal measure. They propose a novel curriculum combined with tools for an engaging learning experience as the means of accomplishing these goals. Of course, cybersecurity can be taught by a variety of means depending on the goals of the intervention creators and the needs of the target population. Researchers at Universitat Ramon Llull in Barcelona, Spain present the full pedagogical background for an Internet of Things (IoT) cybersecurity course [Sánchez et al., 2020]. The course is framed around the need for students to develop specific competencies related to security concerns, but an additional design decision was to similarly emphasize generic competencies such as learning skills and the ability to collaborate within a multidisciplinary group to accomplish tasks. Students began with basic procedural knowledge units and the creation of IoT network architectures in order to establish a knowledge base for networking and cybersecurity before moving on to more advanced scenarios. The course culminated with a real-world problem presented by a company in a related field that required the skills of the entire multidisciplinary group to sufficiently solve. This course in particular meshes well with the recent rise in IoT and the need to secure such devices [Hamza et al., 2020]. Preparing students to prevent attacks on these devices even from the perspective of being more aware of the dangers at a household consumer level can greatly reduce overall vulnerabilities. As with computer networks, IoT devices are vulnerable to similar passive and active attacks as we aim to address in our curriculum.

As an additional curricular example at the high school level closer to our desired age range, Trabelsi and Barka [Trabelsi and Barka, 2019] present a five-unit curriculum for teaching information security to high school students. The curriculum specifically addresses malicious programs, computer passwords, phishing and other similar schemes, computer networks, and internet services plus application filtering. Example assessment questions and exercises for each section are provided along with the desired learning outcomes. The outcomes are also aligned with sections of the recommended CS guidelines for the UK, Canada, and the UAE. This inspired us to undertake a similar process with national and state standards. Coffman-Wolph and Gray [Coffman-Wolph and Gray, 2019] present some simple and short exercises to introduce computer security to middle school students. The activities cover substitution ciphers (such as Caesar Cipher), modern encryption methods through public-key exchange, and the basics of social engineering. Though not a standalone curriculum on its own, these lessons could be used as a means of first presenting these and similar topics to students before progressing into the hands-on aspects with the robotic programming platform. A further approach [Yuan et al., 2019] emphasizes novel pedagogical techniques for teaching cybersecurity,

with a focus on guided inquiry learning in a collaborative setting. Curricular materials were developed to facilitate student progression from understanding cybersecurity concepts as a baseline and then on to analyzing and applying critical thinking skills in order to solve the presented activities. They provided specific survey categories and an example module implementation for firewalls as evidence of the results of their curriculum development framework.

A recent survey examined all cybersecurity education papers from either SIGCSE or ITiCSE conferences between the years 2010 and 2019 [Švábenský et al., 2020]. Using such a literature review, we can gain some insight into common practices outside of those previously examined in more detail. The most common cybersecurity topic areas were secure programming and development, network security and monitoring (tying into our desired increase in networking emphasis), cyber attacks, human aspects, cryptography, and authentication and authorization. Almost all of the reviewed interventions involved college-level students with only seven of the full 71 aimed at K-12 students. A similarly low number of examined papers use various artifacts produced by students (screen recordings, log actions, project or assignment submissions) when compared to subjective surveys or objective tests. Some concerning statistics are that only 22 of the 71 papers produced any materials that others can still access and the average citations of each paper is only 2.5 after removing self-citations. This indicates a lack of collaboration between researchers and a possible failure in building upon the theories and implementations of the past. On a more positive note from our perspective, the fact that only a few studies are aimed at K-12 students or incorporated more than surveys or exams for assessment techniques leaves our research path largely unexplored. Another survey [Mouheb et al., 2019] instead targeted cybersecurity curriculum design specifically. Results included three main activity areas for cybersecurity exercises to provide realistic experiences to students - Prepare, Defend, and Act. Phases for these activities could then be categorized as Prevention, Detection, and Response.

2.1.4.3 Networking

Networking has clearly been identified as an important subject, as the K-12 Computer Science standards have “Networks and the Internet” as one of only five concepts recommended to be introduced to students starting in Kindergarten and progressing through the end of high school [Seehorn and Clayborn, 2017]. Computing Systems and Networks are one of the “Big Ideas” of the AP Computer Science Principles Curriculum Framework released for the 2020-2021 academic year [K-12, 2020], with an emphasis on how networks facilitate the transfer of data. An ACM Sigcomm conference established a first set of recommended topics for the undergraduate level that could be used as a baseline for determining prerequisite knowledge for younger students. Among these topics were the basics of physical networks, concepts such as circuit switching vs. packet switching, the means of connecting multiple networks together, protocols such as Transmission Con-

tronic Protocol (TCP) and User Datagram Protocol (UDP), relevant programming skills, and the need for strong cybersecurity practices to protect networks [Kurose et al., 2002]. There are also the networking topics mixed in with cybersecurity ones throughout the standards outlined in Section 2.1.4.2.

A possible solution to the issue of abstracting away some of the required technical skills involved the creation of an interactive laboratory-based approach for undergraduate students [Sarkar, 2006]. The labs allowed students to gain first-hand experience in configuring actual computer networks, showcasing the benefits of practical experience with server hardware and software in order to facilitate student learning. Topics included networking hardware and software, Local Area Networks (LANs), topologies, protocols, network administration, and TCP/IP, with different practicals targeting different items from this list. Though we cannot achieve a perfect correlation between a networking curriculum with robotics for junior high students and an undergraduate course for computer networking, there are some takeaways to draw from [Sarkar, 2006]. Assignments should be hands-on and have a clear, useful purpose. They should be challenging enough to keep students interested, while also providing for loose time constraints so they can work at their own pace and experiment. Results indicated the potential for knowledge growth using such a method.

Through a collaboration between the School of Computing at Clemson University and the Department of Mathematics and Computing at Lander University [Feaster et al., 2013], a “serious toy” was developed for use with a middle school curriculum entitled “Learning Networks, Protocols, and Algorithms”. The “serious toy” in this case is a small embedded hardware platform to give students the hands-on experience that supplements such a curriculum. The curriculum begins with a series of lectures to introduce a definition for a network including a framing within common networks such as postal service. Next is an introduction to protocols while still staying within these easier to understand physical realms. The following section introduces algorithms and the differences between centralized and distributed computing. Then, the toys are used for a trio of demonstrations related to using sensors, how a serial bus network works, and finally for students to create their own networks considering constraints of different peer-to-peer protocols. Students were assessed through both a survey related to CS interest, content understanding, and whether they enjoyed the curriculum as well as a content quiz on the topics that were covered. The same survey and quiz was used before and after the intervention for consistency.

Chen [Chen, 2003] identifies a constructivist approach to teaching computer networks at the university level. Despite the age difference, many of her presented curricular ideas appear as if they would translate well to a younger audience. Examples include objectifying various network topologies (ropes for bus topology, key rings for ring topology, post-it notes for star topology, etc.), the use of children’s toys as hardware to symbolize the various hardware components of a network, and hands-on group activities without formal grades to encourage collaborative learning. Alternatively, Yuan and Zhong [Yuan and Zhong, 2009] take

an active learning approach to teaching networking through a hands-on series of labs featuring a variety of simulation-based tools. They target networking basics as well as some lessons related to common attacks and security measures. It appears that the majority of students responded favorably to this hands-on approach, though the results are not presented particularly clearly.

2.1.4.4 Co-Design

Outside of sources already mentioned [Javidi and Sheybani, 2018][Knobelsdorf and Vahrenhold, 2013], Roschelle et al. [Roschelle et al., 2006] present early work in establishing key components of the co-design process. These components are displayed in Table 2.4 and were a key source of inspiration for this dissertation. Matuk et al. [Matuk et al., 2016] furthered our understanding of co-design by combining a theoretical framework with a practical implementation in four areas - discussing physical artifacts, reacting to scenarios, customizing prototypes, and writing user stories. The emphasis was on the process itself to aid future research via documentation. This is further emphasized in [Kelly et al., 2019] through a distinction between time for work and time for development during the co-design process as well as splitting co-design into phases such as generating and refining. Though our final implementation necessarily evolved into a more free-flowing format, these formal approaches provided a baseline to build upon. A framework closer to our approach was presented by Grover et al. [Grover et al., 2020]. Teachers were first exposed to a formal training period, then given an opportunity to teach curricular materials to students. These experiences then culminated in a collaborative co-design period to refine those materials.

For co-design as practical implementation, we first turn to [Holstein et al., 2019] for a detailed recounting of creating a learning analytics tool. They found that - among other things - all teachers involved wanted to be able to see student's thought processes and determine who was truly stuck or who just needed a small nudge. They then produced prototypes such as an analytics dashboard and other in-class technology deployments. Hutchins [Hutchins, 2022] progressed more specifically in the area of teacher dashboards via a co-design process. The end result was meant to support teachers as they provided evidence-based instruction to students undergoing the complex process of problem-based learning. The co-design process itself was split into needs analysis, professional development, and planning periods. Lin and Brummelen [Lin and Van Brummelen, 2021] focuses on applying co-design to the creation of curricular materials in the field of K-12 AI. A Value-Sensitive Design approach was used to ultimately produce three full projects using pre-existing tools. Finally, structured and unstructured co-design components were combined in [Scornavacco et al., 2022]. Teachers and researchers would propose changes to existing materials based off of feedback from students. We undertook a similar process on a smaller scale as part of this work.

Table 2.4 Characteristic Features of Co-Design[Roschelle et al., 2006]

Co-design takes on a concrete, tangible innovation challenge
The process begins by taking stock of current practice and classroom contexts
Co-design has a flexible target
Co-design needs a bootstrapping event or process to catalyze the team's work
Co-design is timed to fit the school cycle
Strong facilitation with well-defined roles is a hallmark of co-design
There is central accountability for the quality of the products of co-design

2.1.5 Learning Analytics (and Other Educational Analysis Approaches)

The field of learning analytics (LA) aims to answer a variety of questions about a given course or other educational intervention. These are primarily related to the effectiveness of the intervention and its ability to meet the needs of the students. Of course, these could be answered by traditional pre- and post-tests or other assessment techniques, but these structures are rigid in purpose and cannot be easily applied towards on-the-fly adjustments. The value in LA comes from the ability to quickly analyze data and to perform corrective actions or offer feedback to students based on findings. Some early approaches to accomplish this process were compiled together by Elias [Elias, 2011] as shown in Figure 2.1. The final refinement on the far right of the image describes the LA process. It requires not only gathering and analyzing the appropriate data, but also using the results to predict future problems and determine appropriate steps to address them. Much of the process should ultimately be automated, but people still have a role in interpreting results within the context of the particular students and intervention. Considering this, LA ultimately requires computers, theory, people, and organizations all working together [Elias, 2011]. Unlike related areas such as educational data mining or academic analytics, and despite the wide variety of analysis techniques available, the one key facet of LA is a focus on student learning above all else. Specific techniques were examined by Clow [Clow, 2013]. They included predictive modelling, social network analysis, usage tracking (data generated from a learning management system or similar), content and semantic analysis, and recommenders. One concept tying all of these use cases together is the value a student places on their own learning. If incorporated properly, LA can allow a student to take responsibility for this learning and best take advantage of the tools available to them. Clow further espouses the necessity to properly select metrics and to design appropriate assessments for LA. This reinforces the value of keeping teachers and researchers in-the-loop to provide context. The field has of course progressed since these overviews, which we will examine in the paragraphs to follow.

In [Koh et al., 2016], researchers explicitly address the pedagogical side of LA while compiling a Team and Self Diagnostic Learning framework focused on experiential and collaborative learning. This approach brings together a pedagogical model with teamwork conceptualizations and competencies and LA to study collaboration in small groups. Evaluation of the framework was limited to assessing student and teacher

Knowledge Continuum	Five Steps of Analytics	Web Analytics Objectives	Collective Applications Model	Processes of Learning Analytics
Data	Capture	Define goals	Select	Select
		Measure	Capture	Capture
Information	Report		Aggregate	Aggregate & Report
Knowledge	Predict		Process	Predict
Wisdom	Act	Use	Display	Use
	Refine			Refine
		Share		Share

Figure 2.1: Different LA-related frameworks and models [Elias, 2011]

perceptions at this stage. Both groups responded positively and with that as the impetus the group planned future applications and refinements for the framework. Another recent and novel approach [Olsen et al., 2020] focuses on the use of temporal analysis to study student collaboration. Data was collected from middle school students working in dyads within an intelligent tutoring system. Modalities that could be recorded with a high frequency including gaze and audio results were found to be the most predictive of student learning gains. On top of the time-based analysis in this work and their recommendations for collecting data from different time scales, researchers also combined multiple data streams via multimodal learning analytics.

Building on the LA field, Blikstein [Blikstein, 2013] first presented the idea of multimodal learning analytics (MMLA) in 2013. By collecting and analyzing data from multiple sources, Blikstein hypothesized and later showed that we could obtain a better understanding of complex cognitive behaviors that would be unobservable when only one mode was available for analysis. In effect, MMLA involves capturing the data that is necessary to measure certain constructs that are both important for learning but that cannot be analyzed through programming actions alone. This includes specific areas such as confusion and cognitive load [Mangaroska et al., 2020]. Other steps forward in MMLA include the development of SLAM-KIT [Noroozi et al., 2019] to facilitate data visualization and processing. When multimodal data is captured and combined in this way, researchers, teachers, and students will all have new ways to observe regulation processes, allowing them to find evidence of critical learning moments and act upon the knowledge gained in order to improve the effectiveness and efficiency of student learning. Self-reported data should be combined with objective results such as logs, videos, physiology, and biology in order to provide a comprehensive view of how learning is being regulated.

In short, MMLA can “do the work” of a teacher/instructor in observing students working together and assisting them when it is present as a component of an intelligent tutoring system or other similar implementation [Spikol et al., 2018]. Other applications of this theory include the work of Grover et al. [Grover et al., 2016b] as they present the first stages of their MMLA framework as applied towards studying col-

laborative problem solving. By combining Microsoft Kinects with laptop mounted webcams, MMLA could provide measures of proximity, engagement, joint attention, communication, turn-taking, activity level, a “dominance” rating, and user actions. Starr, Reilly, and Schneider [Starr et al., 2018] also make use of a Kinect sensor to collect a variety of data and perform MMLA. They aimed to apply these techniques in order to explore the relationship between collaboration quality, learning gains, and task performance. Findings indicated a correlation between code quality and all of the following: collaboration quality, tasks completed, task understanding, and improvement over time. The task in question involved maze navigation with a robot, leading us into the next set of learning analytics papers related to their application in our primary areas of interest.

In the area of educational robotics and learning analytics, one approach [Nasir et al., 2019] has been to combine data collected from the robots themselves with traditional assessment results. Students participated in a constructivist learning activity to build their CT skills while researchers analyzed the gathered robot sensor and interaction log data. Though only at an experimental phase at the time, early results showed promise for distinguishing between high and low performing students using such robotics data for classification. Amo et al. [Amo et al., 2021] surveyed the literature on LA as related to specifically robotics sensors. Their search results did not return the previous paper despite it being within the appropriate time period and seemingly matching their search criteria, leaving something to be desired. The only two returned approaches involved k-means clustering in one case and the collection of environmental data in the other. Further exploring the role of a robot’s movement and sensors in student learning thus remains a priority. LA and educational data mining are blended in a framework developed to perform constructionist research in areas such as robotics and CT [Berland et al., 2014]. Potential benefits included improved precision of formative assessments, further support for qualitative research, and the generation of actionable data for learners and teachers alike. However, specifics of the choices of analysis techniques and quantitative results to back up the claims are not provided here.

Moving further into CT and programming examples, Event Navigator software [Blikstein, 2011] was created to allow researchers to view and analyze student programming action logs over time. The framework was tested on nine undergraduate students in a programming course, and just this tool with processed logs was enough to recognize seven coding strategies and three coding profiles (copy and pasters, self-sufficients, and a mix of the two) amongst the learners. The primary takeaway was that LA could be used to identify which behaviors students are exhibiting before providing assistance to address areas of weakness or to reinforce areas of strength. Though this software does not appear to be readily available at this point in time, features of NetsBlox could allow for similar approaches to be replicated.

Building upon theory provided in other examples, Berland et al. [Berland et al., 2013] put those ideas into

practice. They applied LA along with educational data mining to understanding how students develop their programming skills and advance between the behaviors of exploration, tinkering, and refinement. They particularly use these tools to highlight the value that tinkering can provide for novice programmers learning how to code, including in such areas as planning skills and quicker progression to more advanced concepts. Results from a study conducted with a virtual soccer-playing robot included the analysis of transitions between different types of program edits done within the context of clusters that defined sequences of programming actions based on their ratios of action primitives. Tinkering was verified as a valuable component of this full programming learning progression of students, but results were impacted by immediate feedback provided by the environment, and the development into refinement behaviors is still a necessary component.

Applications or theory of LA in work related to cybersecurity education have been minimal. An informative framework for specifically cybersecurity exercises was presented in [Maennel, 2020]. The exercises that were reviewed are more for undergraduate or professional learners and therefore more complicated than anything that would be suitable for our learners. Still, the framework can assist us in the framing of our analysis and raise awareness of various analysis and curricular considerations still to be made. LA is an integral component to the learning environments created for the purpose of teaching cybersecurity at different ages as presented in [Vykopal et al., 2021]. In particular, it was applied during classes to aid teachers in monitoring the learning progression of students, after classes to allow learners to reflect on their own learning, and after classes to lead to improvements in the particular course. Interaction events are logged in much the same way as programming logs from NetsBlox. As with many examples, the specific LA techniques applied to these events lack full descriptions. The literature review turned up no related findings in networking, though this last example does have some overlap with networking in the form of topologies.

Despite this focus on LA, educational data mining or other techniques occurring after an intervention can still provide valuable information for refining curricular materials and more accurately identifying student misconceptions. The previously discussed robotics and CT example of Berland et al. [Berland et al., 2014] bridges the gap between the two fields by framing the path forward for better understanding student interactions within learning environments. This includes automated feedback generation based on educational data mining approaches adapted from previously created methods intended for large solution spaces. Additionally, Kinnebrew et al. [Kinnebrew et al., 2013] applied differential sequence mining on results collected from Betty's Brain [Biswas et al., 2005; Leelawong and Biswas, 2008]. They were attempting to address a problem in previous results where patterns of actions representing key learning strategies were applied by a majority of students but not by enough students for common threshold values to recognize the pattern. The primary solution was a combination of techniques able to identify differentially frequent patterns between groups of students.

More importantly and following from one of the key points from [Elias, 2011], context was provided in the form of student performance in order to better investigate and ultimately assist with the cognitive and metacognitive learning behaviors and strategies of students. Another relevant concept for our work is the processing and abstracting of logs to remove extraneous information and ensure relevant context is retained. A follow up to this [Kinnebrew et al., 2017] performed LA by starting from learner activity logs from Betty's Brain. Sequence mining and model-driven analysis then led to the observation of discrete patterns of behaviors. The patterns could then be linked to desirable or undesirable tasks and processes, and after further sequential analysis results could be applied to identifying gaps in the model. This information would then be fed back into the system and used to provide data-driven recommendations to learners. The dedication of these approaches to improving the experience of learners indicates the potential of adapting these techniques as part of a LA framework as well.

2.1.5.1 Analysis of Progress of Student Learning

More specifically for this work, we were interested in previous techniques and theories of quantifying how students learn over time via their progression through a curriculum. MMLA has applications in this regard as well, as properly assessing complex fields such as CS requires focusing on the learning process undergone by students rather than only the final results of programming projects or tests. Such an analysis can reveal distinctions between students, with one example provided in this work separating the 'tinkering' vs. 'planning' tendencies of students via clustering and programming patterns [Blikstein, 2013]. Another approach was to set debugging behaviors as the primary dependent variable for analysis while undergoing predictive modelling via multiple data streams [Mangaroska et al., 2020]. [Mao et al., 2021] proposed a neural network model combining abstract syntax trees and a Long-Short Term Memory model. This combination could handle both the linguistic nature of programming and the temporal features of progressions of student learning. These temporal features were further highlighted in [Liu et al., 2018], where learning trajectories across knowledge components were utilized to identify high-leverage moments of potential successes or failures. These moments could then be analyzed in more detail in order to optimize efficiency of the analysis.

As expected from the nature of the content, many attempts to predict future performance were based on programming logs. Researchers in [Watson et al., 2013] implemented an extension to an IDE to capture the code, timestamp, result (success or failure), and any error returns upon a failure. They created and applied an algorithm intended to score student compared to their peers in terms of time taken to resolve specific error types. Results from the algorithm were found to explain 42.49% of variance in performance. Meanwhile, Piech et al. [Piech et al., 2012] produced graphical models of programming progression and compared performance differences of three techniques for measuring distance between expert solutions and student

artefacts. They ultimately took a Support Vector Machine approach combining a weighted sum of Abstract Syntax Tree (AST) change and Application Program Interface (API) dissimilarity. They also applied Hidden Markov Models (HMM) and specifically clustered paths students took through the HMM as another means of measuring student learning. This approach was statistically significant for distinguishing between mean midterm performance based on results from a first homework assignment. Another graphical approach meant to display information for teachers in particular is described in [Biswas and Sulcer, 2010]. Concept maps and variable tracking are both represented visually. This work was further advanced in a number of areas, including Hutchins' creation of a teacher dashboard [Hutchins, 2022].

The analysis we ultimately implemented was largely inspired by the work of [Mahzoon et al., 2018] as they analyzed temporal patterns of student data. Data from each week of a course was combined into nodes representing all assignment performance and other information from the week. By applying Support Vector Machines for classification, the analysis then attempted to predict final performance (passing or failing the course) by calculating the distance to the decision boundary as the result of each node. With the available data, linear regression was deemed to be a better fit for our work, but this idea of predicting results from previous scores was a core component of that. Other inspiration included previous work targeting the role of computational thinking as a conduit for then transferring learning gains to another related subject [Hutchins et al., 2020a] - science in the work of Hutchins et al., and networking and security in ours. Additional related work can be found in the regression approaches of [Jose et al., 2016] and [Kim et al., 2016] and the correlations drawn between student programming progressions of [Cabo, 2019].

2.2 Learning Environments and Educational Robotics

Learning environments are an effective supplement to curricular materials and teaching approaches across a wide variety of settings. This includes CT and robotics which focus heavily in this work. We observe the components that make up a learning environment and why such components tend to be selected in Section 2.2.1. The combination of educational robots with learning environments is presented in Section 2.2.2. Finally, recommendations for appropriately scaffolding learning in the areas of CS and robotics through learning environment usage is shown in Section 2.2.3.

2.2.1 Key Features

Early work in the area of learning environments [Harel and Papert, 1990] studied the effects of constructionism by giving students active roles in their own knowledge acquisition. The students would serve as the teacher or explainer while building models within the learning environment. This was posited and then shown to lead to greater mastery of the programming language (Logo), the domain (fractions), and metacognitive

thinking skills. The authors further highlight the development of activities that allow for students to do, learn, and think along with a self-directed but assisted project-based framework as key considerations. Dillenbourg, Schneider, and Synteta [Dillenbourg et al., 2002] surveyed virtual learning environments (VLEs) with the goal of compiling a comprehensive list of necessary features. These included that VLEs should be:

- Information spaces allowing for educational interactions
- Explicitly represented in some way
- Places for students to be both active participants and co-constructors
- Beneficial for both distance learning and classroom activities
- Made up of combinations of multiple technological and pedagogical approaches

In addition to a zoomed-in look at each of these, the authors also caution that VLEs are not intrinsically positive contributors to learning gains. Instead, they offer affordances that can be beneficial if designed and used properly. Chou and Liu [Chou and Liu, 2005] took a different approach by comparing the effects of VLEs to traditional classroom implementations. In their case, the VLE led to greater learning achievement as well as higher self-reported self-efficacy, satisfaction, and well-being within a group setting.

Another approach [Moons and De Backer, 2013] instead branched into constructivism instead of constructionism as a key consideration of a learning environment. Other specific design principles included were the needs to demonstrate relevant programming concepts that would normally be included in a standard course, to minimize cognitive load on students while following other cognitive science guidelines, and to incorporate findings related to human visual perception when designing visual aspects of the environment. In particular for the relevant programming concepts area, a compilation of a new student survey and previous findings was included to serve as a baseline for future CS learning environments moving forward. Other key features of a Web 3.0-based VLE were discussed in [Cabada et al., 2018]. They first outlined a research process involving designing the software architecture, implementing various components, integrating those components, and finally evaluating the results through interventions. As part of this process, the authors were able to incorporate, validate, and encourage the use of new technologies or newer improvements to old ideas. These included an affect recognizer, a recommender system, sentiment text analysis, and an authoring tool to aid teachers and researchers in creating new exercises.

All of the above findings followed essentially the same principles when it comes to the structure of the learning environment. They are examples of directed learning, where creators of the tool define knowledge and skills that students should acquire and retain. Alternative approaches could instead provide an open-ended learning environment (OELE) for students. OELEs encourage students to determine what and how

they should learn, if learning goals have been attained, and what the next steps should be. Key assumptions for these environments were captured in the early days of the area [Hannafin et al., 1994], including:

- Understanding can only be achieved through proper context and experience
- Individuals determine their own ideas of what should be understood
- The emphasis is on students developing cognitive skills as opposed to producing learning artefacts
- Understanding has a higher value than knowledge
- Learning processes that differ qualitatively must feature methods that differ qualitatively

To summarize these points, OELEs are intended to increase student understanding through their self-directed exploration of an environment that was explicitly designed for that purpose.

We now turn to other early research related to OELEs. Land and Hannafin [Land and Hannafin, 1996] move beyond simply describing the design process to instead highlight how students learn using OELEs. Their theory-in-action framework has five major components: contexts of the learners and the system, affordances of the system, the relationship between intention and action, feedback from the system, and how learners process the presented environment. Further details are provided in each area, and overall paint the picture of the necessary pieces of an OELE to best encourage student understanding. Land [Land, 2000] would later explore the cognitive side of student learning within OELEs. To address shortcomings of some OELEs despite the strong theoretical ideals, she recommends a trio of design specifications. These included attention-directing towards key variables or other visual information sources, prompting and guiding connections between current happenings in the environment and prior knowledge, and providing metacognitive and teaching-learning strategical scaffolding.

Throughout the remainder of this section, we will highlight some key considerations when it comes to various aspects of learning environments. The role of robotics within learning environments is one example - one previous implementation in particular [Ramachandran et al., 2016] noted that on-demand help features can often lead to students misusing those features in some way without proper guidance. Their solution involved social robots serving as teachable agents and leading students to productive help-seeking behaviors instead. Scaffolding is often necessary within learning environments to ease a student's cognitive load while learning new, complex topics. For problem based learning environments in particular, we can facilitate student learning while leaving agency in the hands of the students, thereby achieving one of the main goals of scaffolding [Ertmer and Glazewski, 2019]. Some specific considerations include developing strong instruction sets to describe complex tasks without confusing students and methods to attract student attention towards important elements of the environment as related to the problem at hand.

2.2.2 Role of Robotics

In a review of the impacts of technology in the classroom [O'Malley and Fraser, 2004], robots in particular are an oft-mentioned topic. They are highlighted as a means for rendering the implicit programming and other skills learned by students in an explicit manner, leading to a stronger reinforcement of the knowledge students had gained. Robots also serve as a visual representation of any programming errors students may have made and can be implemented at varying levels of complexity depending on the age of the students involved. When integrating robotics with a learning environment, tailoring the environment towards the age range of students that will be using it is the ideal situation. This was the thinking behind the creation of NetsBlox [Broll et al., 2017] as a progression from Snap!. NetsBlox is still a block-based environment, but was tailored towards high school students and is more advanced by incorporating networking and cybersecurity concepts that are not present in Snap!. Robots have also moved beyond just serving as a technological tool in the classroom or other learning situation. Instead, educational robots in particular have been recognized as serving an important role in student learning. This learning is not exclusive to conceptual knowledge or practical skill gains, but also reaches into the development of creative thinking and problem-solving skills [Karim et al., 2015]. On top of learning benefits, robots are often used for promoting student interest in STEM subjects or in CS [Khanlari, 2013]. In particular, teachers noted that robots could facilitate learning by playing and raise the self-confidence of students through the hands-on nature of physical implementations. More advanced robotics platforms when combined with learning environments could even address complicated engineering (including cybersecurity and networking) problems in a more easily observed way. This focus on authentic problems further increases student engagement and therefore can lead to additional learning gains and likelihood of remaining in STEM + CS fields [Grubbs, 2013].

An important consideration when combining robots with a learning environment is ensuring that this usage contributes positively to student learning and motivation. Though many previous results have been positive, others have had a negligible or even negative impact [Benitti, 2012]. Therefore, iterating upon a planned curriculum for an intervention to address any shortcomings, student misconceptions, or other concerns that are raised is necessary to achieve maximal effectiveness. We are hopeful that this combination of robotics and learning environments can be fruitful in terms of conveying CS knowledge to students while also encouraging further exploration of the field [Liu et al., 2013]. Even if a sufficient quantity of physical robots cannot be obtained, there are still other options for supporting learning environments in this pursuit of student learning.

Designing an environment capable of running with or without robots allows us or other researchers to designate students not using the robots as a control group. By combining this separation with pre- and post-

examinations and pre- and post-attitudinal surveys, we can gain an insight into the effects robots are having on the students [Nugent et al., 2009]. As a further step for a designated group of students, robots could be implemented via unplugged activities. In this activity, students take on the role of the robot. Possible benefits include teaching important concepts when resources such as computers or physical robots are not fully available, providing a more transparent view of those concepts to students through this physical manifestation, and engaging students via the opportunity to act like a robot [Rand et al., 2018].

Based upon CS and STEM education findings [Amo et al., 2021] related to the effectiveness and heavy usage of project-based learning, we now study key conceptual features of these approaches within a robotics setting. Grover [Grover, 2011] studied the development of CS language skills as students completed various projects with a robotics platform. Though the final project was up to student selection and therefore cannot provide a standardized perspective, it is likely that this decision encouraged creativity amongst participants with the end result of increased learning and engagement. Survey responses were positive and students were better able to communicate using the appropriate academic language of this domain upon completion of the intervention. Meanwhile, for a more standardized project-based setting we turn to results presented in [Karaman et al., 2017]. The projects were designed to require fairly complex programming solutions along with system-level thinking and design skills, but the complexity was mitigated in a positive way through lectures and practice exercises that hinted at the correct implementation.

Another approach [Barak and Zadok, 2009] combined project-based learning with a focus on problem solving techniques. They answered research questions related to the inventive solutions created by students, knowledge obtained while completing projects, and the effects of informal instruction of science, technology, and problem-solving concepts upon students throughout the projects. Project-based learning is an integral component of the SPHERES Zero Robotics competition series [Nag et al., 2013]. Recent work [Karaahmetoglu and Korkmaz, 2019] examined the role of project-based learning within two contexts - a control condition using block-based programming and an experimental condition using Arduino educational robotics along with the block-based programming environment. No significant difference was found between the two in terms of STEM skills or test scores, but the experimental condition was shown to be much more effective for increasing the CT skills of students. A study of robotics within a project-based setting [Damaševicius et al., 2017] was conducted with the goals of contextualizing learning, immersing learners, and increasing engagement due to the presence of robots in a physical setting. Project completion was found to have a significant positive relation to passing the exam of the course and for the acquisition of problem-oriented skills, serving to validate this approach.

Though project-based learning has the advantage of being hands-on and potentially combining multiple technologies, there are also challenges due to the open-ended nature of this learning style. Students still

require some monitoring and scaffolding from teachers or the environment itself to succeed [Spikol et al., 2018]. It is also important to maximize the comfort level of students using the system and allow both students and robots to move around the classroom and interact [Özgür, 2018]. This requires the development of lessons that capitalize on affordances of the platform while co-designing lessons with willing teachers for optimal effectiveness. Survey results of the use of robots intended for K-12 STEM education [Karim et al., 2015] indicated positive trends related to classroom usage. These included robots with simple designs that are relatively easy to assemble and relatively cheap, as well as supporting tools such as a visual drag-and-drop programming environment. An identified weakness was the lack of appropriate pedagogical models to assist teachers and researchers in implementing a robotics curriculum in schools, which is a key focus of our work and should improve the classroom experience.

2.2.3 Scaffolding

When considering various forms of problem-centered instructional approaches including areas of interest to us such as problem-based and project-based learning, issues can arise due to the inherently ill-structured nature of these approaches. In attempting to create a situation true to real-world circumstances, we as researchers and educators may instead confound and frustrate students who do not understand the intended pathway to the solution. Scaffolding can have a role in preventing these problems, generally through instructor scaffolds as they assist struggling students or through computer-based tools designed for such a purpose. There are potential problems with the computer-based approach, such as the possible lack of dynamic adaptation to a student's condition. However, it is clear that each of these forms of scaffolding can provide benefits to students if executed properly. As a start to determining optimal configuration of such scaffolding, Belland et al. [Belland et al., 2017] performed a meta-review on computer-based scaffolding methods in STEM education. Though the primary students in the data set were adults and therefore of less interest for our work, many general insights were still determined and presented. The overall average effect size was strongly positive as a point in favor of using these tools, but the customization logic and context specificity of the scaffolds were not determined to have any significant impact. Another key point was related to the cognitive effects of scaffolding, which were found to relate to how students learned concepts, principles, and applications.

Belland continues this analysis with an entire book dedicated to the definition, roles, and implementation of instructional scaffolding within STEM classrooms and other educational settings [Belland, 2017]. Though it would be infeasible to summarize the book here, many insights are presented as related to primarily computer-based scaffolding within this domain. Considerations should be made for the context of scaffolding, such as various features of the learner population, the specific subject matter, and the instructional means. Findings in this area indicated adults were actually the largest beneficiaries of the use of scaffolding - im-

proving effects at the K-12 level would be an important contribution. Also of importance was identifying learning outcomes of the scaffolded instructional tasks as well as how to assess the impact scaffolding has on students.

Scaffolding was ingrained within problem-based learning development from the beginning, with an expectation of an expert tutor being assigned to each group of students. The specific recommended scaffolding techniques involved progressing students through discussions, ensuring that students remained on the appropriate learning progression towards their objectives, and facilitating the group as a whole [Ertmer and Glazewski, 2019]. At the time this could only be accomplished through the use of a human instructor, but the development of computer-based tools has changed that paradigm. In either case, it is key to maintain this relationship of one expert per group of learners or to specifically make accommodations allowing an expert to manage many groups at once. Another primary consideration when designing scaffolding is to do so for the purpose of bridging the gap between what students can achieve by themselves and what can be achieved with help. In particular, adaptive scaffolding within OELEs can aid students in solving problems and realizing appropriate learning strategies for themselves [Kinnebrew et al., 2017]. It is recommended for scaffolds to trigger for both particular behaviors and the performance of students, though what these particular triggers should be is left as an open question that should be tailored to each application.

A common issue within learning environments is that students struggle when attempting to complete complex tasks without proper scaffolding [Segedy et al., 2013]. This leads to a need for the development of tools that can capitalize upon various data sources to identify common behaviors and misconceptions of students. Then, scaffolding could be created and implemented that is specifically targeted at correcting poor behaviors, reinforcing correct behaviors, and clearing up these misconceptions. Authors of [Touretzky et al., 2013] make the case for an initial scaffolded programming environment to first introduce students to computing concepts before they progress forward to more complicated but fully-featured languages. Though the instructions and other features of the initial environment were still quite complex, the use of scaffolding techniques eased students into their first forays into CS. Recommendations in [Hutchins et al., 2018] in regards to scaffolding involved using an iterative process to identify, refine, and re-evaluate the needs of students. Scaffolding can also apply to the specific tasks students complete. Instructional tasks in particular are cited as introducing students to concepts in a scaffolded manner and having a positive impact on establish a baseline knowledge level with those concepts. Finally, scaffolding should be inclusive for students of different experience levels and other background factors.

Another scaffolding area is related to demonstrations used to first introduce various procedures and concepts to students. This type of scaffolding has proven to be effective, but generally requires much effort on the part of researchers to create appropriate methods. These implementations are also rarely transferable across

domains or even within the same field. Though one option is to just make this a priority of initial development, another means of creating this kind of scaffolding in a more general manner is presented in [O'Rourke et al., 2015]. The presented tool allows for creation of multiple types of scaffolding, which are often required in order to encompass all possibilities from both a student and learning environment perspective. A different type of scaffolding could relate to first introducing students to CS in such a way that they do not become overwhelmed and are therefore more likely to remain engaged. CS Unplugged activities [Taub et al., 2012] were developed to capitalize on this idea, combining constructivist principles with activities not requiring a computer and key CS concepts. Ensuring that these ideas are successfully included in Unplugged activities or any other means of introducing CS to students is of utmost importance. Failing to do so can lead to confusion and disinterest as the activities do not properly convey the information in an accurate or entertaining manner.

Scaffolding for computational thinking (CT) has been studied extensively. An example [Sengupta et al., 2013] specifically includes scaffolds that allow implemented algorithms to be more easily visualized by students. This theoretically prevents students from accumulating many errors that eventually lead to either an arduous debugging process or a total disengagement. The authors also espoused the necessary steps beyond scaffolding, as without progressing through scaffolding and removing abstractions students could conclude their experience of working in a learning environment with an overly simplistic view of the concepts presented. The C2STEM learning environment [Hutchins et al., 2020b] scaffolded physics learning through computational modeling and by modifying an existing block-programming environment with a domain-specific modeling language. Animations would occur over time to realistically portray the actions as programmed by students as another form of scaffolding. The authors also comment on the dichotomy between providing sufficient scaffolding to guide students towards solutions while allowing them the freedom to explore the available problem space.

Another intersection of scaffolding methods worthy of examination involves the use of robotics within CS and CT curricular implementations. For example, in [Angeli and Valanides, 2020] scaffolding was needed to address some deficiencies with the chosen robotics platform. The particular concern was that it would be difficult for students to actually understand CT concepts without a visual representation of the commands within their programs, which was not included as a feature of the robot. The authors also examined the differences between boys and girls within this CT and scaffolding context, finding that boys experienced more benefits from an individual, manipulative-based task while girls gained more from a collaborative writing activity. This was not to detract from either group, as the more important result was that it was possible to scaffold all students towards the desired learning objectives through well-designed activities. Additionally, Wu [Wu, 2016] began the process of identifying effective CT scaffolding for young students enrolled in a robotics programming course. Preliminary findings recognized that different areas of CT required different

scaffolding techniques. Areas like variables and abstraction were best facilitated by teachers intervening, while debugging strategies required a different tactic.

Though robots could be lacking features that would then require scaffolds, robots could in turn be used for scaffolding purposes. The final platform from the previous section [Wu, 2016] takes advantage of the robot being used for programming tasks as a way to visually facilitate the debugging process. Scaffolding could also supplement the standard capabilities of a robot, such as incorporating physical cards representing blocks within a block-programming environment in order to get students thinking about their programs before creation [Angeli and Valanides, 2020]. Such a scaffold could instead be included within the environment itself in some way to better connect actions students take on the computer with the resulting robot actions.

Researchers in [Barth-Cohen et al., 2018] took a different approach to scaffolding, with instructors only intervening as a form of ad hoc scaffolding for particularly struggling students. However, they recognize the importance of gathering data in order to identify better scaffolding opportunities to assist students with debugging programs while working on a robotics platform. A final example [Touretzky et al., 2013] involved using simplistic programming environments as a scaffold to eventually lead students into being able to work with Lego Mindstorms robots. They focused on developing a deep (the ability to recognize concepts in a programming context) and abstract (the ability to ignore syntax and comprehend the intended purpose) understanding within students in order to accomplish this.

2.3 Past Implementations

We now examine past implementations of full or partial learning environments with relevance to this dissertation. Though we did not create a new environment here, the review still reinforced the selected, pre-existing environment. We begin with a sequence of completed environments including the relationships between them 2.3.1. In Section 2.3.2, environments focused on the integration of educational robots to other learning tools are shown. Finally, implementations that include scaffolding and are related to CT, robotics, or other areas of interest are discussed in Section 2.3.3.

2.3.1 Complete Environments

We begin our examination of past implementations by highlighting complete learning environments that attempt to introduce novel methods (or improve upon existing methods) for teaching CS, CT, or other STEM subjects to primarily K-12 students. Betty's Brain is one such example that was developed to take advantage of the learning-by-teaching paradigm, allowing students to teach the on-screen agents and thereby teach themselves [Leelawong and Biswas, 2008]. The system was actually designed to incorporate three separate versions to perform comparison studies. Two options were an agent serving strictly as a tutor and a standard

learning-by-teaching approach that each provided feedback based on the domain. The third was an expanded learning-by-teaching approach that also incorporated self-regulated learning strategies as a form of feedback. The environment allows for three primary methods of learning. These involved teaching the agent (Betty) through modifying a concept map, asking questions of Betty to determine her knowledge and therefore affirm the student's own thoughts, and quizzing Betty to observe her responses. Students were also provided with relevant domain materials to refer back to in-between participation in these methods. As further discussed in [Biswas et al., 2016], the agent's responses were powered by qualitative reasoning mechanisms to determine appropriate answers and questions, providing further explanation to students upon request. Betty's motivational feedback was implemented with visual cues appropriate for the age of students using the system, with the agent expressing happiness or disappointment as necessary. Upon examination of student results - which were often a mix of high and low performers with minimal middle ground - decisions were made for new design implementations. Betty's Brain was transitioned into becoming an open-ended learning environment, providing more tools to students to assist in their learning and as a general means of enabling student experimentation.

Our next analysis of learning environments begins with Scratch, a block-based programming environment first unveiled in the early 2000's [Maloney et al., 2004]. The original purpose of the environment was to attempt to introduce programming to young students from economically-disadvantaged settings through media-rich and network-based activities. Design principles included the immediate demonstration of the value of the environment to the interacting students, the ability of students to produce something worthy of showing off to others, the adaptability towards students with different interests and ages, and an environment that was quick and easy to learn but progressively more interesting and difficult to master. This led to the block-based programming decision as well as programmable manipulation of media sources, shareability of projects and objects, integration with physical tools, and support for multiple languages as other implementation components. Snap! [Harvey et al., 2013] was later introduced as an extension to Scratch. On top of graphical improvements expected from this roughly ten year gap, Snap! was also intended for older students through the removal of some abstractions taken in base Scratch. Named procedures and data procedures were included to accommodate recursion and higher order functions respectively. Other improvements included structured lists and the inclusion of sprites as first class objects able to inherit or pass on properties.

On top of Snap!, the NetsBlox visual programming environment was produced [Broll et al., 2017]. Also intended for a somewhat older audience such as high school students, the key additions for NetsBlox were message passing (and other network features) and Remote Procedure Calls (RPC). With the network features, students were able to collaborate on projects in real-time, while either co-located or working remotely. This took the project-sharing goals of Scratch in a new, more advanced direction. Additionally, RPCs could

provide access to a wide range of services in order to expand the capabilities of the environment while not requiring students to completely construct such calls from scratch. As an example, Google Maps could be incorporated into the visual window of the interface and students could then program a variety of actions to interact with the displayed map. The final step in the process is the Collaborative, Computational STEM (C2STEM) learning environment [Hutchins et al., 2020b]. Design-based research combined with these existing block-based programming environments led to the development of C2STEM as an open-ended learning environment featuring a learning-by-modeling framework in order to convey STEM and CT concepts to students. Key advances beyond the previous implementations involved this pedagogical focus on taking advantage of all aspects of the system to encourage student learning while maintaining the low-barrier to entry as specified by the original Scratch directive. Additional design principles included evidence-centered design, a domain-specific modeling language, students being allowed to explore dynamic processes as a form of learning, and the incorporation of preparation for future learning.

Alice [Cooper et al., 2000] is another example of a complete learning environment, this time entirely targeting programming concepts. The goal was to ease problems students were encountering in terms of algorithm development, the application of problem-solving techniques, and the use of programming constructs. Alice allows students to create 3-D environments and provides tools for scripting and prototyping the behaviors of 3-D objects. The implementation is fairly complex as a first introduction to programming considering the use of the Python language and the varieties of knowledge necessary to use the system including concepts of graphics and animation, state, and events and responses. Further work has enhanced the original outdated features in order to provide an easier to use graphical user interface [Al-Jarrah and Pontelli, 2014]. Additionally, these updates came with an emphasis on the collaboration of students, much like the NetsBlox update to Snap!/Scratch. This extension to Alice is known as AliCe-ViLlAgE. It facilitates concurrent programming amongst multiple students via several options. These included dynamic role changing for pair programming or other desired configurations, imposed roles by teachers, or free-flowing role switching by the students. Additionally, all group members can see the code and collaborate even if not co-located through an entirely synchronous process. In general, the programming process has now been abstracted similarly to a block-based environment.

Game-based learning environments have shown a lot of potential for keeping student engagement high as the main feature, thereby theoretically allowing students to better learn the materials as they remain focused on the tasks at hand. One complete example is the ENGAGE project [Rodríguez et al., 2013]. On top of this engagement idea, it also seeks to contribute to research related to the actual effectiveness of this type of environment while broadening participation in CS to underrepresented students. The programming interface used is much like those of Scratch and Snap!, but the blocks are now modifying certain features

of the game environment. Basic programming concepts, networking, binary, cybersecurity, and big data are all incorporated into the main curricular implementation. Min, Mott, and Lester present an extension to the ENGAGE game-based learning environment aimed at teaching CS to K-12 students [Min et al., 2014]. The curriculum was developed based on AP CSP [K-12, 2020] courses, and the previously-described environment is accompanied by adaptive scaffolding tools to ease the process of teaching CS principles to students as well as providing access to other pedagogical techniques.

We conclude with a few singular environment examples. TOY is presented as an additional virtual learning environment [Arhippainen et al., 2011]. Its design principles included support for inquiry-based learning, cross-disciplinary cooperation, and learning-by-doing for the students with teachers in a supporting role. It also features multiple workspaces depending on the subject domain being studied, making for a modular and versatile tool where students can work on individual or cooperative projects. Finally, a web-based learning environment specifically for Java programming was developed and presented in [Cabada et al., 2018]. This environment's primary calling cards are the incorporation of a variety of learning technologies, such as a recommender tool, an affect recognizer, and the ability for sentiment analysis. Each of these interact in real-time to create the foundation of the environment and provide personalized feedback to students. It also includes authoring tools, which allow teachers or researchers to create new situations for students without requiring an extensive programming background themselves.

2.3.2 Role of Robotics

We now turn to previous implementations of robotics within learning environments or robotics solutions that could be effectively integrated within learning environments. Cellulo [Özgür et al., 2017; Özgür, 2018] was presented as a robotics platform for use as a versatile and generic tool while being blended into the classroom and capable of meeting the restraints of said classrooms. To accomplish this, the group drew from a few distinctly separate sources, including previous handheld haptic-enabled robots, tablets, and printed activity sheets. One interesting facet of their desire for ubiquitous, versatile, and practical robots was removing the social element that can sometimes evolve between students and robots. They also aimed for the robots to be used across disciplines and represent a simple, robust experience for students and teachers with low overhead. The robots move about on simple printed sheets of paper that caused specific behaviors as the robots moved to specific areas. The paper itself had built-in microdot pattern to allow for self-localization of the robots. Other features included holonomic motion from a permanent-magnet assisted omnidirectional ball drive, 6 capacitive touch buttons, and wireless Bluetooth communication. All of these features combined to allow for analysis along three dimensions: passive vs. autonomous robots, independent vs. swarm robots, and individual vs. collaborative groups from the student perspective.

In a robotic learning environment geared towards middle school students, Walker et al. [Walker et al., 2016] present the robo-Tangible Activities for Geometry (rTAG) system. The goal was to observe the impacts of physical robots in terms of student perceptions of teachable agents, motivation, and learning. Though hypotheses expected the physical robot within the embodied environment to improve learning and social perceptions of the robot, the actual results were fairly mixed. Finally, additional details about this system are presented in [Giroto et al., 2016]. rTAG builds upon the combination of robotic learning and teachable agent environments in a fairly unique way, taking advantage of components that are relatively inexpensive or already present in a classroom setting. These components are a Cartesian plan projected onto a floor mat, the agent itself which is an amalgamation of a Lego Mindstorms NXT robot and an iPod Touch, and a mobile interface that is also an iPod touch. Emphasis is placed upon the role of the student in interacting with the robot. The students move around and provide instructions to assist in navigation while generating feelings of embodiment in themselves.

Open-ended projects related to robotics and CT concepts were the final step in the process presented in [Grover, 2011]. The researchers made use of a previously developed tool called the GoGo Board [Sipitakiat et al., 2002] to introduce middle and high school students to this field and to serve as a microcontroller for the final projects. Another approach [Karaahmetoglu and Korkmaz, 2019] likewise made use of a microcontroller within a project-based robotic application, in this case the Arduino platform. Their curriculum was integrated within a standard Information, Technology, and Software course for 6th grade students. They modified several weeks to form an experimental group using the robotic platform while a control group continued on with the standard block-programming environment. On the other hand, alternative approaches have involved pre-built, advanced robots with many features packed in [Karaman et al., 2017]. The intervention culminated in a race which combined previously learned topics with the need for full automation within a collaborative project. Another example [Nag et al., 2013] made use of both a fairly unique robotic platform - miniature satellites called SPHERES - along with a collaborative framework to encourage student interaction during projects.

The Lego Mindstorms series of robots has long been popular for introducing students to both hardware and software aspects of robotics while abstracting away the more advanced concepts. Junior high students underwent a project-based curriculum using this platform [Barak and Zadok, 2009], starting from relatively simplistic but still engineering-focused projects such as building the strongest possible fishing rod or bridge with Legos before moving into the robotics side with a computer-controlled car. A more advanced project required building a robotic arm to quickly throw a ball into a basket. A final example observes the use of educational robotics alongside a project-based teaching approach at the university level [Damaševicius et al., 2017]. The course targeted basic principles of robot programming and control, and used group projects along the lines of developing algorithms intended to solve classic robotic tasks such as wall following or obstacle

avoidance. The researchers also compared results of projects and theoretical exams from the course. These results indicated the potential value of project-based learning for gaining conceptual robotics knowledge.

2.3.3 Scaffolding

As previously mentioned, the Betty's Brain system [Segedy et al., 2013] scaffolds student learning through conversational agents that provide feedback to students. Earlier versions of the system [Segedy et al., 2011] were improved upon to better support such scaffolding and prevent the misinterpretation or ignorance of presented material as much as possible. The different agents provide different types of feedback, such as inquisitive responses from Betty and task or domain information from Mr. Davis. Within the same environment, further work [Kinnebrew et al., 2017] began the process of integrating adaptive scaffolding capabilities by first producing and including analysis techniques to process the behaviors of students working with the system. By combining model- and data-driven techniques, the authors were able to characterize the processes of student learning. Though the scaffolding was not fully implemented at this time, such analysis would allow for future work involving the triggering of feedback based on student behavior and performance.

Scaffolding in [Touretzky et al., 2013] was primarily accomplished throughout a progression between increasingly complex programming languages. Careful structuring was used to ensure students would not become overwhelmed by the complexity of this progression. Additional scaffolding was present within the initial Kodu programming environment in the form of a context-sensitive menu for selecting blocks or naming variables in a more relatable way such as "red score". The authors introduced further scaffolding within the environment to aid in the actual transition, such as showing the programs in a textual form instead of solely icon-based and the use of keywords to guide menu use. Meanwhile, scaffolds within the C2STEM system [Hutchins et al., 2020b] were implemented in order to support interpretation and understanding of modeled phenomena within the context of physics and CT. Animated and step-by-step execution were combined with graphing tools of created and pre-existing variables in order to demonstrate the effects of various model-building actions. The authors also comment on future work involving tailoring scaffolding to achieve a better balance between assistance and exploration-related freedom.

An intelligent game-based learning environment was presented in [Min et al., 2014] that incorporates a few forms of adaptive scaffolding in order to teach CS principles to students. Within the game, scaffolding is accomplished through basic feedback such as on-screen hints. The first of two adaptive scaffolding method applied a dynamic Bayesian network-based approach in order to predict the needs of students in regards to filling knowledge gaps or clarifying misconceptions. Then, appropriate tasks were assigned to meet these needs. The second technique involved improving the pre-existing hints with adaptive tools powered by a data-driven solution space approach. O'Rourke et al. [O'Rourke et al., 2015] did not focus on one particular

learning environment like many of the other approaches, instead opting for a tool capable of being applied across domains. The presented framework was intended to provide authoring tools that others could use to design various interactive scaffolds, with tutorial progressions (providing less explanation as students master concepts), just-in-time feedback (identifying when feedback is necessary and supplying it immediately), and fading worked-out examples (supplying a full procedural demonstration to students at first before allowing them to solve problems completely on their own by the conclusion) all demonstrated by the authors using the tools. A final unique take on scaffolding involved creation of ChalkTalk [Perlin et al., 2018] as a method for rendering concepts in a visual manner and allowing user experimentation with CS ideas. Through initial example implementations of binary search trees and stacks, these concepts could be more easily introduced to students thanks to the ChalkTalk interface. In fact, its goals are similar to a block-based programming language such as Scratch [Maloney et al., 2004] in scaffolding the introduction of new programmers towards higher level conceptual understanding without being dragged down by an overly syntactical approach.

2.4 Critical Summary

In our review of the literature, we were able to determine appropriate approaches taken in past CS education interventions at the K-12 level throughout Section 2.1. This included well-researched areas such as the impact of educational robots (Section 2.1.4.1) and the creation of appropriate assessment packages to uncover student learning of CS (Section 2.1.3). However, neither of these areas have been specifically applied to student learning of cybersecurity and networking concepts of the specific age range of students that we are interested in. Networking in particular has been largely neglected throughout K-12 CS education (Section 2.1.4.3), with the primary examples of education in this area occurring with university students. As identified in various curricular standards, networking is a key component of CS education and should be targeted more heavily at the K-12 level. Cybersecurity has at least been a focus of some work at this level (Section 2.1.4.2), providing a baseline for us to work from. Other areas of the literature review included the theory of learning environments in Section 2.2 and previous implementations of such environments in Section 2.3. Many areas in these Sections have been thoroughly explored, but there is a relative lack of understanding in terms of the robot's role in facilitating the classroom experience as well as within the context of scaffolding complex concepts.

As previously mentioned in Section 1.1, recent updates to various K-12 STEM standards at the national [Seehorn and Clayborn, 2017; K-12, 2016] and state (for example, Tennessee's [of Education, 2018]) levels have focused on cybersecurity and networking like never before. AP CSP [K-12, 2020] is in a similar situation, spurred by demands of the 21st century workforce in these and related fields [Crumpler and Lewis, 2019; Geyamallika and Reddy, 2019; Hoffman et al., 2011]. We were able to identify many specific standards

tailored to the situation at hand. Further analysis would be needed for different states or age groups, but the process is certainly feasible for these separate conditions. The main point to be made is that CS in general along with cybersecurity and networking in particular have a large role to play in the education of students now and the careers they pursue in the future. By making these topics approachable and beginning from a younger age than is traditional, we can better prepare students that are interested in these or related fields, or at the very least raise awareness of the dangers posed by cyber attacks and allow students to reasonably protect their own private information.

There have been efforts to incorporate the prior knowledge of students into overall intervention design. Related past findings were presented throughout Section 2.1, including consideration for the complexity of the platform and curriculum based on prior knowledge [Knobelsdorf and Vahrenhold, 2013] and using prior knowledge as an evaluation metric for students in an intervention [Grover and Pea, 2013]. A more in-depth study of a CS MOOC developed for middle school students [Grover et al., 2015] discovered a correlation between prior computing experience (along with mathematics) and the successful learning outcomes demonstrated by students. Collaboration amongst students is also impacted by prior knowledge of those students. Sometimes students with more previous experience successfully assist their less experienced counterpart in learning the material while boosting their own self-confidence [Denner et al., 2014]. On the other hand, sometimes the students with higher knowledge can struggle to incorporate their partners into collaborative programming. This can lead to situations where the less knowledgeable student or students feel that they are unable to contribute to the programming project, which then prevents all students from learning the desired concepts [Braught et al., 2010].

A recent survey of cybersecurity education papers [Švábenský et al., 2020] found only a few studies incorporating more than basic surveys or exams as their assessment techniques, while the literature review turned up nothing of note pertaining to comprehensive assessments related to networking. Networking was also identified as an unexplored area in terms of learning analytics research through the review in Section 2.1.5. There is certainly a breadth of examples for more generic CS or programming assessments, which were evaluated and presented in Section 2.1.3 primarily. We can learn much from the work of Grover et al. [Grover, 2017] when it comes to developing a suite of assessments that inform us as researchers about weaknesses in the curriculum while also allowing for corrective behaviors on the student side. It is also key to consider how different phases of the curriculum could have different assessment purposes, with assessments for learning, assessments as learning, and assessments of learning serving as three well-defined options [Earl et al., 2006]. It is too easy to ignore the first two categories in favor of assessments of learning, but this neglect can lead to a simplistic understanding of the effectiveness of the curriculum and the struggles that students encounter while completing it. These concepts are true for students of all ages, but the importance

of designing assessments for the proper purpose is heightened for middle school students in particular. Too many assessments or too complicated assessments can lead to students feeling burnt out and overwhelmed, skewing results even if they were able to gain the desired knowledge throughout the intervention. Similar but opposite problems could arise from assessments that are too sparse or too simplistic.

Care must be taken to reach the middle ground between these approaches by drawing upon previous examples and conferring with experts in both the conceptual areas in question as well as those with recent in-classroom experiences with these students - the teacher or teachers. This facet of assessment development - and curriculum creation in general - spurred the need for a co-design process (Section 2.1.4.4). The full assessment suite will also have to encompass alternative options, filling in the gaps left between traditional tests and surveys with less intrusive data collection via programming action logs and project files. Collecting this information is an affordance of the environment, NetsBlox, which will be discussed in more detail in Section 4.2.1. There is still a need to systematically process the data collected into a form useable by educators and researchers via techniques such as Markov chains [Piech et al., 2012] and ECD rubrics [Mislevy and Riconscente, 2005].

CHAPTER 3

Research Plan

This chapter discusses our overall research plan, an overview of the curriculum, and the research questions that drive our analyses of the student data collected in our study. First, Section 3.1 focuses on the primary motivations for this research work. One is the importance of CT, networking, and security and the difficulties computer science instructors face in teaching these concepts to middle school students. Our research also hypothesizes that prior knowledge of CT may impact students as they progress through a curriculum. This section further discusses the difficulties in accurately assessing students' progress in learning the primary concepts and their applications to problem-solving tasks. This is particularly important for learning networking and security concepts at a middle school level. A significant component of our research was to study how hands-on experiences with programming and problem-solving with physical robots influenced students' motivation and engagement with the material.

The next section, i.e., Section 3.2 outlines the translation of our research approach into a conceptual framing and curriculum. This section outlines our primary research area and in particular how the research and development work was advanced. As part of this research, we also studied the effect of prior knowledge in CT as well as their ability to learn and use programming constructs on students' abilities to learn the required concepts and practices in networking and security.

Figure 3.1 presents an overview of this research.

3.1 Overview of Research Problem

Our research problems evolved from the critical summary (Section 2.4) of the literature review that we presented in Chapter 2 and a set of problems that we can make research contributions in, as we indicated in our critical summary. It is from this set of problems that we derived three key research problems that circumscribes the research we have conducted for this thesis.

- Problem 1: The difficulty in teaching CT, cybersecurity, and networking concepts in an effective manner in middle school computing classrooms.
- Problem 2: The potential impact of prior CT knowledge and early learning of CT upon student learning of more complex security and networking concepts.
- Problem 3: The need to generate comprehensive assessment and analysis techniques in this networking and cybersecurity context to assess learning for K-12 students.

The first stems from the importance of cybersecurity and networking as established through a set of standards related to these areas. However, we identified a weakness in previous work on networking instruction. In particular, the only approaches available targeted significantly older students and, therefore, were not directly applicable to our middle school audience. This problem is addressed through the development of the curriculum we will discuss in Section 4.1. The last two modules are especially relevant for this area.

The next challenge was related to the prior knowledge of students, which had not been adequately explored in the past in this context. Students received instruction on the basics of CT knowledge early on in the curriculum (see Section 4.1). In addition, we collected relevant data on student learning and problem-solving using our programming environment and robotics platform (Section 4.2.2) to evaluate students while emphasizing these facets during the studies (Section 4.5).

CT, cybersecurity, and networking have been identified as key components of K-12 CS education. Though past literature provides evidence about CT, networking, and cybersecurity learning individually, there has been little work in building curriculum sequences that include these topics, and student learning in these areas has not been thoroughly studied. In this research, we have carefully designed the learning environment (Section 4.2.1) and introduced CT concepts and practices early in the curriculum, so that they provide scaffolding for learning the more complex networking and cybersecurity constructs that were taught later in the curriculum.

A comprehensive assessment approach was also implemented, and this led to the evaluation work presented in Sections 4.4 and 4.3. These sections present efforts to address the collection and analysis of data and the set of developed assessments, respectively. Later chapters then present the results of the intervention and analysis.

3.2 Research Overview

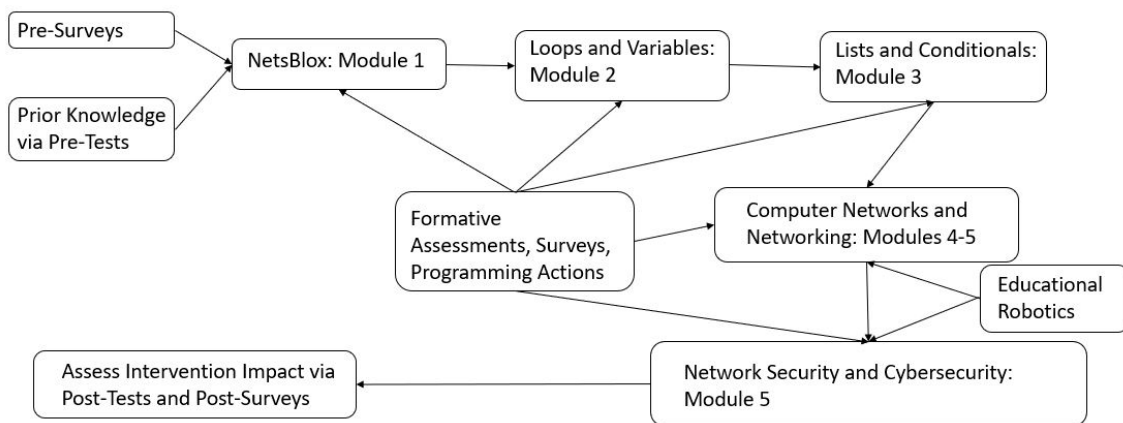


Figure 3.1: Overview of Research

First and foremost, K-12 CS education was the primary focus of the research presented in this thesis. In particular, the aim was to establish a baseline of CT knowledge for the students, many of whom did not have substantial previous experience in this area. Student knowledge levels and computing and educational attitudes were first assessed via pre-surveys and pre-tests that we developed. Though the teaching and learning of CT and programming are distinct, as was identified through the literature in Section 2.1, they are also quite intertwined in terms of the core concepts and practices that they emphasize. We used programming in a block-based programming environment as the conduit for teaching and learning of CT concepts and related practices [Grover and Pea, 2018; Buitrago Flórez et al., 2017]. Within CT, the most important topics as identified through previous work ([Yett et al., 2020a]) and related standards and frameworks ([K-12, 2020], [Seehorn and Clayborn, 2017]) include loops, variables, functions via custom blocks, arrays via lists, and conditionals. Students progressed through these topics in Modules 1, 2, and 3. In addition, we also furthered our understanding of how the CT knowledge of students impacts their ability to extend such knowledge into related areas, such as networking and cybersecurity.

These related areas include key features of computer networks, such as communication protocols and network topologies. In addition, we introduced related topics, such as network security through secure communication and the dangers of unprotected networks. Human factors linked to network security were also considered via an emphasis on strong password generation.

These have all been identified as highly important concepts to begin introducing to students as early as middle school [of Education, 2018]. However, they may still be too complex for students without introducing scaffolding techniques that help represent the concepts in a more practical (e.g., hands-on) and easy-to-understand manner. To better characterize a learning progression for students, we collected and organized assessment, survey, and programming action log data from the interventions. We also included post-surveys and post-tests to have a set of metrics that allowed us to study the effectiveness of the intervention on student learning and behaviors.

Though there is extensive research in general CS and CT education as well as at least some work in cybersecurity education, networking education, particularly at the K-12 level is almost entirely non-existent (Section 2.1.4.3). This is in contrast to sources such as the Tennessee State Standards [of Education, 2018], which specifically highlight the need to include networking protocols and related concepts in the curriculum so students can have a deeper understanding of how computers communicate.

Meanwhile, cybersecurity has more of a supporting role and is applied specifically to methods for protecting data transfer and communication over networks. The relationships between student learning of concepts across the three related domains have also been left unexplored in the past. This has inspired much of this work, with an emphasis on discovering how prior knowledge and CT learning influence the learning of more

advanced cybersecurity and networking concepts.

Additionally, though general areas such as scaffolding have been largely explored, the role of an educational robot in these contexts (Sections 2.2.2 and 2.2.3) have not been studied in a systematic way. In previous work (for example, [Lédeczi et al., 2019] and [Yett et al., 2020a]), we observed the impact of the robot and the platform (NetsBlox) together in enabling student learning of the more complex security and networking concepts. NetsBlox is the block-based programming environment [Broll et al., 2017] and RoboScape is an extension within this environment that allows students to develop and study robot behaviors using block-structured programming. In addition, we have used robots as a primary hands-on approach for scaffolding student learning [Yett et al., 2020a]. The near-immediate communication from student computers accessing the NetsBlox website which enables sending commands through the NetsBlox servers to the physical robots via wireless communication provides rapid feedback for created programs.

For this research study, I, as the primary researcher also participated in co-design activities with the classroom teacher to develop the required curriculum units and assessments that would help us track students learning and progress over the curriculum. This co-design process further involved refining a set of modules beginning with the CT and programming fundamentals and culminating in hands-on experiences in cybersecurity and networking with the educational robots used in that school. Each module was supported with formative assessments relevant to the topics at hand. Results were also partially determined based upon the pre- and post-tests and programming actions and projects.

The overall process is illustrated in Figure 3.1, with formative assessments, surveys, and programming tasks that were developed for each module of the curriculum. In designing this curriculum, we focused on ensuring the integration of concepts and practices throughout the curriculum. This also applies to the use of educational robotics along with the networking and cybersecurity lessons. The end goal of all of this was to evaluate the developed curricula including these sub-components and propose further improvements based on analytical and empirical results.

3.3 Research Questions

Overall, our goal was to develop new or adapt previously created components in a variety of areas. These included an innovative and comprehensive curriculum, formative and summative assessments, the robotics environment to support hands-on problem-based learning, the learning environment, and data collection and analysis. We conducted experiments with and evaluated this semester-long curriculum with end-to-end studies run in 7th- and 8th-grade computing classes. To systematically study this process, we addressed the following research questions:

- Research Question 1 - How did student performance, engagement, and attitudes measured using pre-post-assessments and surveys change as a result of our intervention?
- Research Question 2 - What was the learning progression of students in CT, networking, and security-related assessments, as well as key constructs that bridged these broader categories?
- Research Question 3 - What was the strategy progression of students based on their programming actions during projects and their open-ended questionnaire responses?

To answer RQ1, we computed descriptive statistics detailing changes in student performance and present our results as a set of tables (Section 5.1). Our calculated metrics assessed the significance and effect size of these changes from pre- to post-test and from pre- to post-survey. This included overall categories of interest, but also breakdowns into specific categories of interest and down to the individual question level for selected constructs.

For RQ2 (Section 5.2, two primary analysis methods are used to address the progression of student learning. The first is Spearman correlation analyses. These were intended to compute some of the changing relationships between different evaluation metrics at different points in the curriculum. We also applied linear regression methods to generate plots and related statistics [Jose et al., 2016; Kim et al., 2016]. These models are created using pre-test, formative assessment, and class final data points for the student population in our study. They are then applied toward predicting post-test performance.

Our final research question (RQ3; Section 5.3) required a few different analyses as well. First was a Markov chain sequence analysis using students' activity data obtained from their programming action logs. Specifically, these actions were taken from student work on a sequence of open-ended programming projects. Though visually appealing, it can be difficult to interpret these models. We, therefore, apply Differential Sequence Mining to compare differences in strategies applied by students from project to project. This extends our previous work [Yett et al., 2020b] by restricting to specific subsets of programming actions and by adding the temporal component with the project-to-project comparison. We conclude with a coding scheme applied to mid-module surveys meant to better understand different forms of reasoning students apply to explaining and interpreting pre-existing or developed programs.

3.4 Summary of Chapter

This chapter of the dissertation provided a high-level overview of the research to be conducted. The research problems and research overview (Sections 3.1 and 3.2 connect to Chapter 4 - Methods by informing the development needed. Namely, we created or adapted curricular materials, assessments, and the overall learning platform. Meanwhile, the research questions of Section 3.3 are further discussed in Chapter 5 -

Results. In that chapter, our main findings related to each question including the analysis techniques applied are presented.

For RQ1, we expect to discover improvement from pre- to post-test and survey in the majority of areas. One exception would be certain aspects of the self-assessment that were intended as “control” categories and therefore should not show significant change. As part of RQ2, it was hypothesized that we could track the progression of student learning and use that information to accurately predict post-test performance. We also expect high levels of correlation between early, CT-focused Module performance and later, advanced Modules with networking and security concepts. Finally, RQ3 should reveal an improvement in the programming strategies of students throughout the core sequence of projects. There may also be a similar progression from self-assessment to self-assessment, with students moving towards increased application of advanced forms of reasoning in their answers.

CHAPTER 4

Methods

This chapter outlines the curriculum, supporting tools, assessments, data collection and processing, and analysis methods for answering the research questions presented in the last chapter. This includes details of the co-design process for curriculum creation, specific Modules that were created, and their relationship to specified standards ([of Education, 2018]). Following a description of the curriculum modules, we discuss the block-based programming environment, data collected and processed targeting the RQs, and the analysis approaches. This frames our overall research, which is centered on two key points. First, the implementation of a novel semester-long study covering networking and security concepts in middle school computer science classrooms. Second, the analysis of data collected throughout the study.

Development was based upon the contributions as previously summarized in Section 1.4. A curriculum was needed to facilitate all other requirements and at the very least provide a baseline for teachers that they could tailor to their students' needs. Educational robots paired with a block-based learning environment served as the primary conduit for conveying curricular materials to students. The environment was particularly important for topic introductions and teaching the primary programming concepts. Meanwhile, the robots served to reinforce the application of these concepts using a hands-on approach. Assessments were primarily intended to inform researchers about student learning and identify gaps in the current state of the curriculum and other tools. Some were used for grading purposes by the teacher as well. These were all combined with additional data collection in order to conduct more comprehensive analyses of student work. For reference, all materials used during the primary intervention of interest are available here.

4.1 Curriculum Design

The current version of the curriculum was co-designed (Section 2.1.4.4) with the classroom teacher. This process involved regular meetings and planning sessions with the teacher to design the entire curriculum. This involved the following tasks:

- adapting lesson plans, instructional materials, and NetsBlox projects from prior implementations ([Lédeczi et al., 2019],[Yett et al., 2020a]) to fit with the current classroom objectives, learning goals, and needs of the participating teacher,
- creating and refining lesson plans for content previously identified as difficult, or where students did not demonstrate significant learning gains in prior implementations (e.g., material for Conditionals and

Lists, as discussed below), and

- designing and aligning formative assessments to be integrated within the curriculum targeting the identified learning goals.

In the Fall of 2021, we tested our pilot curriculum in the teacher’s classroom with one 7th-grade and one 8th-grade class to identify potential issues and areas of improvement for the longer semester-long implementation. As part of this process, a total of five modules were developed, under the general guiding principles shown in Figure 3.1. These modules included:

- Module 1 - Introduction to NetsBlox and CT
- Module 2 - Loops, Variables, and Custom Blocks
- Module 3 - Conditionals and Lists
- Module 4 - Messages, Networking Introduction, and RPC’s
- Module 5 - Robot Driving, Networks, and Network Security

The first four modules were intended as introductions to NetsBlox and a number of CT concepts to provide students with the necessary foundational CT knowledge needed to succeed in the more advanced Networking and Security tasks. It was estimated that these modules would take about two weeks of class time per module. In reality, they lasted between two and three weeks each. Some delays were caused by school network or NetsBlox issues, while some were a natural result of observing day-to-day student progress and providing additional time to help students get over their difficulties in understanding and applying these concepts. Module 5 featured Robot Driving, Networks, and Network Security and contained enough material for approximately one month of class time with several logical stopping points that depended on the circumstances and time available. With disruptions such as standardized testing, this was expanded to roughly a month and a half of real-time. Some modules were adapted (in collaboration with the classroom teacher) from projects previously used during cybersecurity camps ([Lédeczi et al., 2019],[Yett et al., 2020a]), while Module 3 (Conditionals and Lists) in particular required a complete overhaul. This will be discussed further in Section 4.1.3.

Each developed module was broken down into four main components. These components repeat several times in the final module, but each module adheres to these standards. First is an introduction to the topic, often through opportunities for students to explore a completed version of a programming project or through embodied activities for some of the complex security and networking concepts. Second is a guided programming assignment, with students following along with written instructions in order to build up certain features of the programming project previously used as the demonstration. Third is an assessment either integrating

debugging tasks, Parsons problems [Denny et al., 2008], or short conceptual experiments within the same context as the first two components of the module. These serve to reinforce the earlier concepts while providing important information about the knowledge development of students to us as researchers and to the teacher or teachers involved. Finally, the last component involves extensions to the created project from step two, with some requirements provided to students but with additional opportunities for them to explore the space on their own.

These projects were generally part of the grade for the course provided by the teacher but were also assessed in this research as evidence of each student's capabilities. As a last general note, these modules were initially developed through the co-design process from the perspective of the teacher rather than the student. This teacher-focus then allowed us to tailor the curricular materials to the students as the pair of us gained an understanding of their capabilities.

One specific early result derived from the co-design process and based on both of our previous experiences was that students needed significant scaffolding to complete initial versions of projects with some added flexibility to work at their own pace. The physical classroom environment was also not especially conducive to students being able to both observe teacher instruction and work on programming tasks. This was a change from past implementations that featured more of a follow-along programming style, with the instructor crafting and explaining each component of a project while students were expected to keep up and build the program themselves. Instead, using the prepared teacher-focused materials as a starting point, we crafted step-by-step documentation for each project. These featured descriptions of relevant blocks and concepts and instructions on how to slowly put together a full-featured program. The documentation could then be provided to students via their usual learning management system. Thanks to a dual computer screen set up at most workstations, students could have screens dedicated to both the instructions and the programming environment.

A key feature of the co-design process was a sequence of interviews that the author conducted with the teacher at the conclusion of each module. Additional interviews were also interspersed through the longer final module. These interviews followed the same set of structured questions:

- What areas of the module did you most notice students struggling during?
- What areas of the module did students perform better than you would have expected?
- What areas of the module seemed to be most or least engaging for students?
- Do you have any proposed changes for the future based on your observations during the module?

- Did you notice any significant differences between the different grades (as in 7th- and 8th-grade students)?
- Have we neglected to convey any key knowledge components in the module that should be included in the future?
- How was the pacing of the module? Too much, not enough, or the correct quantity of content?
- In your opinion, how effective were various components of the module?
- Are there any other areas that need to be addressed for future work or further comments?

Answers from these questions will be included in the Discussion (6) and Conclusions (7) chapters. This will help frame our overall findings and define directions for future work. In the following sub-sections, we detail the individual curriculum modules.

4.1.1 Module 1 - Introduction to NetsBlox and CT

Module 1 included a cat-and-mouse game played entirely within the simulation window of NetsBlox. It was modified from a cybersecurity camp version that was created for high school students to better accommodate the 7th- and 8th-grade students participating in this intervention. Primarily, the project was scaffolded to not require the students to immediately write programs with more advanced concepts, such as loops and conditionals. The demonstration for this module was a full version of the project, with the students given time to play the game and answer some guiding questions about how the game worked based on their experiences with the game.

They next moved on to building parts of the project by themselves. As this was their first introduction to NetsBlox and block programming in general, their tasks primarily focused on some basic game behavior such as moving the player sprite around along with manipulating the background and sprite images. The behavior of the adversary sprite and other more complicated components were pre-created for the students. There was also a brief introduction to sensing blocks as these would be more relevant later on.

The debugging task featured a template project that was intended to introduce diagonal movement in place of the standard up/down/left/right movement. The template was intentionally erroneous. Students had to find these errors by playing the game and looking through the code. Extensions to this project were primarily focused on other alternative approaches to moving the player sprite around. One example would be to have the sprite turn in place left or right instead of moving left or right. This option was intended to set up the idea of robot driving, which follows a similar set of movement rules.

4.1.2 Module 2 - Loops, Variables, and Custom Blocks

Module 2 is an adaptation of the previously developed square art project. It again started with a demonstration of the full project, giving students time to draw various shapes while engaging with the material and hopefully getting some ideas for extensions down the line. Students completed a survey about their findings during this exploration. They then participated in a guided building of some parts of the project, learning about loops, variables, and the use of custom blocks along the way. The project was also extended into drawing shapes with more sides, with the end goal of developing code that could be used to draw a polygon of any number of sides and of any length. This code was then inserted into a “drawPolygon” custom block, which could be used multiple times within a program, greatly shortening the length of that program.

Their next task involved debugging a ticking clock project that was re-purposed from here, which was itself an adaptation from the Beauty and Joy of Computing course. This also required students to think about the concepts they learned in a whole new domain while keeping the same principles of sprite movement and drawing. Our version of this implementation is shown in Appendix C.1. Also featured in the Appendix is the quiz that was assigned as a formative assessment in the module and the survey that was included with the debugging task. A final set of images is related to the final project for the module, which required the students to create shapes using the “drawPolygon” block to meet a specified set of requirements. We provided examples of strong and weak implementations as well.

4.1.3 Module 3 - Conditionals and Lists

Originally, Module 3’s task was for students to create an application that mimicked the real-world behavior of creating a shopping list before going to the store. A full demonstration project was once again provided to students to start things off, allowing them to try out how the final project should work. This then framed the rest of the process. They began with a simple example of only keeping track of one particular item. Next, they extended the application to ask for multiple items. However, in the current state, the program could only “remember” the last item entered. This led to the idea of lists, which could be used to keep track of multiple items at once in a specific order. The final version totaled the number of items in this virtual shopping cart and also included extensions focusing on finding the total cost of all items as well as the minimum, maximum, and average cost among items in the cart. Conditionals were also important for the implementation. They were required to perform tasks such as checking to see if an item was already on the list and if one value was larger or smaller than another.

The debugging project associated with this module made use of the autograding capabilities recently introduced into NetsBlox. A Parsons problem was created challenging students to fill in the blanks of a custom block that checked to see if a specific word was present in a list of words.

As a result of feedback provided by the teacher and his students during the trial study implementation, it was discovered that this module was not appropriately fulfilling the needs of the students. They were not engaging with the shopping cart application at the same level as the previous projects. The debugging task via a Parsons problem also did not work as intended. Students were simply hard-coding their programs for the conditions of the test cases instead of creating a general solution to the presented problem. It is possible that similar to past literature [Grover and Basu, 2017] students were struggling to learn these difficult concepts with the initial format. This was then leading to a lack of engagement and the circumventing of intended approaches. Though we cannot quantitatively discuss these results from this trial study, there was enough support from observations to move forward with overhauling the existing Module.

It was instead agreed upon to focus on creating music with NetsBlox. The demonstration component now showcased a few methods for combining multiple musical notes and beat lengths. By “beat lengths”, we mean the duration of each note. This project also included more traditional use of lists, such as storing the results of several math operations on a list and summing the items of the list at the end. The guided project for the unit provided more background information on the “Sound” category of blocks within NetsBlox. The students then had to add all of the blocks to generate a song for their project.

This resulted in an inefficient program with twenty or more blocks stacked together. The solution for this was to use lists, loops, and eventually conditionals in an effective manner. A sample student-facing document given as part of this final transition to conditionals is shown in Appendix C.2. Musical notes and beat lengths were added into separate lists. These lists were then iterated through one item at a time. The beat lengths were matched with either the specified musical note or the length of a rest period in between notes. This process resulted in much more consolidated code.

The debugging task featured a further extension in the same direction for generating music. Students were given a broken piano featuring simple bugs like a key moving to the wrong location and more complex bugs like reversed conditional statements or improper variable initialization. They went through the project on their own before coming together as a class to discuss any errors that were observed. They then went back to their programs to try to remedy as many bugs as possible. The final task involved a combination of these new music ideas, numerical lists, and the formerly completed game project. The students added background music and game over music to their game while implementing a high score table.

4.1.4 Module 4 - Messages, Networking Introduction, and RPC's

Module 4 was based on past messaging applications in order to introduce messages, basic networking, and remote procedure calls (RPCs) in NetsBlox. The demonstration involved completed versions of the teacher/server-side as well as the student/client-side, with students able to send messages to the server to be

displayed on the screen in their classroom. This task also had a survey that asked students to explain certain features of this chat room. In particular, the chat automatically shortened very long chat messages, and a cool-down between chat messages was enforced. This change was a direct result of findings from the trial study implementation.

Next, students created a simpler version of this chat room, focused on direct messaging from one student to another. The challenge task for this module was a bit unorthodox. It was meant to test students' critical thinking skills rather than a specific debugging activity. It was based upon a networking problem to guarantee if a message was successfully received or not by sending a confirmation message back to the original person.

The final extension to this module was for students to finish creating a chat room for themselves. Some components of the code were already completed, making this more of a Parsons problem than a truly novel creation. The project was still fairly complex, however, as the chat room needed to properly handle receiving and acknowledging join requests from users. Another requirement was for received messages to be sent to all users in the room other than the sender. Students were able to test their code at any time using simple autonomous agents that would follow certain joining and messaging behaviors when requested. The project write-up and sample stronger and weaker implementations are included in Appendix C.3.

4.1.5 Module 5 - Robot Driving, Networks, and Network Security

Module 5 was quite extensive, but we cover the main points here. Students began the module by exploring a project intended to introduce RoboScape and the available commands of that Remote Procedure Call (RPC). They then applied those commands to a few initial robot driving programs. They began by manually issuing commands to the robot to drive it around and access the available sensors, such as the range sensors and whiskers.

They also completed autonomous driving projects. The first step was just to drive forward and turn, which they built upon in the following project to drive in a complete square around their chair. This autonomous driving task and the next topology task also involved some of our featured surveys with a large variety of answers from students providing interesting data points for analysis.

The next step in the process was an introduction to networking, with a specific focus on learning about different topologies. Demonstrations for these were handled in a few different ways. The initial idea was for simple embodied activities, such as passing around pieces of paper representing messages. These techniques were still used to some degree, but after receiving feedback from the committee, the primary implementation instead drew inspiration from NetLogo [Tisue and Wilensky, 2004] and C2STEM [Hutchins et al., 2020b]. We first provided virtual representations of the Bus and Star topologies. Students learned that these topologies were relatively cheap from a resource perspective but had trouble quickly communicating between nodes.

Those two methodologies also are very rigid in the direction of communication.

Ring and Mesh topologies solve those issues – at the expense of increased cost – and were presented next. Students explored working versions of these topologies in the NetsBlox environment. In both virtual and physical demonstrations, the main concepts were grounded within more common concepts from other domains as a scaffolding technique. For example, home postal addresses were a proxy for IP and MAC addresses. The debugging task involved determining the problems in one or more of the originally presented topologies. The extension related to this set of topics involved using addressing techniques in RoboScape in order to locate a particular robot from a group of robots. The general idea was to autonomously iterate through the list of robots returned by the “getRobots” command and determine which one was being singled out.

The next overarching networking topic was networking protocols, split into packet-related protocols – Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Protocol (IP) – and web protocols – Domain Name System (DNS) and Hypertext Transfer Protocol (HTTP). The embodied demonstrations were similar to before, with an emphasis on distinguishing between the different practices incorporated into TCP and UDP in particular. This time, the protocols were reinforced via a program using the robots. UDP was explained as the standard communication technique. Robots could be sent a large number of commands and would act upon them as quickly as possible. They did not consider the order of the sent commands or verify that the command was successfully carried out before moving on to the next command. Meanwhile, the slower but more accurate TCP could be replicated on the robots using a few options already available in NetsBlox. A “repeat until” block allowed the user to repeatedly send commands until the command was successfully received.

Additionally, the sequence numbering features of NetsBlox corresponded to packet numbers in normal communication to match with that component of TCP. The debugging project again fell more into the realm of Parsons problems, with the goal of ending up with a program that listened to one robot and the commands it receives before passing those commands onto a second robot. The idea was to mimic a basic follow-the-leader game, with the second robot following the same actions that the first one took.

Extensions to this took two potential routes. One was more of a “do the opposite” program that had the second robot receive set speed commands that were the same as the first robot but multiplied by -1 to reverse the direction. The second alternative was to use the message-passing features from Module 4 as the communication method for relaying information from one robot to another.

After this, the network security portion of the module began. We started by covering examples of passive attacks - in which no changes are made to the data or service - as well as active attacks where service is interrupted. This involved demonstrations of eavesdropping, denial of service, and man-in-the-middle attacks

through embodied activities and simple NetsBlox demonstrations shown to the whole class to start things off.

The same attacks were then implemented on the robot to reinforce the concepts and further build upon students' programming knowledge and skills. Next, the logical step was to begin securing the robot against such attacks through the use of encryption. We showed students how to apply Caesar's cipher manually with some simple examples and reading materials. We could not really show off the protocols on the robots, so the robot portion would primarily involve the use of encryption to ensure safe communication with robots. This is fairly similar to HTTPS and that comparison was made.

Students went on to perform their own encryption implementation on a basic manual driving controller. Other students along with the researcher attempted attacks on the robot during this time, so proper encryption was necessary to fend off these basic attacks. The debugging task associated with this area of the module was to determine the program errors within a brute force attack implementation. All of this could then be extended with additional robot security techniques, with students having the option of exploring things like the "encrypt++" blocks that mimic the Vigenère cipher to prevent brute force attacks or sequence numbering to prevent replay attacks.

Next was a multi-stage final project of progressively more complex attack and defense techniques incorporated between different rounds of an obstacle course race. Students were particularly engaged in this process, coming up with inventive solutions to locking down the robots of other groups. It is also worth mentioning that Module 5 in particular was developed with the intention of students working in groups of two or three, with some additional opportunities for multiple groups of students to either collaborate or compete.

One final component separate from the main learning environment and robotics platform was the activity on secure password creation and use. This was intended as a means of engaging students with the main ideas of human roles in terms of cyber and network security to match with the observed standards (Table 2.1) and recommendations from my committee.

4.2 Learning Environment

4.2.1 The NetsBlox Environment

NetsBlox [Broll et al., 2017] was selected as the primary learning environment for a variety of reasons. Its block-based interface is appropriate for providing scaffolding to students as they first learn programming and CT without causing excessive cognitive load upon students with complex syntax that would be present in text-based approaches. Its networking features including message passing between students, communication from sprite to sprite, and commands sent along with information received from the robots via RPC's are all instrumental for implementing the networking sections of our curriculum.

The existing RoboScape service allows for remote control of the robots and the existing cybersecurity

framework allows for various encryption and decryption techniques. These were necessary components for the desired hands-on project-based experiences to aid students as they learned cybersecurity and networking concepts while reinforcing the CT skills gained while working directly in NetsBlox. Overall, though we explored other options as outlined in Sections 2.2.2, 2.3.2, and 4.2.2 in particular, NetsBlox made the most sense in practical terms for providing a stable environment for us to build from.

On top of the already mentioned cybersecurity and networking features, we made use of a few other NetsBlox tools to establish the curriculum. We were able to create templates of projects already including sections of the code in order to scaffold student learning towards the important concepts and reduce the necessary programming time. This does have to be balanced with the desire to keep tasks open-ended, but with a light touch, these templates were a valuable tool to guide students. They were easily distributed amongst students, teachers, and the research team as well via shareable links. This technique was, in particular, used for the first module to simplify the introductory project as well as during the demonstration tasks to showcase how a completed program would work before allowing students to explore on their own on the way to replicating the finished product.

4.2.2 The RoboScape Environment and Physical Robotics Platform

The original vision for this work was to build upon the physical robots already in use with the NetsBlox programming environment and the RoboScape RPC - Parallax ActivityBot 360 robots [act, 2021]. With the inclusion of a WiFi module, these robots are capable of communicating with the NetsBlox server wirelessly. This was a step up from many previous robotics solutions requiring a USB cable to be connected to the robot to update any related programs. The robot also features a pair of optical encoders on the wheels to aid with the tracking of the motion of the robot, an ultrasonic range sensor for detecting obstacles ahead of the robot, and a pair of touch sensors in the form of “whiskers” to detect when contact with an obstacle is made. We also considered a few virtual robotics solutions. Gordon Stein as part of Ákos Lédeczi’s lab created a simple, browser-based version of a simulated robot environment. Other options included existing physics simulation tools such as Gazebo (available at <http://gazebosim.org/>) and Webots (available at <https://cyberbotics.com/>), which are compared in [Ayala et al., 2020]. This was supported by a recently published list of simulators in educational robotics [Tselegkaridis and Sapounidis, 2021].

Most options did not fit our needs. They generally required moving away from the desired block-based programming environment most suitable for middle school students learning to program for the first time. Gazebo was also very computationally demanding for the hardware available. Webots was determined to be the most feasible option at the time, and several months were spent exploring this space. It featured a similar virtual robot to the ActivityBot to ease that component of the process. It was also possible to add

the additional sensors present on the ActivityBot without much additional complexity while recreating the controller software that was used to communicate with NetsBlox. Eventually, the biggest limitation was determined to be the hardware that we had available to us. Even though Webots was less computationally demanding when compared to other solutions, our server was not capable of handling multiple robots, which could be a requirement for the studies to come with many groups of students working at once. The cost of upgrading our equipment when combined with the additional development time that would be required was determined to be a final roadblock on this route, and it was ultimately abandoned.

All of this eventually led to a collaboration between Gordon Stein and me working as part of Dr. Ákos Lédeczi's lab to create a simulated robotics solution using Unity (available at <https://unity.com/>). Gordon had already created a baseline version of the majority of the needed components for such an environment as well as setting up the networking back-end of the simulation in order to communicate between the robots and NetsBlox and to synchronize the robots across multiple computers connected to the same server. More information on Gordon's work can be found in [Stein and Lédeczi, 2021].

My contributions were centered on creating and testing specific environments for a cybersecurity camp in the Summer of 2021, including the autonomous square driving task, the tug-of-war competition, and various obstacle courses for students to navigate with their robots. These required further collaboration with Gordon to create needed scripts to track values over time (and synchronize those values across multiple students) and to otherwise enable appropriate game behavior for each of these tasks. These scenarios were actually an affordance of the new simulated system. During previous summer camps with the physical robots, we had to manually track the time it took for students to complete objectives, the number of obstacles that the robot ran into to be counted as penalties, and other such simple but distracting tasks. We also had to deal with constantly carrying the physical robots back to the starting positions, replenishing batteries running out of juice, or batteries constantly falling out of the robot and needing to be replaced. All of these and more could be largely automated in the simulated environment, allowing for competitions to be executed much more efficiently. Though we did ultimately decide on the physical robots for our primary intervention for a truly hands-on experience, these affordances were lost in the process.

Another already developed component of this environment that I created was a logging system to track the position of each robot, the commands received by the robot, other triggers sent from the robot to NetsBlox, and whenever a user joined or left a particular server. This information served as a supplement to data sources such as NetsBlox logs and audio plus visual recordings in order to form a more comprehensive picture of student programming and other activities in the environment. Future work in this area would require additional environments to be created more suitable for the networking-related tasks that have been developed as part of the current curriculum as described in Section 4.1.

As stated, we did return to in-person studies using only the physical robots. This does not completely invalidate the work done in the simulated realm however, as we were able to gain a better understanding of the needs of students that can be addressed by either version of the robots and the strengths and weaknesses of each option.

4.3 Assessments

Assessment development was guided by the overall plan for the research as shown in Figure 3.1. The set of assessments included: (1) summative, pre-post assessments targeting overall learning gains in CT, Networking and, Network Security and (2) embedded assessments supporting a deeper understanding of knowledge construction through the curriculum (discussed in the module design above).

4.3.1 Pre-Post Assessments

There are three components to our suite of comprehensive pre- and post-assessments: (1) attitudinal surveys; (2) a CT test; and (3) a combined networking and network security test.

4.3.1.1 Survey

The first assessment was a survey composed of Likert-scale questions adapted from studies of motivational profiles [Linnenbrink-Garcia et al., 2018] and CS attitudes [Haynie and Packman, 2017], with two questions chosen per profile or attitude. Each made use of a scale ranging from “Strongly Disagree” to “Strongly Agree” corresponding to scores from 1 to 5. In addition, modified Likert-scale questions ranging from either “Very Interested” to “Not at all interested” or from “Very confident” to “Not at all confident” were used from an NCWIT survey that was specifically developed to measure the interest in and confidence in computing and computing skills among middle school students [for Women & Information Technology,]. A collaboration component adapted from the STUDENT QUESTIONNAIRE FOR PISA 2015 (available at <https://www.oecd.org/pisa/data/2015database/>) was added to the pre-surveys to measure initial openness to collaboration with peers (i.e., other students) before being fully integrated into the post-surveys to observe if any changes had occurred after the collaborative robotics challenges at the end of the curriculum.

Overall, these surveys were designed to capture the self-efficacy, task value, tech attitudes, and collaboration perceptions of students along with other areas of interest. Additional open-ended responses on the post-survey attempted to capture student attitudes towards different pedagogical and scaffolding techniques incorporated into the curriculum. This was intended to be combined with other data sources, such as interviews with the teacher with the goal of suggesting refinements to the curriculum because of this feedback. The full post-survey is presented in Appendix B.3. The pre-survey was almost identical to the post-survey.

4.3.1.2 CT

The second component - the CT assessment - was created based on our previous work for the cybersecurity camps while also drawing from sources such as the ECS assessments and the work of Wiebe [Wiebe et al., 2019], Román-González [Román-González et al., 2017], and an approach from Rachmatullah et al. that built upon those examples [Rachmatullah et al., 2020]. We targeted the CT concepts of loops, conditionals, and data handling using variables, lists, and sequences as these are key features of the curriculum covered during both the NetsBlox and the robotics projects. As stated, most questions were adapted from the sources discussed above, but a custom question that focused on lists with some elements of variables and loops was gap in the topics covered by the assessment. The CT test is presented in Appendix B.1.

4.3.1.3 Networking and Network Security

The final component - a networking and network security assessment - was constructed based on previous work for a number of summer camps run at Institute for Software Integrated Systems (ISIS) and led by Dr. Lédeczi. The assessment was revised to correlate with the networking focus of our curriculum. Khan Academy's content (Note: All Khan Academy content is available for free at www.khanacademy.org) and sample questions within the AP CSP course - specifically units on "The Internet" and "Online data security" (available at <https://www.khanacademy.org/computing/ap-computer-science-principles>) - were particularly useful. However, it was necessary to keep in mind the experience and knowledge gaps between the high school students this material was developed for versus the middle school students we worked with. The overarching goal with this pair of tests (see Appendix B.2 for full implementation) was a focus on connecting learning on definitely unknown topics in networking and cybersecurity with pre-existing and developed knowledge in the more commonly covered area of CT.

4.3.2 Embedded Assessments

Two different types of embedded assessments were developed to interpret student understanding over time. These were also useful for the teacher as a low-stakes means for grading students as they participated in the curriculum. One form of embedded assessment was directly built into the curriculum modules in the form of student-created projects. We built upon previous work [Yett et al., 2020a] as well as other related examples, such as the Evidence-Centered Design framework [Mislevy et al., 2003; Mislevy and Riconscente, 2005; Mislevy and Haertel, 2006] (ECD) to create appropriate rubrics. These rubrics are an additional piece of the comprehensive assessment puzzle, allowing us to gather and interpret evidence of student learning through their programming projects. They are intended to provide a deeper understanding of students' programming while supplying a baseline version for the development of project grading tools.

Our previous grading was done post hoc, but the use of the evidence-centered rubrics allowed us to collect programming action data during the preliminary study of Fall 2021. This informed the development of the rubrics which were then applied in Spring 2022. We identified key components of each of the major programming projects throughout the curriculum and manually analyzed each project to determine to what extent each student implemented the desired components. One difficulty was the different techniques students applied to solving the same problems in our open-ended learning environment of NetsBlox, all of which must be accounted for in some way.

Another component of the embedded assessment plan involved homework and quizzes covering specific details in each module in a much more focused manner than the pre- and post-assessments. These were meant to provide a zoomed-in view of student learning. Each module had at least two such assessments. For example, the assessments for Module 2 required students to understand and demonstrate their understanding of loops, with a step-by-step breakdown of the commands (aka NetsBlox blocks) being used and their effects. The current version of our assessments are displayed in Section C.

The homework and quizzes were co-designed with the teacher in order to best capture student learning while providing usable results to the teacher as well. Overall, these assessments greatly aided our understanding of student learning as they progressed in the curriculum. It also allowed us to determine the current strengths and weaknesses of the curriculum.

4.4 Data Collection and Processing

Most of the data collection was discussed in Section 3.3 along with the research questions. We collected data from the pre- and post-tests and surveys in order to assess student learning and the effectiveness of the curriculum, leading to the identification of areas in need of improvement. Other data was collected from formative assessments in the form of homework, quizzes, and mid-module surveys. We also accessed programming projects and logs for the purposes of quantifying actions as a measure of engagement and to evaluate student projects in two ways: (1) grading of solutions (shown as part of RQ2 in Section 5.2) and (2) progression in programming behaviors (shown as part of RQ3 in Section 5.3).

4.4.1 Assessments

To ensure result validity, we used identical pre-test and post-test questions for CT, networking, and security. We emphasized well-framed multiple-choice questions for reduced complexity while maintaining a high degree of reliability. The majority of study data was collected and managed using REDCap electronic data capture tools hosted at Vanderbilt [Harris et al., 2009; Harris et al., 2019]. REDCap (Research Electronic Data Capture) is a secure, web-based software platform designed to support data capture for research studies.

Other data such as in-class homework and quizzes were handwritten initially. They were then de-referenced and transferred to secure digital locations before the physical versions were returned to the teacher. In general, students were given a de-identified ID that doubled as their NetsBlox username, which allowed us to develop easy mechanisms for linking and analysing student data post hoc.

4.4.2 Surveys

Surveys were handled similarly to our assessments, with identical or matching questions or statements used for pre- and post-versions. Some of the mid-Module surveys were handwritten initially and needed to be manually entered into secure digital locations.

A select number of these surveys with the largest portion of student responses that generated valuable insights into student reasoning processes were chosen for a more thorough coding process. Initial rubrics to categorize student responses were created by the author with feedback from two other researchers. At least one-quarter of student responses on each survey were then coded to verify the effectiveness of the coding scheme and obtain an inter-rater reliability (IRR) score using Cohen's kappa [Cohen, 1960]. For three of the four surveys of interest, this was sufficient to obtain high IRR ($\kappa = 0.82$, $\kappa = 0.91$, and $\kappa = 0.87$). However, IRR was lower than desired for the fourth survey ($\kappa = 0.56$). The author and one researcher met to reconcile differences across all four surveys, with a particular emphasis on the problematic fourth survey. After discussion, there was a better understanding of the coding scheme, and the inter-rater reliability increased significantly ($\kappa = 0.90$). The author and researcher then split the remaining surveys to complete the full coding process. The codes themselves were primarily a conduit to then fit within one of the types of reasoning (inspired by [Hutchins et al., 2021]) displayed in Table 4.1. In other words, similar student responses were grouped together via the coding scheme, and all of those responses were associated with one particular type of reasoning. The mapping of codes to reasonings was performed by the author with feedback from the other researchers. This information was processed as part of our observations on student strategy progression in Section 5.3.

4.4.3 NetsBlox Projects

NetsBlox projects were coded and analyzed in two modes. One involved the actual project files and manually scoring student performance. The other involved studying logged actions in context and autonomously generating models and performing sequence mining.

Table 4.1 Reasoning codes, listed in order from highest to lowest reasoning levels.

Reasoning Type	Description
Mechanistic	Describing how the components of a system influence the system's behavior [Russ et al., 2008]
Causal	Using logic and facts to reach conclusions about cause and effect relationships
Numerical	Demonstrating knowledge of correct and relevant numerical values without exploring causal relationships
Visual	Roughly equivalent to numerical, but relating non-numerical information shown on screen to answers instead
Developing	Answering with information that is missing key points or focusing on components unrelated to the question
Minimal	Answering incorrectly or failing to adequately reach any other reasoning type with answers

4.4.3.1 Project Files

Extensible Markup Language (XML) files were accessed from the NetsBlox database based on the set of usernames used by our students. These files were then used to search through student projects to find the primary solution to each of the four projects. Projects were scored based on a set of criteria established during the co-design process with input from the teacher. We considered how to best measure student learning within the context of the module and within the ECD [Mislevy et al., 2003] paradigm. Students could earn full, partial, or no credit for each category of the developed rubrics. Scores were then simplified to percentages of total possible points for comparison across the full set of projects. The teacher used these projects as grades for the course. Separately, I assessed project quality using the same rubric that had been given to students. These are the scores that were then used to evaluate student performance throughout the results (Chapter 5).

4.4.3.2 NetsBlox Logs

Actions were queried from the NetsBlox database by pre-processing logs for the specific set of usernames assigned to students during the intervention. Each action type was grouped into a category, some of which are simply re-naming of the initial action, but some of which encompass multiple related action types. The results of this process including a description of each action name are shown in Table 4.2. Actions are further grouped into three overarching categories as part of our task model. These categories are described as follows:

- Solution Construction (SC) - Actions related to building up the solution or modifying the existing program.
- Solution Assessment (SA) - Actions related to executing all or part of the existing program. Also includes manipulating an ongoing execution process.

- Solution Independent (SI) - Actions related to code organization or modifying visual features of the program.

Solution Construction and Solution Assessment have been featured in our previous work (for example, [Yett et al., 2020b]). Solution Independent is a new attempt to classify behaviors that may not directly move the project toward the desired objective, but may still have value. These included tracking when students view different components of the program stored on different sprites, when students were customizing their projects in some way, or when students were re-organizing their programs to better understand their code. This hierarchy is further visualized in Figure 4.1.

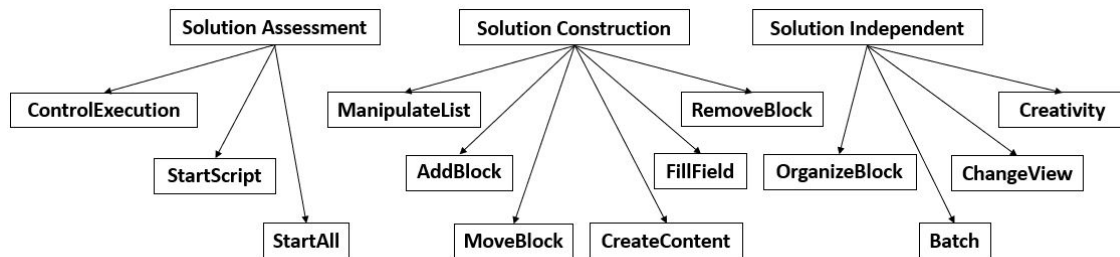


Figure 4.1: Task model categorizing our Action Categories into Solution Assessment, Solution Construction, and Solution Independent

4.5 Study Demographics and Description

Studies were implemented in two phases. A trial study was run in Fall 2021. The trial nature of that study was necessitated by school district restrictions limiting the presence of researchers in the classroom. The teacher involved also missed multiple weeks due to family illness. This all led to an inability to collect publishable data in the trial study between the complications entering the classroom and permissions issues through the school district and Vanderbilt’s IRB.

As was outlined further in the discussion on Module 3 (Section 4.1.3), we were still able to make classroom observations and improve upon our curriculum with the goal of increasing learning and engagement. The full studies were implemented in Spring 2022. They lasted the entire semester, with the pre-test occurring in late January and the post-test concluding the curriculum in late May. The studies took place in both 7th- and 8th-grade classrooms, where each class had entirely different students for each semester of the school year. This provided a clean slate for the Spring semester after refining the process used during the Fall.

The curriculum and assessments used were previously described in Sections 4.1 and 4.3 respectively. Updates were made as necessary, with the teacher participating in the co-design of various components as previously outlined. Demographic information of the students who fully participated in our Spring 2022

Table 4.2 Selected actions related to either general programming behaviors or specifics of the projects being completed.

Action Name	Action Category	Description
addBlock	AddBlock (AddBlk)	Add a block directly from the categories on the left to the Scripts window. This also accounts for when one or more blocks are duplicated and placed in the Scripts window. Note that this does not account for blocks immediately attached to other blocks, which is considered a “moveBlock” instead.
batch	Batch	Occurs when the “undo” button is pressed, and contains information on which actions were undone.
selectSprite	ChangeView	Change the view from one sprite to another - a possible technique for separating code for organizational purposes.
selectTab	ChangeView	Switch tabs (e.g. scripts to simulation).
addSprite	ChangeView	Create a completely new sprite.
duplicateSprite	ChangeView	Duplicate an existing sprite.
togglePause	ControlExecution (ControlEx)	Press the Pause button to temporarily halt program execution.
stopAllScripts	ControlExecution (ControlEx)	Press the Stop button to completely halt program execution.
addCustomBlock	CreateContent	Create a new custom block.
addVariable	CreateContent	Create a new custom variable.
setColorField	Creativity	Similar to setField, but will specifically relate to the pen color.
addCostume	Creativity	Add a costume from the available set of costumes.
updateCostume	Creativity	Change the costume for a sprite or the stage.
setField	FillField	Edit a block parameter. This includes drop-down options for set/change blocks for custom variables, typing values into empty spots within other blocks, etc.
addListInput	ManipulateList (ManipList)	Add a new blank item to a list block. This includes things like script variables that can have slots added to them as well.
removeListInput	ManipulateList (ManipList)	Remove the last item from a list block. This includes things like script variables that can have slots removed from them as well.
moveBlock	MoveBlock (MoveBlk)	Could indicate adding a completely new block to the end of a script, or removing one or more blocks from one script and adding them to another script.
setBlockPosition	OrganizeBlock (OrganizeBlk)	Either disconnect a block from other blocks and place it by itself somewhere in the Scripts window, or pick up a block and place it somewhere in the Scripts window still not attached to any other blocks.
removeBlock	RemoveBlock (RemoveBlk)	Remove one or more blocks from the Scripts window, either by dragging to the appropriate part of the screen to delete them or by right-clicking and choosing “delete”.
pressStart	StartAll	Press the Green Flag button to initialize program execution.
startScript	StartScript	Click on a hat block or a script to initialize script execution (NOT full program execution).

Table 4.3 Demographic information of the $n = 48$ students who were full participants in the Spring 2022 semester-long study. Students were free to choose multiple options for race, so the total for those classifications will sum to $> 100\%$

Classification	Percentage
8th grade	60.42%
7th grade	39.58%
Female	53.13%
Male	46.87%
Native American/Alaska Native/First Nations	2.08%
Asian	16.67%
Black or African-American	37.5%
White	43.75%
Hispanic/Latinx	2.08%
Middle Eastern/North African	6.25 %
Other	5.21 %

studies is shown in Table 4.3. Our student population featured roughly 60% 8th-grade students, 53% female students, and a heavy concentration of Asian (17%), Black or African-American (38%), and White (44%) students.

4.6 Summary of Chapter

Throughout this Chapter of the dissertation, we presented the results of the planning and development phases. This began with the Curriculum Design (Section 4.1), including the co-design process to identify new materials needed and continuing until full revisions of certain Modules were completed. Each Module was described in detail, starting with the CT and NetsBlox focus and concluding with advanced networking and security content. The specific Learning Environment (Section 4.2.1) was a combination of NetsBlox for block-based programming and RoboScape plus Parallax Activity Bots for the hands-on component. This combination has been implemented successfully in the past ([Lédeczi et al., 2019],[Yett et al., 2020a]). Each representation provides visual feedback to students based on their programming actions, aiding and scaffolding their learning.

The next discussed feature was the suite of assessments (Section 4.3). Comprehensive tests and self-assessments were replicated at both the beginning and the end of the intervention for a standardized view of learning gains and attitudinal progressions. Embedded assessments were instead implemented within each of the five Modules and provided snapshots of student understanding throughout the curriculum. Following this was an overview of the data that was collected and processed (Section 4.4). This was primarily meant to set the stage for our next Chapter on Results. Finally, a brief summary of information related to the in-classroom study was provided in Section 4.5.

CHAPTER 5

Results

We laid out our three research questions in Section 3.3. RQ1 focuses on the pre-post-assessments and surveys. We present descriptive statistics on the changes from pre to post in each case. Changes are measured via p-scores and Cohen's d to indicate significance and effect size respectively. Starting at the end of RQ1 and moving on to RQ2, we calculated Spearman correlations between a variety of metrics of student performance and attitudes. These are intended to show any potential relationships between these metrics.

In RQ2, we also study the progression of student learning over the course of the curriculum. This is accomplished via linear regression modeling of pre-tests, formative assessments, and the classroom final. The model is then applied towards predicting post-test performance, and the predictions are compared to actual post-test performance to assess the accuracy of the model.

RQ3 focuses on the strategy progressions of students as compared to the previous study of learning progression. We begin with Markov chain (MC) models and Differential Sequence Mining of student actions on selected projects. This serves to isolate model-building strategies of students. The final approach shown involves the progression of survey responses of students. These responses were coded to determine the level of reasoning students were able to apply at different times during the curriculum.

5.1 Research Question 1 - How did student performance, engagement, and attitudes measured using pre-post-assessments and surveys change as a result of our intervention?

This section discusses a number of measures that cover student performance, engagement, and attitudes. Assessments and surveys were conducted before and after the intervention to gather the required information. Surveys targeted student task value, self-efficacy, learning approaches, engagement, and computing confidence and interest (Table 5.6). We split our primary research question into a few sub-questions:

- What was the pre- to post-test change of content material for the students?
- What was the change in pre- to post-survey responses of students?

5.1.1 Pre-Post-Test Analysis of Content Questions

To determine the pre- to post-test performance differences among our students we computed: (1) the p-score, and (2) the effect size. The p-score is calculated via the two-tailed t-test [Fisher, 1936]. A p-score < 0.05 has long been considered as significant. However, this information does not adequately indicate the importance or direction of the observed learning changes.

The effect size as measured by Cohen's d [Cohen, 1992] metric, is calculated via the following formula:

$$d = \frac{\overline{Post} - \overline{Pre}}{\sqrt{(PostSD^2 + PreSD^2)/2}}$$

\overline{Post} indicates average post-test score, \overline{Pre} indicates average pre-test score, $PostSD$ indicates the standard deviation among post-test scores, and $PreSD$ indicates the standard deviation amongst pre-test scores. As an additional note, an effect size of 0.2 is considered low, 0.5 is considered moderate, and 0.8 or higher is considered high when comparing if two population means are equivalent. [Cohen, 1992].

For the CT pre- to post-test changes as shown in Table 5.1, students showed significant improvement ($p < 0.01$) though with a small effect size ($d = 0.13$). There were two other significant changes from pre- to post-test in the areas of Loops and Evaluation. These were both recurring themes throughout the intervention (see Figure 5.2 and Table 5.10). The increase in understanding of Loops had a small effect size ($p < 0.01$, $d = 0.38$), but the increase in Evaluation reached a moderate effect size ($p < 0.01$, $d = 0.55$) as students improved by 50% from pre- to post-test. The lone insignificant decrease occurred in the Testing and Debugging category, which was covered by one question.

The reason for the small overall effect size, for Conditionals in particular, seemed to partially stem from problem six - see Table 5.2 for the scoring and Appendix B for the specific question. Students' performance decreased significantly from pre- to post-test ($p = 0.04$). The primary issue seemed to be the nested conditional required for this problem, which led to a different correct answer than the similar condition present in problem three. Of the thirteen students who answered correctly on the pre-test and proceeded to answer incorrectly on the post-test, ten of those repeated the answer for problem three. Meanwhile, question five had the same correct answer as question two, which may have led to the significant ($p < 0.01$) and moderate ($d = 0.69$) increase on student performance for this problem. Other significant improvements are shown in Table 5.2 and tended to be concentrated in the specific subjects of loops and functions.

We saw a significant ($p < 0.01$) but small ($d = 0.34$) increase in student performance from pre- to post-test on Networking concepts (Table 5.3). There was only a slight improvement in scores in the messaging subcategory. Student understanding of network topologies appeared to significantly increase, with scores doubling from pre- to post-test (1.4 compared to 2.85). This increase was significant with a moderate effect size ($p < 0.01$, $d = 0.62$). For security-related concepts, their pre-post gains were significant ($p < 0.01$) but the effect size was small ($d = 0.33$). An important subcategory of interest was encryption, where students did not change much in terms of pre- to post-test performance.

Improvement on Networking concepts was concentrated in a few areas that were emphasized during the intervention (Section 4.1). Students demonstrated moderate or better learning gains on three questions

Table 5.1 Changes from pre- to post-test - Computational Thinking Overall and Categories. p-scores determined via two-tailed t-test. Negative effect size implies a decrease from pre- to post-test and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d .

Category	Max Points	Pre Average (SD)	Post Average (SD)	p-score	Effect Size
Overall CT	26	15.25 (3.07)	16.85 (3.57)	< 0.01	0.13
Variables	9	3.56 (1.55)	3.85 (1.46)	0.33	0.07
Conditionals	14	9.79 (2.08)	9.98 (2.14)	0.59	0.03
Loops	7	2.48 (1.19)	3.79 (1.77)	< 0.01	0.38
Evaluation	2	0.94 (0.69)	1.46 (0.64)	< 0.01	0.55
Testing and Debugging	1	0.75 (0.43)	0.67 (0.47)	0.37	-0.18

Table 5.2 Significant ($p < 0.05$) changes from pre- to post-test - Computational Thinking Individual Questions. p-scores determined via two-tailed t-test. Negative effect size implies a decrease from pre- to post-test and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d .

Number	Targeted Concepts	p-score	Effect Size
5	Conditionals	< 0.01	0.69
6	Conditionals	0.04	-0.43
15	Loops, Functions, and Sequencing	< 0.01	0.80
16	Loops and Functions	< 0.01	0.57
17	Loops and Functions	< 0.01	0.61
18	Loops and Lists	< 0.01	0.61

related to network topologies (Table 5.4). This included large effect sizes on question eight ($d = 1.00$) and question ten ($d = 0.86$), which were specifically related to star and mesh topologies, respectively. Students also demonstrated good understanding of the relationship between Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) with a moderate effect size ($d = 0.63$) and significant ($p < 0.01$) positive growth from pre- to post-test on question fifteen.

For the security-focused component of our summative assessments, improvements were primarily concentrated in student understanding of several common types of attacks that were discussed and then implemented on the robots during the intervention 5.5. Students demonstrated a significant ($p < 0.01$) improvement with a large effect size ($d = 0.91$) on a standalone question related to distributed denial of service attacks. They also improved on a related set of questions that required matching real-world examples to attack types, particularly on the man-in-the-middle ($p < 0.01, d = 0.57$) and brute force ($p < 0.01, d = 0.66$) portions.

Table 5.3 Changes from pre- to post-test - Networking and Security Overall and Categories. p-scores determined via two-tailed t-test. Negative effect size implies a decrease from pre- to post-test and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d .

Category	Max Points	Pre Average (SD)	Post Average (SD)	p-score	Effect Size
Overall Networking	13	5.17 (1.55)	7.35 (3.04)	< 0.01	0.34
Messaging	2	0.75 (0.83)	0.9 (0.85)	0.31	0.15
Topology	5	1.4 (0.99)	2.85 (1.68)	< 0.01	0.62
Overall Security	9	3.92 (2.08)	5.40 (2.14)	< 0.01	0.33
Encryption	2	0.5 (0.65)	0.54 (0.58)	0.74	0.05

Table 5.4 Significant ($p < 0.05$) changes from pre- to post-test - Networking Questions. p-scores determined via two-tailed t-test. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d .

Number	Targeted Concepts	p-score	Effect Size
8	Topology	< 0.01	1.00
9	Topology and Logic	0.06	0.39
10	Topology and Logic	< 0.01	0.86
11	Topology and Logic	0.01	0.51
12	Topology and Logic	0.04	0.42
14	Communication Protocols	0.06	0.39
15	Communication Protocols	< 0.01	0.63

Table 5.5 Significant ($p < 0.05$) changes from pre- to post-test - Security Questions. p-scores determined via two-tailed t-test. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen's d .

Number	Targeted Concepts	p-score	Effect Size
17	Denial of Service	< 0.01	0.91
18	Man in the Middle	< 0.01	0.57
20	Brute Force Attack	< 0.01	0.66
21	Eavesdropping	0.04	0.42

5.1.2 What were the pre- to post-survey changes of students?

As previously discussed in Section 4.3, the surveys targeted a wide variety of student beliefs and attitudes. These are summarized in Table 5.6. Unlike the domain-related assessments, there is no easy comprehensive score that summarizes all of these areas. This is especially true considering that some areas such as performance-avoidance goals (AV), performance-approach goals (AP), and mastery-approach goals (MA) were not expected to change significantly from pre- to post-survey [Midgley et al., 2000]. They are more ingrained within student beliefs about their relationship to school and schoolwork and less related to any particular course or intervention. Despite this, we saw the potential for relevant findings on a category-by-category and question-by-question basis.

Only a few of the surveys categories showed significant ($p < 0.05$) change in the pre- to post-surveys. Student responses to a series of seven questions related to Specific Computing Confidence (CC) increased significantly ($p = 0.03$) with a small-to-moderate effect size ($d = 0.44$). On the other hand, student scores decreased for the single question dedicated to Specific Behavioral Engagement ($p = 0.01$, $d = -0.58$). This result is partially supported by the set of seven questions targeting Specific Computing Interest ($p = 0.06$, $d = -0.38$). No other categories showed significant changes from the pre- to post-surveys; as expected there were no significant changes in the areas of MA, AV, or AP.

Findings related to specific questions are shown in Table 5.8. The first question listed was related to the Task Value of students. The question was necessarily phrased slightly differently on the pre-survey - “I think the things I will learn in this class will be useful”. Students had significantly ($p < 0.01$) lower responses with a small-to-moderate effect size ($d = -0.44$) on the post-survey as compared to the pre-survey. The second question noted in Table 5.8 was taken from a Quantity of Self-Regulation scale initially established in [Linnenbrink, 2005] before a re-framing into Behavioral Engagement in [Linnenbrink-Garcia et al., 2018]. Early work ([Linnenbrink, 2005]) had established a positive correlation between this area and generally desirable traits such as Mastery-approach goals (see Table 5.6). Students once again responded with significantly ($p < 0.01$) lower agreement when comparing pre- to post-survey results, with a moderate effect size ($d = -0.58$).

In contrast, students appeared to demonstrate a significant increase in their computing confidence via three questions all dedicated to this attitude. The largest impact ($p < 0.01$, $d = 0.63$) of change from pre- to post-survey responses belonged to the fourth question listed - “Right now, how confident are you in your ability to design computer games?”. This corresponds with our Module 1 content that was dedicated to modifying and creating certain components of the cat-and-mouse game. Other questions related to finding solutions to world problems and learning about CS are listed in the third and fifth rows of Table 5.8. Students responded

Table 5.6 Summary of abbreviations used in upcoming tables related to survey data.

Abbreviation	Full-Text	Explanation
TV	Task Value	Perceived worth associated with a domain or task [Eccles, 1983].
MA	Mastery-approach goals	“When oriented to mastery goals, students’ purpose or goal in an achievement setting is to develop their competence. They seek to extend their mastery and understanding. Attention is focused on the task. A mastery goal orientation has been associated with adaptive patterns of learning.” [Midgley et al., 2000]
AV	Performance-avoidance goals	“When oriented to performance-avoidance goals, students’ purpose or goal in an achievement setting is to avoid the demonstration of incompetence. Attention is focused on the self. A performance-avoid orientation has been associated with maladaptive patterns of learning.” [Midgley et al., 2000]
AP	Performance-approach goals	“When oriented to performance-approach goals, students’ purpose or goal in an achievement setting is to demonstrate their competence. Attention is focused on the self. A performance-approach orientation has been associated with both adaptive and maladaptive patterns of learning.” [Midgley et al., 2000]
PC	Perceived Competence	Related to the PALS [Midgley et al., 2000] measure of academic self-efficacy. In particular, students are questioned about their perceived competence to learn and to complete assignments.
BE	Behavioral Engagement	Meant to assess student persistence when working on assignments [Linnenbrink, 2005].
CE	Cognitive Engagement	Meant to assess a student’s ability to plan, monitor and evaluate their learning and performance. Questions were created based on scales from the MSLQ [Pintrich et al., 1993] and the work of [Fredricks et al., 2005].
CC	Computing Confidence	Questions formulated as “Right now, how confident are you in your ability to...” [for Women & Information Technology,]
CI	Computing Interest	Questions formulated as “Regardless of whether or not you have actually tried it, how interested are you in...” [for Women & Information Technology,]

Table 5.7 Changes from pre- to post-survey on a category-by-category level largely corresponding with Table 5.6. “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. p-scores determined via two-tailed t-test. Negative effect size implies a significant decrease from pre- to post-survey and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen’s d .

Category	Max Points	Pre Average (SD)	Post Average (SD)	p-score	Effect Size
General TV	5	3.35 (0.88)	3.15 (1)	0.29	-0.22
Specific TV	15	10.02 (2.52)	9.23 (2.91)	0.16	-0.29
General MA	5	3.73 (1.02)	3.77 (1)	0.84	0.04
Specific MA	5	3.81 (0.93)	3.5 (1.17)	0.16	-0.29
General AV	5	3.27 (1.27)	3.17 (1.18)	0.68	-0.08
Specific AV	5	2.5 (1.08)	2.65 (1.22)	0.54	0.13
General AP	5	2.48 (1.17)	2.63 (1.17)	0.55	0.12
Specific AP	5	2.15 (1.02)	2.5 (1.12)	0.11	0.33
General PC	5	3.79 (0.64)	3.83 (0.82)	0.79	0.06
Specific PC	10	7.67 (1.34)	7.46 (1.83)	0.53	-0.13
General BE	5	3.35 (1.07)	3.48 (0.96)	0.55	0.12
Specific BE	5	4.08 (0.79)	3.58 (0.91)	0.01	-0.58
General CE	5	3.67 (0.85)	3.79 (1.02)	0.52	0.13
Specific CE	10	6.54 (1.53)	6.44 (2.02)	0.78	-0.06
Specific CC	28	14.77 (4.64)	17 (5.33)	0.03	0.44
Specific CI	28	16.77 (5.19)	14.6 (5.95)	0.06	-0.38
Collaboration	20	14.15 (3.36)	14.23 (3.55)	0.91	0.02

Table 5.8 Significant ($p < 0.05$) changes from pre- to post-survey on a question-by-question level. p-scores determined via two-tailed t-test. Negative effect size implies a significant decrease from pre- to post-survey and is indicated by red text. Bold rows indicate a moderate ($d = 0.5$) or greater effect size as measured by Cohen’s d .

Question or Prompt	p-score	Effect Size
I think the things I learned in this class were useful	< 0.01	-0.44
Even if I don’t see the importance of a particular coding assignment, I still complete it	< 0.01	-0.58
Right now, how confident are you in your ability to find technological solutions to world problems using computer science?	0.03	0.46
Right now, how confident are you in your ability to design computer games?	< 0.01	0.63
Right now, how confident are you in your ability to learn computer science concepts?	0.05	0.41

significantly ($p = 0.03$ and $p = 0.05$, respectively) higher to each of these prompts on the post-survey with small-to-moderate effect sizes ($d = 0.46$ and $d = 0.41$ respectively).

An additional consideration of interest was whether age, gender, or race significantly impacted performance, attitudes, or engagement. We studied this using the Spearman correlational measure ([Cabo, 2019]). As part of our data (survey results) was of a ranked nature (Likert scores), it was determined that the Spearman correlation would be most appropriate. Spearman scores of 0.4 or higher are commonly treated as moderate [Akoglu, 2018]. The equation applied is:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)}$$

ρ , the Spearman's rank correlation coefficient is computed by comparing each d_i term is the difference between the ranks of each observation, and n is the total number of observations. These analyses were the only ones showing moderate impacts:

- White students tended to perform better on the security component of the pre-test ($\rho = 0.4$)
- Female students and Black/African-American students tended to perform worse during Module 5 ($\rho = -0.4$ and $\rho = -0.56$ respectively)
- Black/African-American students tended to perform worse on the Module 3 Project ($\rho = -0.47$)
- 8th-grade students tended to take fewer actions during Module 1 ($\rho = -0.43$)
- Female students and Black/African-American students tended to have more tardies ($\rho = 0.45$ and $\rho = 0.56$ respectively)
- Female students tended to have lower Specific Task Value scores on the post-survey ($\rho = -0.43$)

The full tables are presented in Appendix D.

We conclude this research question by considering the overall relationship between pre- and post-test performance in each of CT, networking, and security, as assessed via Spearman correlational analyses (Table 5.9). Though there were several areas with a low-to-moderate correlation noted in the table, here we will highlight only those with correlations greater than 0.5. Post-Test Networking met this requirement when correlated to both Pre-Test CT ($\rho = 0.55$) and Pre-Test Security ($\rho = 0.54$) performance. This could lend some credence to the idea that an initial understanding of CT aided students in learning networking. Additionally, Post-Test Networking and Post-Test Security were found to be moderately correlated with each other ($\rho = 0.56$).

Table 5.9 Spearman correlations are displayed as one measure of the relationship between a student’s pre-post-test performance in the main areas of CT, networking, and security. Positive correlations ≥ 0.4 are in bold and blue text.

Category	Pre-Test CT	Pre-Test Networking	Pre-Test Security	Post-Test CT	Post-Test Networking
Pre-Test Networking	0.28	-	-	-	-
Pre-Test Security	0.43	0.42	-	-	-
Post-Test CT	0.42	0.18	0.34	-	-
Post-Test Networking	0.55	0.23	0.54	0.24	-
Post-Test Security	0.14	0.26	0.46	0.32	0.56

5.2 Learning progression of students in CT, networking, and security-related assessments (RQ2)

One analysis technique applied here was linear regression as inspired by [Jose et al., 2016] and [Kim et al., 2016]. The x -axis represents the different assessments numbered as an integer-valued sequence starting with the pre-test given the value 0. The y -axis represents the scores on each assessment expressed as a percentage. The data points shown via the scatter plots correspond to the individual student percentages, with larger shapes indicating more students with the same performance. For the regression lines we used the average scores for each assessment. Finally, the R squared (R^2) value is also presented. The regression model is was designed to predict student performance on the post-test and evaluated based on the mean squared error (MSE) of the predicted value as compared to the actual performance of students.

To answer this question, we begin by tracing student performance on the main categories - CT, networking, and security - as well as across all categories. We use pre- and post-test assessments, formative assessments, and the final exam that was given to students at the end of the course. We begin by combining student scores at each of these time periods with linear regression to gain a better understanding of the relationship between learning and understanding at different points in the curriculum. Each assessment is treated as a distinct x value for the purposes of performing linear regression on the first $n - 1$ metrics, where n is the total number of different metrics. This includes any possible formative assessments on top of the always-present pre-test, class final, and post-test.

To give some context on the plots themselves, the matching triangular shape of the scatter plots for the pre- and post-tests indicates that these tests are identical. The matching circular shape for Modules 1 through 5 indicates that these all follow roughly the same pattern as described in Section 4.3.2. Module 5 was unique in that the length of the module necessitated additional assessments. The classroom final was comprehensive in nature. However, it was also distinct from the pre- and post-test given its emphasis on concepts being integrated within a NetsBlox programming context. Thus, it was given its own unique star shape. The coloring is also meaningful, with all blue markers corresponding with assessments that are incorporated into the linear regression model. The red markers for the post-test indicate that these values are only used to determine the previously mentioned MSE.

From initial observation of the category-based plots (Figure 5.1) and summary of statistics (Table 5.12), we can see that there was a slow but consistent rise in student performance for overall and CT scores. The regression lines have identical slopes ($slope = 0.03$) and nearly identical intercept values ($intercept = 0.58$ for overall, $intercept = 0.57$ for CT). This positive trend was also an accurate predictor of post-test performance in each of these cases, with very small MSE values calculated ($MSE = 0.04$ for overall, $MSE = 0.03$ for CT). It is worth noting that our predicted value did slightly overshoot the actual average performance for both of

these categories.

Networking performance showed a large positive trend ($slope = 0.1$). Though there was a significant increase from pre- to post-test as shown in Table 5.3, this led to a predicted post-test percentage of 90.35% compared to a true average post-test percentage of 56.57%. Only a few students exceeded the predicted score ($MSE = 0.17$). Though students did perform well during Module 5, they struggled on the in-class final. This led to a negative slope (-0.06) and ultimately a fairly low predicted result on the post-test. For the most part, students exceeded this expectation and did improve their post-test scores (Table 5.3). That led to an $MSE = 0.10$ due to the model predicting lower student performance (40.08%) than their actual performance (60%).

With this overall framing in place, we turn to the evaluation of specific areas from within our broader categories. They were selected because of their prevalence throughout the curriculum, as can be seen in Figure 5.2 and Table 5.10. They were in turn prevalent in the curriculum due to their inclusion in sources such as AP Computer Science Principles [K-12, 2020], the CSTA Standards [Seehorn and Clayborn, 2017], and the Tennessee CS Standards [of Education, 2018]. The primary requirement was for the construct to have been a part of the pre-post-tests, the classroom final, and at least one assessment from within one of the five modules. A mapping process with iterative refinements was done to properly associate learning and assessment opportunities in each area. This led to our selection of Variables, Conditionals, Loops, Encryption, Messaging, Topology, Evaluation, and Testing & Debugging as categories for further study.

CT concepts and strategies were covered earlier in the curriculum during introductory programming tasks. Therefore, we tend to have the most data points for these topics because they occur throughout the curriculum (Figure 5.2 and Tables 5.10 and 5.11). We see areas like Evaluation and Testing & Debugging in every module, while Variables and Loops do not occur in Module 1 but are featured in large quantities after that. Conditionals were similar, but less emphasized as compared to Variables and Loops. Encryption and Topology do not feature until our final module, matching up with the overall curricular implementation described in 4.1. Messaging was an exception due to a heavy focus on the NetsBlox chatting and messaging features in Module 4 as an introduction to networking.

We next highlight a few key points from Table 5.11 and Figure 5.3 related to assessment performance over time. Variables featured throughout the curriculum and showed mixed results. This is contrasted by the most comparable area (Loops), where students had significantly improved from pre- to post-test despite the roughly same trajectory overall. Students also improved significantly in the area of Evaluation from pre- to post-test, which largely seems to correspond with the results from within each module. Average percent scores improved from 47% on the pre-test, 58% during Module 1, 67% during Module 2, and then a large jump to 89% during Module 5. Students tended to perform better on the mid-module assessments and the

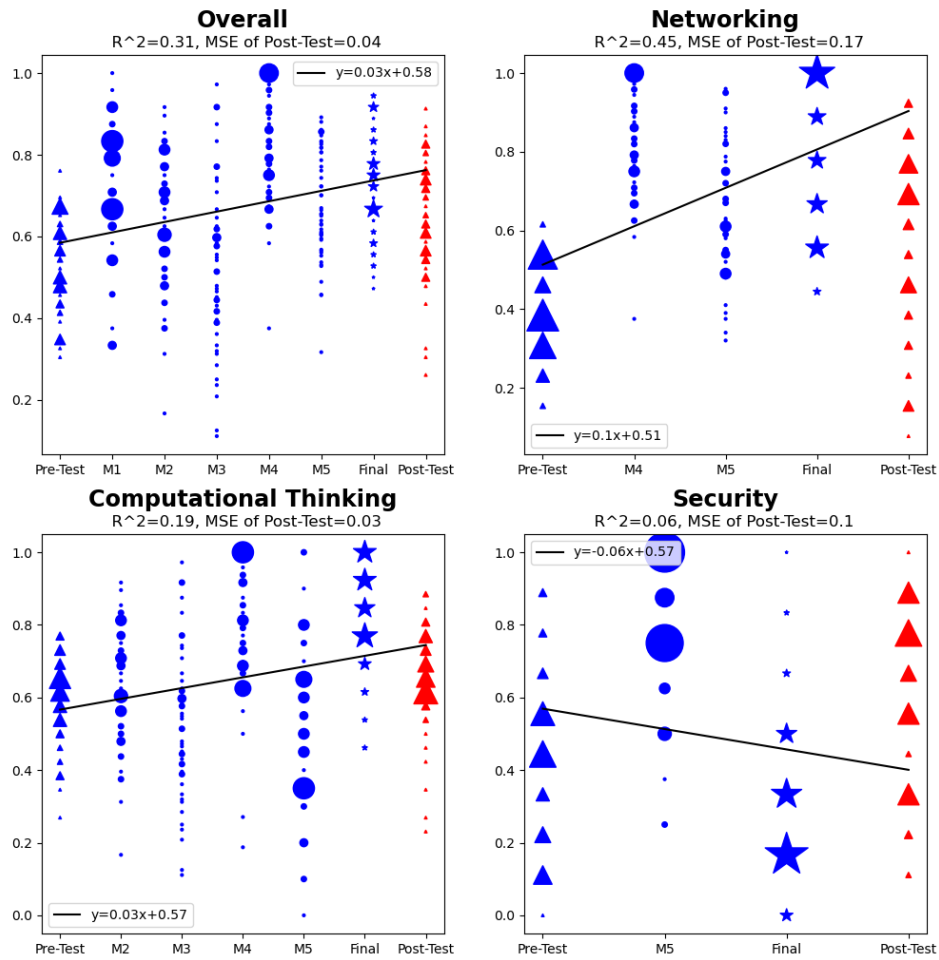


Figure 5.1: Starting from the top left and progressing in a clockwise order, these plots represent Overall, Networking, Security, and CT performance. Linear regression was applied to student performance on assessments at different points in the curriculum. M1, M2, etc. refer to Module 1, Module 2, etc. If a subject area was not present during assessments for a module, that module was not included in the progression. Increasing shape size corresponds with a larger number of students finishing with the corresponding percentage of points. All scores prior to the post-test were included in the regression to then predict post-test performance, as shown via the black line. Post-test scores are colored red to indicate this separation. Pre- and post-tests were identical and are shown as identical shapes. The classroom final, a distinct summative assessment, is given a distinct shape.

classroom final relative to the pre-post-test.

The next step was to take the previously mentioned constructs and perform linear regression analysis. Starting with CT and security concepts 5.4, a value of 0.41 was computed to be the y-intercept (b) for Variables and scores generally trended upward ($slope = 0.06$). Most of the variation over time was explained by

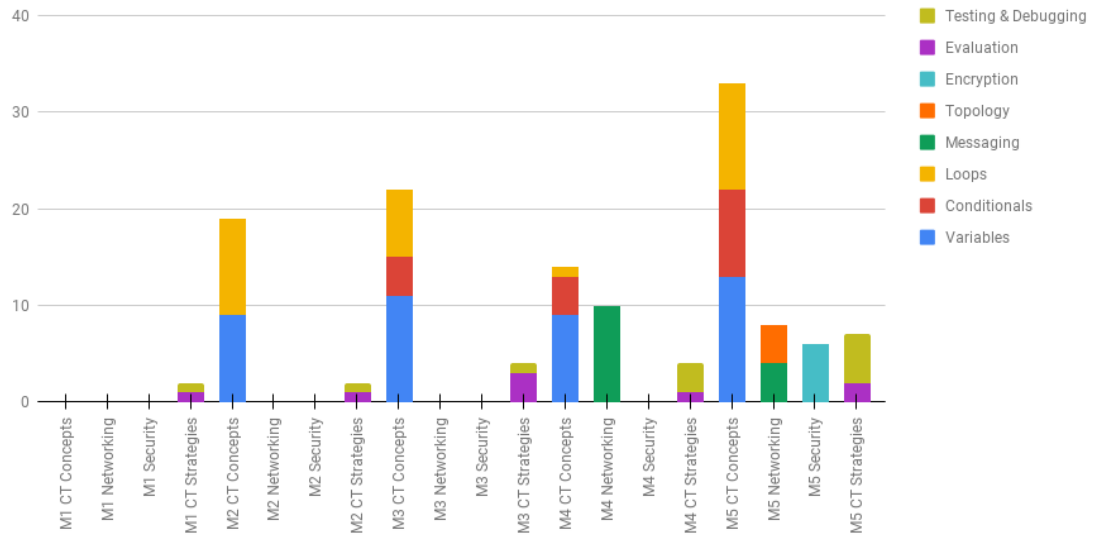


Figure 5.2: Number of subsections of a module that were related to each concept of interest. M1, M2, etc. refer to Module 1, Module 2, etc.

Table 5.10 Number of occurrences of key constructs throughout the various modules of the curriculum. Multiple constructs may occur within the same sub-module. Total number of sub-modules provided for reference

Construct	Module 1	Module 2	Module 3	Module 4	Module 5
Variables	0	9	11	9	13
Conditionals	0	0	4	4	9
Loops	0	10	7	1	11
Encryption	0	0	0	0	6
Messaging	0	0	0	10	4
Topology	0	0	0	0	4
Evaluation	1	1	3	1	2
Testing & debugging	1	1	1	3	5
Total	8	12	11	10	29

the model as indicated by the high R squared term ($R^2 = 0.66$). There was a relatively large error between predicted post-test scores and actual post-test scores ($MSE = 0.15$). Loops and Variables produced similar results, with a nearly identical equation of the line ($y = 0.07x + 0.42$). Fewer assessments led to a lower predicted post-test score, which was then more accurate when related to actual post-test scores ($MSE = 0.12$).

Conditionals and Encryption appeared in a smaller number of assessments in the intervention. In addition to the required pre-test, classroom final, and post-test, Conditionals were only featured during Module 3 and Encryption was only featured during Module 5. Despite this, the linear regression model for Conditionals tracked student performance fairly accurately over time. There was a relatively steady increase in students' performance and a relatively low error between the predicted and actual performance of students on the post-

Table 5.1.1: Assessment performance over time broken down by key constructs. Results are presented as “Mean (SD)”. “N/A” indicates that the concept was not present during that particular assessment. Scores were normalized to allow for easier comparisons across assessments.

Construct	Pre-Test	Module 1	Module 2	Module 3	Module 4	Module 5	Class Final	Post-Test
Variables	0.4 (0.17)	N/A (N/A)	0.51 (0.23)	0.53 (0.2)	0.66 (0.23)	0.51 (0.25)	0.8 (0.15)	0.43 (0.16)
Conditionals	0.7 (0.15)	N/A (N/A)	N/A (N/A)	0.62 (0.25)	N/A (N/A)	N/A (N/A)	0.87 (0.21)	0.71 (0.15)
Loops	0.35 (0.17)	N/A (N/A)	0.66 (0.18)	0.5 (0.23)	N/A (N/A)	0.51 (0.25)	0.78 (0.17)	0.54 (0.25)
Encryption	0.25 (0.32)	N/A (N/A)	N/A (N/A)	N/A (N/A)	N/A (N/A)	0.54 (0.39)	0.27 (0.24)	0.27 (0.29)
Messaging	0.38 (0.41)	N/A (N/A)	N/A (N/A)	N/A (N/A)	0.76 (0.16)	0.52 (0.5)	0.76 (0.2)	0.45 (0.42)
Topology	0.28 (0.2)	N/A (N/A)	N/A (N/A)	N/A (N/A)	N/A (N/A)	0.72 (0.25)	0.75 (0.3)	0.57 (0.34)
Evaluation	0.47 (0.34)	0.58 (0.49)	0.67 (0.19)	N/A (N/A)	N/A (N/A)	0.89 (0.23)	0.66 (0.23)	0.73 (0.32)
Testing & de-bugging	0.75 (0.43)	0.4 (0.38)	N/A (N/A)	N/A (N/A)	N/A (N/A)	N/A (N/A)	0.56 (0.5)	0.67 (0.47)
Total	0.53 (0.11)	0.71 (0.17)	0.63 (0.16)	0.53 (0.2)	0.81 (0.14)	0.68 (0.13)	0.73 (0.12)	0.64 (0.15)

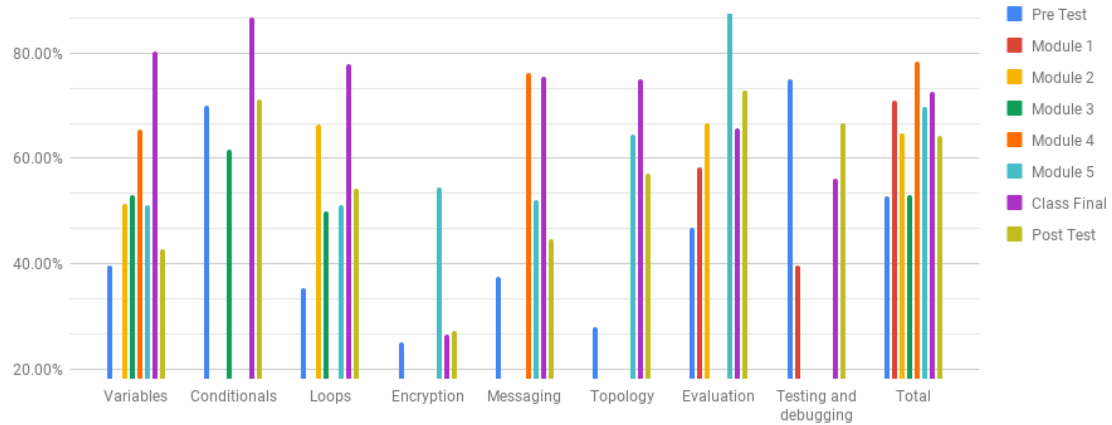


Figure 5.3: Performance on a concept during each module as related to maximum possible performance. Total is included for comparison

test ($MSE = 0.06$). Meanwhile, performance on the Encryption assessments was roughly stagnant throughout the curriculum. There was minimal improvement from pre- to post-test as shown in Table 5.11 and via the slope (0.01) of the regression line.

The networking and CT strategies plots are shown in Figure 5.5. The actual data from the Topology-related assessments could be largely explained by the model, as indicated by the highest R squared value ($R^2 = 0.8$). However, the large slope of the line of best fit ($slope = 0.24$) led to a prediction above 100% for the post-test with a high $MSE = 0.35$ when compared to actual results. The model for the Messaging assessments was not much more accurate ($MSE = 0.32$) despite an additional module worth of data and a much more restricted expectation of growth ($w = 0.09$). The application of linear regression to Testing and Debugging assessment data could correctly account for the downward trend in performance after beginning from a fairly high initial score ($b = 0.66$ along with an initial average percentage of 0.75 from Table 5.11). This is shown via the negative slope ($w = -0.09$). This model was still not particularly accurate for predicting post-test performance, with a $MSE = 0.3$ between the predicted value and the spread of actual percent-based post-test scores. The most informative of the plots from this section was focused on Evaluation of existing programs. Questions in this area were prevalent throughout the intervention, providing a reasonable spread of data points to consider. The linear regression model ($y = 0.07x + 0.52$) could explain much of the variation in scores over time ($R^2 = 0.49$) and fairly accurately predicted post-test scores as well ($MSE = 0.12$). A summary of the results from each of the presented linear regression plots are shown in Table 5.12.

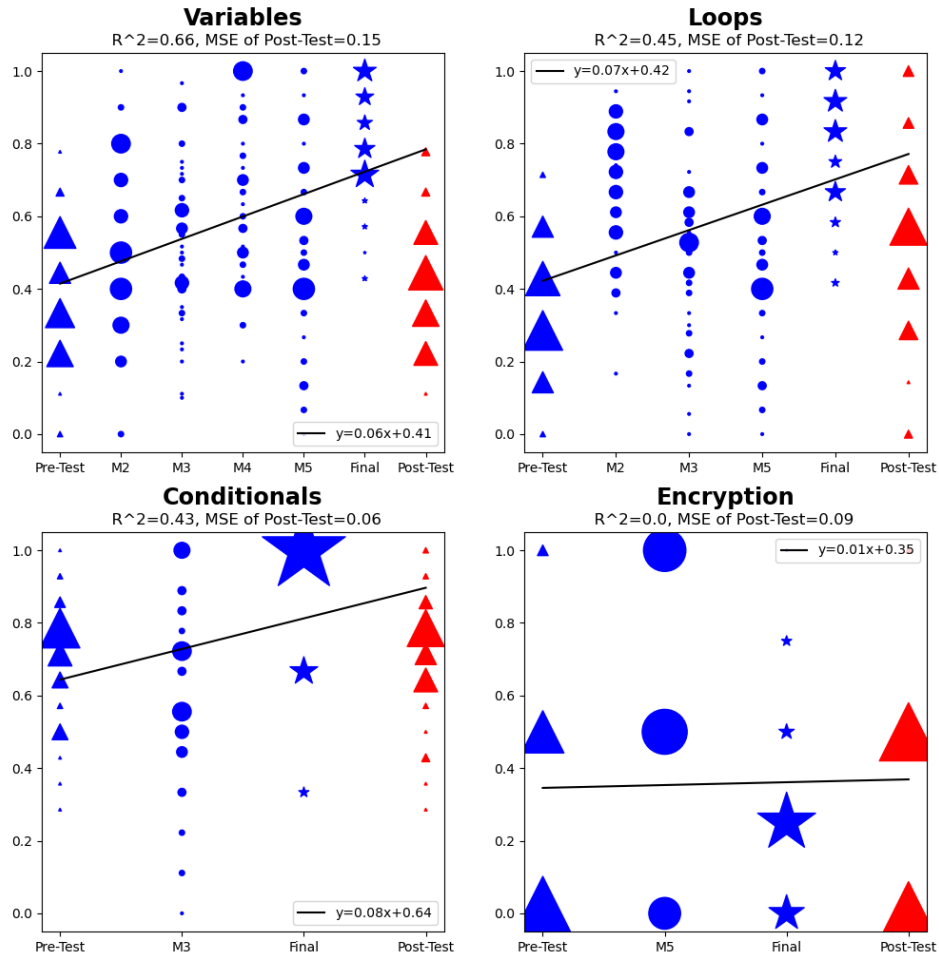


Figure 5.4: Starting from the top left and progressing in a clockwise order, these plots represent Variables, Loops, Encryption, and Conditionals performance. Linear regression was applied to student performance on assessments at different points in the curriculum. M1, M2, etc. refer to Module 1, Module 2, etc. If a subject area was not present during assessments for a module, that module was not included in the progression. Increasing shape size corresponds with a larger number of students finishing with the corresponding percentage of points. All scores prior to the post-test were included in the regression to then predict post-test performance, as shown via the black line. Post-test scores are colored red to indicate this separation. Pre- and post-tests were identical and are shown as identical shapes. The classroom final, a distinct summative assessment, is given a distinct shape.

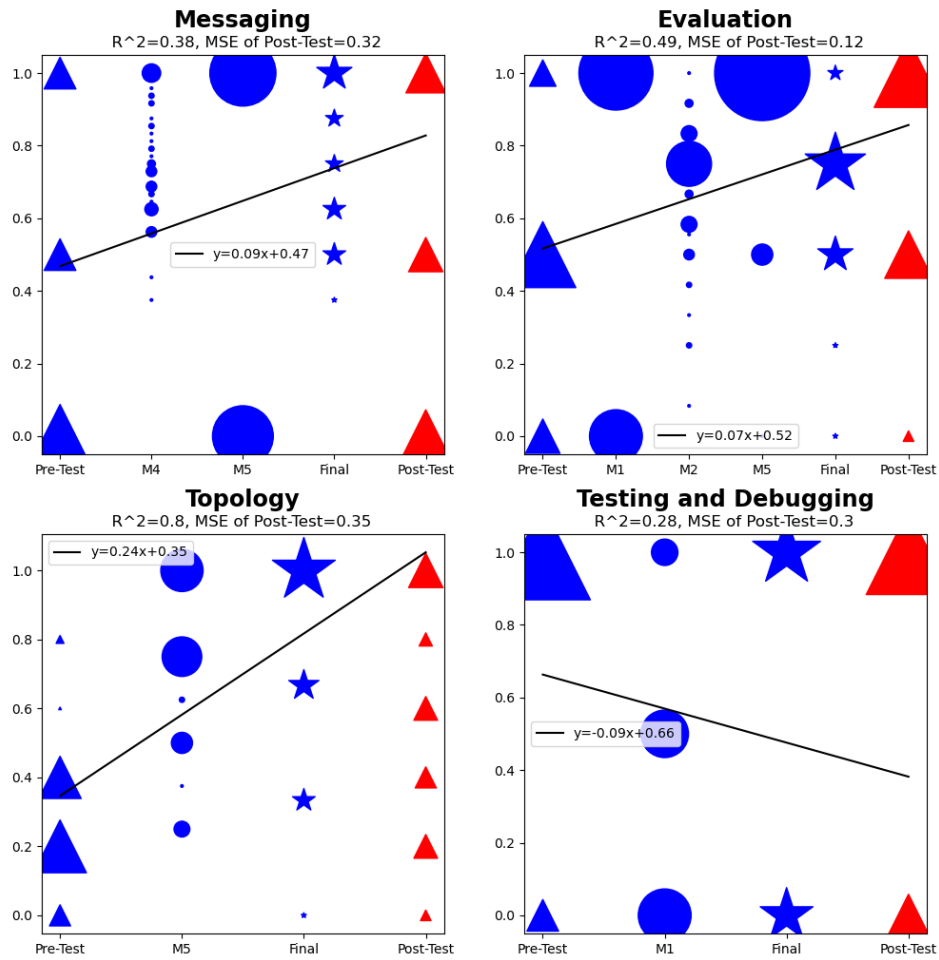


Figure 5.5: Starting from the top left and progressing in a clockwise order, these plots represent Messaging, Evaluation, Testing and Debugging, and Topology performance. Linear regression was applied to student performance on assessments at different points in the curriculum. M1, M2, etc. refer to Module 1, Module 2, etc. If a subject area was not present during assessments for a module, that module was not included in the progression. Increasing shape size corresponds with a larger number of students finishing with the corresponding percentage of points. All scores prior to the post-test were included in the regression to then predict post-test performance, as shown via the black line. Post-test scores are colored red to indicate this separation. Pre- and post-tests were identical and are shown as identical shapes. The classroom final, a distinct summative assessment, is given a distinct shape.

Table 5.12 Statistics related to each of the presented linear regression plots.

Category	R^2	MSE of Post-Test	slope (weight)	y-intercept (bias)
Overall	0.31	0.04	0.03	0.58
CT	0.19	0.03	0.03	0.57
Networking	0.45	0.17	0.1	0.51
Security	0.06	0.1	-0.06	0.57
Variables	0.66	0.15	0.06	0.41
Conditionals	0.43	0.06	0.08	0.64
Loops	0.45	0.12	0.07	0.42
Encryption	0	0.09	0.01	0.35
Messaging	0.38	0.32	0.09	0.47
Topology	0.8	0.35	0.24	0.35
Evaluation	0.49	0.12	0.07	0.52
Testing and Debugging	0.28	0.3	-0.09	0.66

We further wanted to investigate potential relationships between student performance on various assessments. Many correlations of moderate strength were found when comparing pre-test performance across the four broad categories to performance on module assessments, module project scores, the classroom final, and post-test scores (Table 5.13). Though this analysis is not predictive in nature, strong correlations provided some context for the impact of prior knowledge on future performance. Some of this information was included in Table 5.9 as part of our exploration of RQ1, but is repeated here for comparison. The strongest noted correlations are:

- Pre-Test Total to Post-Test Networking ($\rho = 0.62$), Post-Test Total ($\rho = 0.59$), and Module 4 Average ($\rho = 0.54$)
- Pre-Test CT to Post-Test Networking ($\rho = 0.55$)
- Pre-Test Security to Post-Test Networking ($\rho = 0.54$) and Post-Test Total ($\rho = 0.52$)

It is also worth mentioning some areas with only weak correlations. Pre-Test Networking scores seemed to have no relation with performance at any other point in the curriculum. Module 1 assessment scores and project score were only lowly correlated to any of the pre-test categories. One possible explanation of this (to be further expanded upon in Chapter 6) could be that there was little to do with CT, networking, or security during Module 1. It was instead an introduction to NetsBlox with minimal requisite knowledge of the aforementioned categories.

We additionally compare post-test performance across the four broad categories to performance on module assessments, module project scores, and the classroom final (Table 5.14). The intention is to contextualize student learning and better observe the relationship between performance throughout the curriculum to student knowledge after the intervention. The strongest noted correlations are:

- Post-Test Total to Module 5 Average ($\rho = 0.65$), Module 2, 3, and 4 Projects ($\rho = 0.51, 0.58, \text{and } 0.54$), and Class Final Total, CT, and Networking ($\rho = 0.56, 0.5, \text{and } 0.57$)
- Post-Test Networking to Module 5 Average ($\rho = 0.71$) and Class Final Total ($\rho = 0.54$)
- Post-Test Security to Module 5 Average ($\rho = 0.6$), Class Final Total ($\rho = 0.59$), and Class Final Networking ($\rho = 0.6$)

Other areas display only weak correlations. For example, Post-Test CT scores did not reach the $d \geq 0.4$ threshold in terms of correlation with any of the other assessment scores shown here. Another case was the Module 3 Average scores on the combination of the homework and quiz formative assessment.

Table 5.13 Spearman correlations are displayed as one measure of the relationship between a student’s pre-test performance and their performance throughout the remainder of the intervention. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate pre-test component, and each row corresponds to a separate assessment category from later in the intervention.

Metric	Pre-Test Total	Pre-Test CT	Pre-Test Networking	Pre-Test Security
Module 1 Average	0.35	0.22	0.29	0.27
Module 2 Average	0.41	0.28	0.16	0.42
Module 3 Average	0.34	0.23	0.08	0.4
Module 4 Average	0.54	0.43	0.35	0.4
Module 5 Average	0.44	0.21	0.29	0.49
Module 1 Project	0.33	0.19	0.23	0.29
Module 2 Project	0.49	0.4	0.39	0.32
Module 3 Project	0.42	0.42	0.27	0.26
Module 4 Project	0.3	0.29	0.25	0.1
Class Final Total	0.47	0.33	0.2	0.47
Class Final CT	0.37	0.29	0.16	0.36
Class Final Networking	0.36	0.22	0.1	0.43
Class Final Security	0.47	0.39	0.29	0.37
Post-Test Total	0.59	0.49	0.26	0.52
Post-Test CT	0.44	0.42	0.18	0.34
Post-Test Networking	0.62	0.55	0.23	0.54
Post-Test Security	0.37	0.14	0.26	0.46

Table 5.14 Spearman correlations are displayed as one measure of the relationship between a student’s post-test performance and their performance on other assessments during the intervention. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate post-test component, and each row corresponds to a separate assessment category from earlier in the intervention.

Metric	Post-Test Total	Post-Test CT	Post-Test Networking	Post-Test Security
Module 1 Average	0.33	0.34	0.18	0.4
Module 2 Average	0.46	0.31	0.38	0.46
Module 3 Average	0.38	0.37	0.26	0.25
Module 4 Average	0.42	0.34	0.32	0.4
Module 5 Average	0.65	0.31	0.71	0.6
Module 1 Project	0.37	0.16	0.45	0.47
Module 2 Project	0.51	0.39	0.45	0.43
Module 3 Project	0.58	0.39	0.37	0.48
Module 4 Project	0.54	0.28	0.45	0.41
Class Final Total	0.56	0.3	0.54	0.59
Class Final CT	0.5	0.35	0.43	0.48
Class Final Networking	0.57	0.37	0.41	0.6
Class Final Security	0.41	0.14	0.49	0.48

In Table 5.15, we present one view of the relationship between a student's performance on two of the early programming projects and their performance on advanced networking and security content later in the curriculum. Module 2 project scores were moderately correlated with almost all other scores of interest from the rest of the intervention. This project heavily focused on the custom block feature and concepts of loops and variables. Mastering those skills and persisting in accomplishing most or all of the objectives of the project appeared to set the student up to continue engaging with the curriculum and to perform well on future projects. Meanwhile, the only moderate correlation involving Module 3 is with the Module 4 project ($\rho = 0.65$).

Another perspective on the relationship of project performance to other areas is shown in Table 5.16. We applied our curricular mapping scheme to each of the main projects from Modules 2, 3, and 4. By doing so, we identified the specific constructs from our core set (as shown in Table 5.10) that were integrated within the project. Then, we split student performance on the pre-post test, class final, and module-based formative assessments into those that were completed before the project and those that were completed after. One intention was to observe whether performance before a project was correlated with performance on the project. The other view was whether understanding obtained during a project could lead to a trend towards higher scores after the project.

The module 2 project emphasized only variables and loops from our primary set of constructs. For variables, both pre-project ($\rho = 0.54$) and post-project ($\rho = 0.6$) performance are moderately correlated with project performance. The situation is similar for loops - ($\rho = 0.49$) and ($\rho = 0.56$) respectively. Project 3 featured more core constructs, with variables, conditionals, loops, and evaluation all featured. Correlation improved in terms of the relationship between pre-project variable performances ($\rho = 0.43$) and post-project variable performance ($\rho = 0.53$). On the other hand, conditionals and evaluation demonstrated a moderate positive correlation before the project ($\rho = 0.51$ for each) that disappeared after the project ($\rho = 0.37$ and $\rho = 0.22$ respectively). The Module 4 project then had no particularly strong indicators one way or the other for the main categories of variables, conditionals, and messaging.

Another sub-area of interest was whether there were any relationships between student performance and student attitudes or engagement. From Table 5.17, we can see that there was a moderate correlation between pre-test total performance and quantity of actions during Module 3 ($\rho = 0.56$), Module 5 ($\rho = 0.53$), and Total Actions ($\rho = 0.5$). On the attitudinal survey side, only the General Behavioral Engagement (BE) score on the post-survey when related to Pre-Test Total ($\rho = 0.46$) and Pre-Test CT ($\rho = 0.43$) could approach the threshold of interest.

The other progression to consider involved starting from Pre-Survey results before correlating to assessments and other scores from throughout the intervention as well as Post-Survey results. On the performance

Table 5.15: Spearman correlations are displayed as one measure of the relationship between a student's early CT performance and their performance and participation during networking and security modules and assessments. Each row corresponds to a separate CT-related assessment, and each column corresponds to a metric from later in the intervention. This early CT performance is based on the Module 2 and 3 project submission scores. Positive correlations ≥ 0.4 are in bold and blue text.

Metric	Module 4 Actions	Module 4 Networking	Module 4 Project	Module 4 Actions	Module 5 Networking	Module 5 Security	Class Final Networking	Class Final Security	Post-Test Networking	Post-Test Security
Module Project 2	0.44	0.54	0.48	0.62	0.56	0.13	0.41	0.46	0.45	0.43
Module Project 3	0.23	0.32	0.65	0.34	0.35	0.09	0.39	0.29	0.37	0.48

Table 5.16 Spearman correlations are displayed as one measure of the relationship between project performance and student performance on related topics both before and after each respective project. Different projects have different associated concepts. M2, M3, M4 are short for Module 2, Module 3, Module 4. Dashes indicate that scores were not relevant for that particular project. Each column corresponds to a separate project, and each row corresponds to a related metric compiled from assessments before or after that project for “Through” and “After” respectively. Positive correlations ≥ 0.4 are in bold and blue text.

Concept	Module 2 Project	Module 3 Project	Module 4 Project
Variables Through M2	0.54	-	-
Variables After M2	0.6	-	-
Loops Through M2	0.49	-	-
Loops After M2	0.56	-	-
Variables Through M3	-	0.43	-
Variables After M3	-	0.53	-
Conditionals Through M3	-	0.51	-
Conditionals After M3	-	0.37	-
Loops Through M3	-	0.54	-
Loops After M3	-	0.55	-
Evaluation Through M3	-	0.51	-
Evaluation After M3	-	0.22	-
Variables Through M4	-	-	0.36
Variables After M4	-	-	0.43
Conditionals Through M4	-	-	0.35
Conditionals After M4	-	-	0.18
Messaging Through M4	-	-	0.3
Messaging After M4	-	-	0.32

side (Table 5.18), Specific Mastery Approach (MA) - Pre showed a low-to-moderate correlation with Module 1 Project scores ($\rho = 0.45$). As soon as students reached the more advanced CT and networking content on the remaining projects, this relationship largely disappeared. We calculated similar correlations for either Perceived Competence (PC) or General BE when associated with Post-Test Total ($\rho = 0.45$; PC), Post-Test CT ($\rho = 0.47$; PC), and Post-Test Networking ($\rho = 0.47$; BE).

Meanwhile, on the attitudes and actions side (Table 5.19), some results such as General BE - Pre to General BE - Post ($\rho = 0.61$) and Specific AV - Pre to Specific AV - Post ($\rho = 0.53$) were expected. Others such as Specific MA - Pre to Module 3 Actions ($\rho = 0.44$) and Specific Computing Interest (CI) - Pre to Specific TV - Post ($\rho = 0.59$) make sense in the context of past work (such as [Linnenbrink-Garcia et al., 2018]). High Mastery Approach students were more likely to interact with the programming environment at a higher level and students with a high Computing Interest were more likely to value related tasks. Other relationships meeting our threshold were atypical connections. General BE - Pre correlating to Specific Performance-Approach (AP) and Performance-Avoidance (AV) on the post-survey ($\rho = 0.49$ in each case) could indicate that our engaged students were only engaged in the interest of demonstrating competence to others or avoiding feelings of incompetence.

Table 5.17 Spearman correlations are displayed as one measure of the relationship between a student’s pre-test performance and either attitudes (see Table 5.6) or actions. “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate pre-test component, and each row corresponds to a metric from later in the intervention. Additional survey categories can be found in Appendix D

Metric	Pre-Test Total	Pre-Test CT	Pre-Test Net- working	Pre-Test Security
Module 1 Actions	0.12	0.09	0.16	-0.01
Module 2 Actions	0.18	0.06	0.3	0.13
Module 3 Actions	0.56	0.44	0.43	0.42
Module 4 Actions	0.34	0.32	0.18	0.23
Module 5 Actions	0.53	0.44	0.34	0.39
Total Actions	0.5	0.37	0.4	0.38
General BE - Post	0.46	0.43	0.23	0.32
Specific BE - Post	0.14	0.04	0.34	0.05

Table 5.18 Spearman correlations are displayed as one measure of the relationship between a student’s pre-survey performance (see Table 5.6) and their performance throughout the remainder of the intervention. Only selected survey categories with at least one relevant correlation $\geq |0.4|$ are presented for the sake of space. “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. Positive correlations ≥ 0.4 are in bold and blue text. Each column corresponds to a separate pre-survey category, and each row corresponds to a metric from later in the intervention. Further correlations are included in Appendix D

Metric	Specific MA - Pre	Specific PC - Pre	General BE - Pre
Module 1 Project	0.45	0.32	0.32
Module 2 Project	0.16	0.37	0.07
Module 3 Project	0.11	0.29	-0.11
Module 4 Project	0.04	0.2	0.19
Post-Test Total	0.1	0.45	0.29
Post-Test CT	0.07	0.47	0.11
Post-Test Networking	0.31	0.32	0.47
Post-Test Security	0.08	0.36	0.17

Table 5.19 Spearman correlations are displayed as one measure of the relationship between a student’s pre-survey performance and their attitudes (see Table 5.6) and actions. Only selected survey categories with at least one relevant correlation $\geq |0.4|$ are presented for the sake of space. “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. Each column corresponds to a separate pre-survey category, and each row corresponds to a metric from later in the intervention. Positive correlations ≥ 0.4 are in bold and blue text.

Metric	Specific MA - Pre	General BE - Pre	General AV - Pre	Specific AV - Pre	Specific CI - Pre	Specific CC - Pre
Module 1 Actions	0.12	0.26	0.3	0.26	0.23	0.05
Module 2 Actions	0.26	0.17	0.15	0.17	0.23	0.1
Module 3 Actions	0.44	0.3	0.12	0.01	0.18	-0.07
Module 4 Actions	0.32	0.13	0.14	0.01	0.21	0.2
Module 5 Actions	0.26	0.29	0.11	-0.12	0.29	0.15
Total Actions	0.31	0.31	0.15	0	0.3	0.1
General TV - Post	0.09	0.06	0.01	-0.03	0.13	0.2
Specific TV - Post	0.35	0.08	0.24	0.22	0.59	0.42
General MA - Post	0.34	0.24	0.02	0.12	0.41	0.28
Specific MA - Post	0.47	0.22	0.18	0.17	0.37	0.2
General AP - Post	0.16	0.34	0.44	0.36	0.25	0.24
Specific AP - Post	0.13	0.49	0.5	0.56	0.18	0.07
General BE - Post	0.35	0.61	0.35	0.14	0.03	0.21
Specific BE - Post	0.24	0.22	0.08	0.02	0.11	0.25
General AV - Post	0.17	0.42	0.34	0.22	-0.04	0
Specific AV - Post	0.11	0.49	0.57	0.53	0.13	0.05
Specific CI - Post	0.19	0.24	0.36	0.34	0.62	0.47

5.3 Strategy progression (RQ3)

One key component of this section is the Markov chain (MC) models, which are presented for each of the four primary projects. These MC models [Russell and Norvig, 2002] were generated from students' sequences of programming actions. Actions were obtained and categorized using the procedure laid out in Section 4.4. We represented each state in the state space by an action category (Section 4.4.3), which occurred at time t . There is then an associated transition probability indicating the likelihood that another specific action would occur at time $t + 1$. Increasing thickness of the ring around each state corresponds with increasing quantity of that type of action, which are listed in Table 5.20.

A line with an arrow at the end represents a significant (≥ 0.1) transition probability from one action type to another. This can also occur for self-transitions, with the line and arrow simply looping back into the initial state. It should be noted that these are specifically representative of the conditional probability of each transition, also known as the first-order Markov transition probability. Models were generated using GraphViz software [Gansner and North, 2000]. Similar previous work can be found in [Zhang, 2020]. Supplemental information is shown via tables to include the less common transitions. This information can still be informative for determining any desirable sequences that are underutilized or any undesirable sequences that students are successfully avoiding.

To supplement the Markov chains and Differential Sequence Mining (DSM; to be discussed later), we present the counts and percentages for specific key action categories for each of the four evaluated projects (Table 5.20). The most common actions from the first project were StartAll (24%), FillField (16%), and both MoveBlock and ChangeView (13% each). Students were not adding as many blocks as expected for this project, with AddBlock actions only making up 5% of all actions. StartAll and ChangeView were related to the game project in particular, as StartAll was needed to start the game and different content was intentionally stored on different sprites. However, the scripts stored on different sprites were not directly related to the task students were completing. Creativity, though still at a fairly small 3%, was more prominent in this project than any other. There were more opportunities to customize the various sprites used in the game and the background stage than for other projects.

Module 2 features almost entirely MoveBlock (22%), FillField (20%), and StartAll (20%) actions. Students were primarily building one long script to satisfy the art-based requirements, so all of these make sense in that context. In particular, MoveBlock and FillField represent model-building, Solution Construction actions. MoveBlock would correspond to attaching new blocks to the end of the bottom of the main script or inserting a block within an existing block. FillField would indicate selecting values from a drop-down menu (such as changing the number of degrees to turn) or typing values into blocks (such as controlling the length

Table 5.20 Counts (percentages) for each action category during each of the four evaluated projects.

Category	Module 1	Module 2	Module 3	Module 4
AddBlock	260 (0.05)	323 (0.03)	524 (0.04)	53 (0.02)
Batch	0 (0)	50 (< 0.01)	204 (0.02)	225 (0.06)
ChangeView	690 (0.13)	283 (0.03)	1142 (0.09)	32 (0.01)
ControlExecution	272 (0.05)	184 (0.02)	236 (0.02)	86 (0.02)
CreateContent	2 (< 0.01)	82 (0.01)	296 (0.02)	37 (0.01)
Creativity	172 (0.03)	82 (0.01)	89 (0.01)	1 (< 0.01)
FillField	855 (0.16)	2254 (0.20)	2018 (0.16)	221 (0.06)
ManipulateList	0 (0)	0 (0)	915 (0.07)	30 (0.01)
MoveBlock	694 (0.13)	2387 (0.22)	3085 (0.24)	1067 (0.30)
OrganizeBlock	431 (0.08)	1241 (0.11)	1391 (0.11)	738 (0.21)
RemoveBlock	198 (0.04)	654 (0.06)	618 (0.05)	173 (0.05)
StartAll	1321 (0.24)	2248 (0.20)	1144 (0.09)	709 (0.20)
StartScript	524 (0.10)	1268 (0.11)	1353 (0.10)	151 (0.04)

of each side of a shape). StartAll would then be the primary Solution Assessment action for this long singular script, though StartScript (11%) was also more common here than for other projects. It occurred when students clicked the script itself instead of the green flag button.

Module 3 was the only project with a focus on lists, leading to the highest portion of ManipulateList (7%) actions. Students returned to the game project, but with the new context of including a high score list and a musical component. Therefore, the return of ChangeView actions (9%) is not surprising. Otherwise, relative action quantities were similar to Module 2, with an emphasis on FillField and MoveBlock as opposed to AddBlock for Solution Construction purposes. StartScript (10%) surpassed StartAll (9%) as the primary Solution Assessment tool.

The project for module 4 required additional scaffolding. Components of the program were pre-built for students, and many of the blocks they would need were already included in the template project. This led to a change in tactics for students. In particular, Solution Independent actions like OrganizeBlock (21%) and Batch (6%) likely corresponded with attempts to undo ineffective code modifications. MoveBlock (30%) and StartAll (20%) were still highly featured as well, similarly to previous projects. On the other hand, FillField actions were only minimally required and therefore only minimally represented (6%) relative to previous projects.

With the previous context in mind, we turn to viewing and analyzing the transitions between these actions for each of the four projects. In this way, we are better able to frame the order in which students are performing actions. Looking at the MC model (Figure 5.6) and transition table (Table 5.21) for the Module 1 project, there are some patterns related to successful strategy implementation. This includes transitions from FillField to StartAll (29%) and RemoveBlock to StartAll (21%). However, other Solution Construction actions rarely led to immediate Solution Assessment, such as from MoveBlock to StartAll (15%) or AddBlock to StartScript (7%). These types of sequences amongst novice programmers can indicate a lack of testing the model, which can lead to a need for large changes at later points in the process.

Other behaviors may appear negative at first glance, but are largely explainable by the specific project context. For example, there is an extremely high transition probability (52%) for a ChangeView action to immediately follow a ChangeView action. This may seem like an inefficient use of time on Solution Independent actions. With the game project in particular, these actions likely indicated students exploring the project to better understand the pre-created behaviors of the adversary sprite and the objective sprite. They then may have switched back to the main user sprite to modify its behavior directly. In this scenario, implementing Solution Construction actions on the additional sprites would have been unnecessary. Repeated StartAll (52% self-transition probability) actions are also more acceptable for this project than in many other scenarios. This follows from the green flag serving as the means of starting the game. Ideally, students would only repeat this action multiple times after finishing the primary objective. However, exploring the game was a main objective of this Module, so this behavior may have played a valuable role in preparing students for future Modules.

Our next MC model (Figure 5.7) and corresponding transition probability table (Table 5.22) describe student action sequences during the second project. This project was outlined in Section 4.1 with examples shown in Appendix C.1. To summarize, students were expected to create one long script that accomplished a few different sprite drawing objectives using loops, variables, and a custom block. We again highlight some transitions that are appropriate for this specific project. When working towards one of the objectives, performing Solution Assessment actions would often not provide useful information to students. Therefore, longer sequences of Solution Construction actions would not indicate poor programming behaviors, but could instead be construed as students adapting to the problem at hand. Common examples of this include AddBlock to MoveBlock (34%), MoveBlock back to MoveBlock (37%), and FillField back to FillField (31%). Another potentially valuable combination is alternating Solution Assessment and Solution Construction actions. StartAll and MoveBlock, StartAll and FillField, and StartScript and FillField all met our threshold of > 10% transition probability in both directions.

Another context-sensitive transition is the 39% likelihood of a Creativity to StartAll sequence. In many

Table 5.21: Transition probabilities of students' NetsBlox actions during the Module 1 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.

Category	AddBlock	ChangeView	ControlExecution	CreateContent	Creativity	FillField	MoveBlock	OrganizeBlock	RemoveBlock	StartAll	StartScript
AddBlock	0.1846	0.0231	0	0	0.0038	0.2577	0.2692	0.1231	0.0654	0.0077	0.0654
ChangeView	0.0362	0.5174	0.0145	0	0.1522	0.0667	0.0304	0.0333	0.0072	0.1072	0.0261
ControlExecution	0.0772	0.1176	0.2463	0	0.0074	0.1213	0.0662	0.0368	0.0147	0.2574	0.0404
CreateContent	0.5	0	0	0	0	0	0	0	0	0.5	0
Creativity	0	0.5	0.0116	0	0.2674	0	0	0	0	0.2151	0.0058
FillField	0.0526	0.0327	0.007	0	0	0.3006	0.1392	0.0608	0.0199	0.2947	0.0912
MoveBlock	0.0533	0.0274	0.0043	0.0014	0.0014	0.2565	0.2896	0.0994	0.0519	0.1484	0.0663
OrganizeBlock	0.0302	0.0116	0	0.0023	0.0023	0.0951	0.2645	0.3387	0.0951	0.0626	0.0905
RemoveBlock	0.096	0.0505	0.0152	0	0	0.0404	0.298	0.0556	0.1919	0.2071	0.0404
StartAll	0.025	0.0977	0.131	0	0.0106	0.1014	0.0386	0.0386	0.0167	0.5163	0.0144
StartScript	0.0115	0.0172	0.0153	0	0.0038	0.1737	0.0763	0.0706	0.0344	0.0515	0.5458

circumstances, this would be considered an ineffective transition from a student modifying non-programming features of the project and immediately testing to view the impact of the changes. For this project, though Creativity actions are considered Solution Independent, they were required to demonstrate student understanding of aspects such as changing the color of a shape each time through a loop. Not all transitions can be construed as positive even with the relevant context. Common self-transitions for Batch, StartAll, and StartScript actions are indicative of repeated large-scale changes or repeated Solution Assessment with no code modifications in between. Both of these strategies are not conducive to completing this project or to successfully programming in general.

The Module 3 project required students to revisit the game project and include new features such as a high score list and musical components. ChangeView was once again a necessary component of this process for viewing scripts that were spread across multiple sprites, so the 61% self-transition likelihood (Figure 5.8; Table 5.23) is not an immediate cause for concern. However, the complete lack of > 10% transitions to other actions indicates that students may not have been using this feature effectively. Instead of switching views, observing code, switching back, and making changes, it was more likely students were just repeatedly changing views in a manner that was truly Solution Independent. The lack of > 10% transitions from StartAll to any of our Solution Construction actions and the high self-transitions of both StartAll and StartScript (47% in each case) also indicate repeated Solution Assessment with no code modifications in the middle. However, as previously mentioned with Module 1, this is somewhat necessitated by the setup of the game.

As with Module 2, repeated Solution Construction actions were often required to reach a specific objective before testing features of the code. High probabilities of self-transitions for MoveBlock (49%), FillField (55%), and even RemoveBlock (41%) could therefore be considered potentially valuable for attempting to reach those checkpoints or remedy observed incorrect results. This is also true for repeated combinations such as OrganizeBlock to MoveBlock (30%) and AddBlock to MoveBlock (40%). A similarly positive trend is related to CreateContent, which was needed for this project to initialize variable names. Multiple actions in this area were often repeated (47% self-transition). After one or more repetitions, the next step was often to insert new creations into the code via MoveBlock (25%) or FillField (12%) actions.

We conclude this portion of the analysis by observing sequences of behaviors during the Module 4 project. Context is once again important here, as students were expected to spend more time modifying existing complex Messaging scripts rather than truly creating programs from scratch. This led to the prevalence of MoveBlock and OrganizeBlock actions as a conduit for students' Solution Construction. These action types were a highly probable destination for almost all other actions as well. Common sequences included repeated MoveBlock actions (44%), repeated OrganizeBlock actions (35%), or combinations of the two (25% from MoveBlock to OrganizeBlock, 39% from OrganizeBlock to MoveBlock). Creativity and ChangeView

Table 5.22: Transition probabilities of students' NetsBlox actions during the Module 2 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.

Category	AddBlk	Batch	ChangeView	ControlEx	CreateContent	Creativity	FillField	MoveBlk	OrganizeBlk	RemoveBlk	StartAll	StartScript
AddBlk	0.0836	0	0.0433	0	0.0062	0.0093	0.1579	0.3375	0.0991	0.1455	0.0155	0.1022
Batch	0.02	0.5	0	0	0	0	0.16	0.16	0.12	0	0.04	0
ChangeView	0.0495	0	0.5548	0.0035	0	0.0318	0.0353	0.0389	0.106	0.0318	0.0777	0.0707
ControlEx	0.0326	0	0.0109	0.2989	0	0	0.1359	0.0598	0.0978	0.0217	0.2554	0.087
CreateContent	0.2439	0	0	0	0.1463	0	0.0976	0.4512	0.0122	0	0	0.0488
Creativity	0.0122	0	0.0732	0	0	0.0976	0.0854	0.2195	0.0366	0	0.3902	0.0732
FillField	0.0151	0.004	0.0027	0.0022	0.0084	0.0004	0.3128	0.165	0.0439	0.0146	0.3004	0.1295
MoveBlk	0.0318	0.0038	0.0034	0.0008	0.0088	0.0138	0.2153	0.3749	0.1072	0.0348	0.1483	0.0566
OrganizeBlk	0.025	0.0016	0.0089	0.0024	0.0032	0	0.0653	0.3441	0.2973	0.1064	0.0717	0.0733
RemoveBlk	0.0596	0.0031	0.0291	0.0015	0.0229	0	0.0642	0.2125	0.0917	0.3792	0.0826	0.0505
StartAll	0.0205	0.0009	0.0156	0.044	0.0027	0.0076	0.222	0.1103	0.1023	0.0227	0.3981	0.0476
StartScript	0.0174	0.0008	0.0189	0.0142	0.0024	0.0087	0.239	0.0875	0.1025	0.0363	0.0497	0.4164

Table 5.23: Transition probabilities of students' NetsBlox actions during the Module 3 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.

Category	AddBlk	Batch	Change View	ControlEx	CreateContent	Creativity	FillField	ManipList	MoveBlk	OrganizeBlk	RemoveBlk	StartAll	StartScript
AddBlk	0.1508	0	0.0324	0	0.0363	0	0.0821	0.0153	0.4008	0.0973	0.1088	0.0038	0.0706
Batch	0.0196	0.598	0.0098	0	0	0	0.098	0	0.1569	0.0882	0.0196	0	0.0049
Change View	0.063	0.0026	0.6077	0.0079	0.0105	0.0324	0.021	0.0018	0.0508	0.0788	0.021	0.0595	0.0403
ControlEx	0.0212	0	0.1398	0.3432	0.0042	0	0.0466	0.0127	0.0636	0.1186	0.0169	0.1907	0.0381
CreateContent	0.0743	0	0.0169	0	0.4662	0	0.125	0	0.2466	0.0203	0	0.0135	0.0236
Creativity	0	0	0.4157	0	0	0.5056	0	0	0	0	0	0.0674	0
FillField	0.0183	0.0099	0.0134	0.0015	0.005	0.0005	0.5451	0.0431	0.167	0.0327	0.0104	0.0649	0.0872
ManipList	0.0109	0	0.0022	0	0.0109	0	0.1399	0.6077	0.129	0.0164	0.0022	0.0098	0.0699
MoveBlk	0.0441	0.0081	0.012	0.0029	0.0188	0.0006	0.1112	0.0444	0.4901	0.1222	0.0301	0.0535	0.0613
OrganizeBlk	0.0489	0.0144	0.0446	0.0029	0.0122	0	0.0295	0.0223	0.2955	0.335	0.0762	0.0424	0.0748
RemoveBlk	0.0663	0.0162	0.0615	0.0049	0.0162	0	0.0469	0.0049	0.1958	0.0939	0.4061	0.0453	0.0388
StartAll	0.0166	0.0017	0.1259	0.0988	0.007	0.0026	0.0629	0.0096	0.0516	0.0708	0.0149	0.4738	0.0577
StartScript	0.0229	0.0015	0.0259	0.0103	0.0096	0.0007	0.1256	0.0569	0.102	0.0976	0.0288	0.0517	0.4656

actions are fairly rare as they were not necessary for this project, but there are still undesirable disconnected states for these Solution Independent actions. Frequent and repetitive StartAll actions (64% self-transition likelihood, 20% of all actions for this project) are also likely indicative of poor programming practice. Unlike for the game-focused projects, this chat server application (Appendix C.3) should not have required multiple consecutive executions of the program to observe the impact.

Repeated Batch actions (70% self-transition) indicate a higher than expected likelihood of multiple undo actions at once. This further implies a lack of systematic testing of performance of the program. Students were instead likely getting to a point where the outcome was not as expected before opting to reset the project back to an earlier state and try again. This is more applicable to this project than it would be otherwise, since many required blocks were provided to students. However, it does not indicate a productive strategy implementation. Neither do similar sequences such as repeated RemoveBlock (35%) actions.

Table 5.24: Transition probabilities of students' NetsBlox actions during the Module 4 project. Rows correspond with the initial action, and columns indicate the next action immediately following that initial action.

Category	AddBlk	Batch	Change View	ControlEx	CreateContent	Creativity	FillField	ManipList	MoveBlk	OrganizeBlk	RemoveBlk	StartAll	StartScript
AddBlk	0.0377	0	0	0.0189	0.0189	0	0.1887	0.0189	0.4906	0.1321	0.0566	0	0.0377
Batch	0	0.7022	0	0	0	0	0.04	0	0.1822	0.0711	0	0.0044	0
Change View	0	0	0.7188	0	0	0.0313	0	0	0.0625	0.0313	0	0.0938	0.0313
ControlEx	0	0	0	0.5116	0.0116	0	0.0116	0	0.1047	0.1395	0	0.2093	0.0116
CreateContent	0.0541	0	0	0	0.1081	0	0.2162	0	0.4865	0.027	0.027	0.0811	0
Creativity	0	0	1	0	0	0	0	0	0	0	0	0	0
FillField	0.0181	0.0407	0	0.0045	0.0136	0	0.3394	0.009	0.3213	0.1176	0.0317	0.0498	0.0498
ManipList	0.0333	0.0333	0	0	0.0333	0	0.1	0.2667	0.1333	0.1667	0.0667	0.0333	0.1
MoveBlk	0.0169	0.03	0.0009	0.0037	0.0122	0	0.0609	0.0084	0.4442	0.2493	0.045	0.1115	0.0141
OrganizeBlk	0.0122	0.0257	0.0014	0.0095	0.0081	0	0.0366	0.0068	0.3889	0.3455	0.0366	0.0799	0.0434
RemoveBlk	0.0347	0.0058	0.0116	0.0116	0.0116	0	0.0289	0	0.2428	0.1329	0.3526	0.1098	0.0405
StartAll	0.0085	0.0071	0.0028	0.0381	0.0085	0	0.0155	0	0.079	0.1368	0.0197	0.6446	0.0197
StartScript	0.0265	0	0.0066	0	0	0	0.0464	0.0331	0.1921	0.1656	0.053	0.053	0.4172

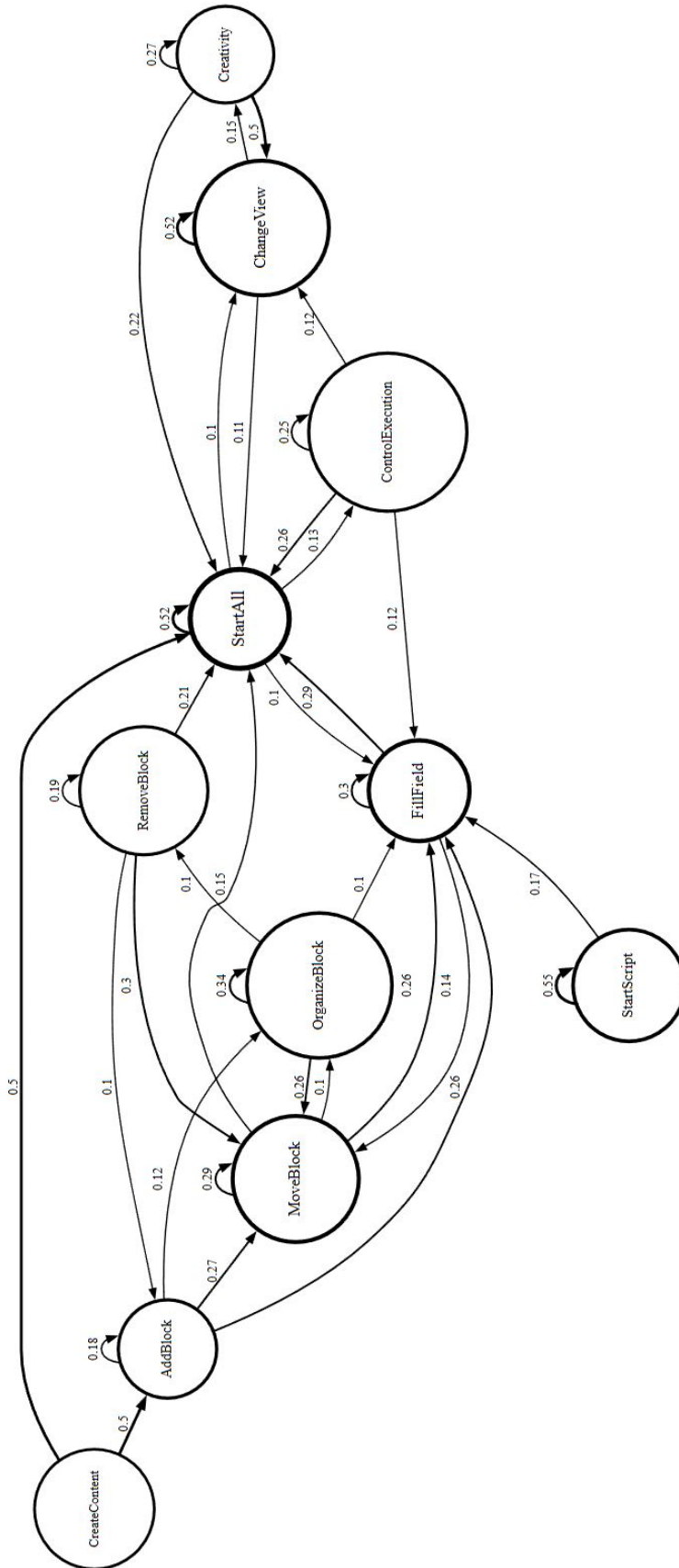


Figure 5.6: Markov chain model of students' NetsBlox actions during the Module 1 project.

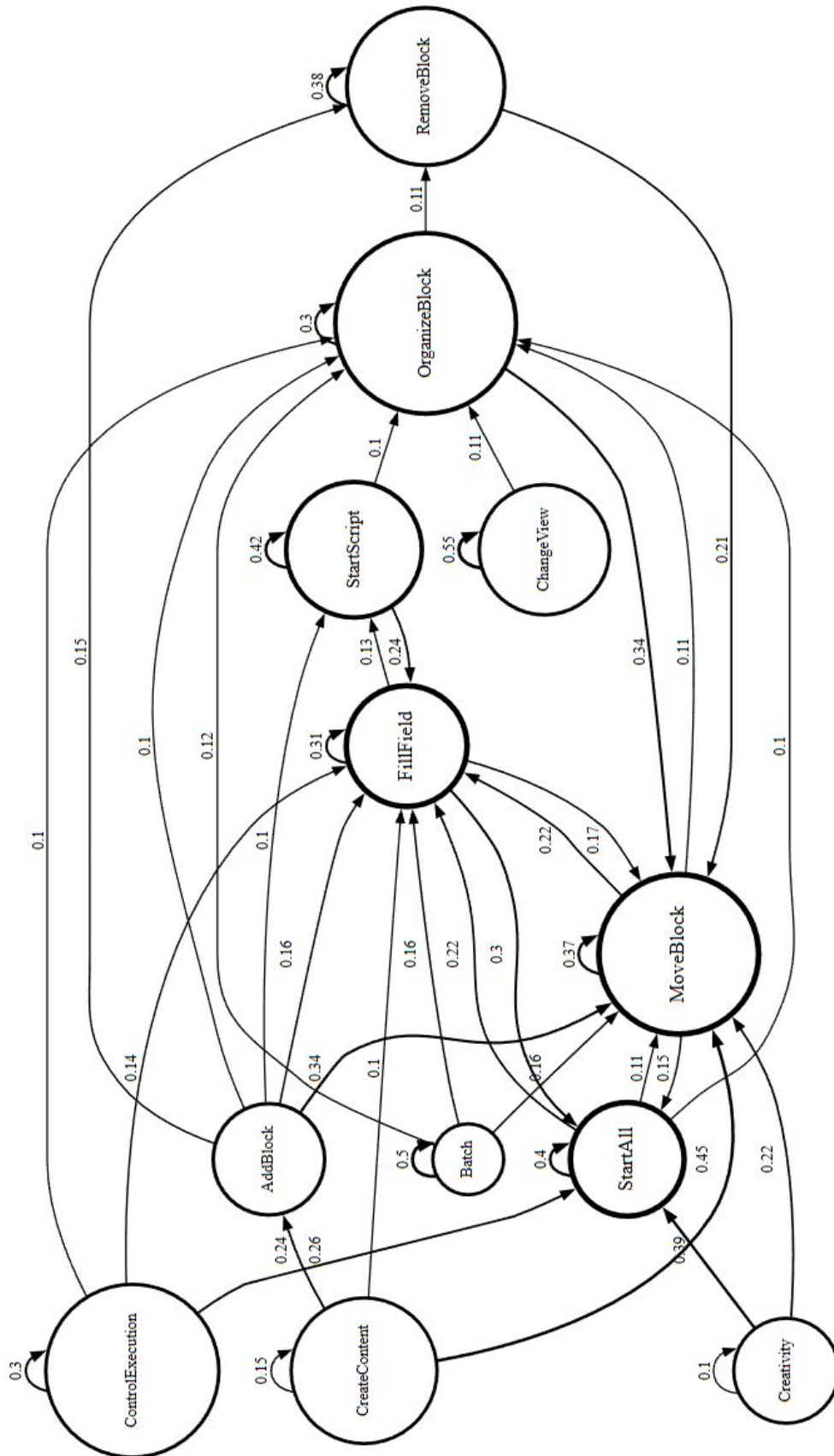


Figure 5.7: Markov chain model of students' NetsBlox actions during the Module 2 project.

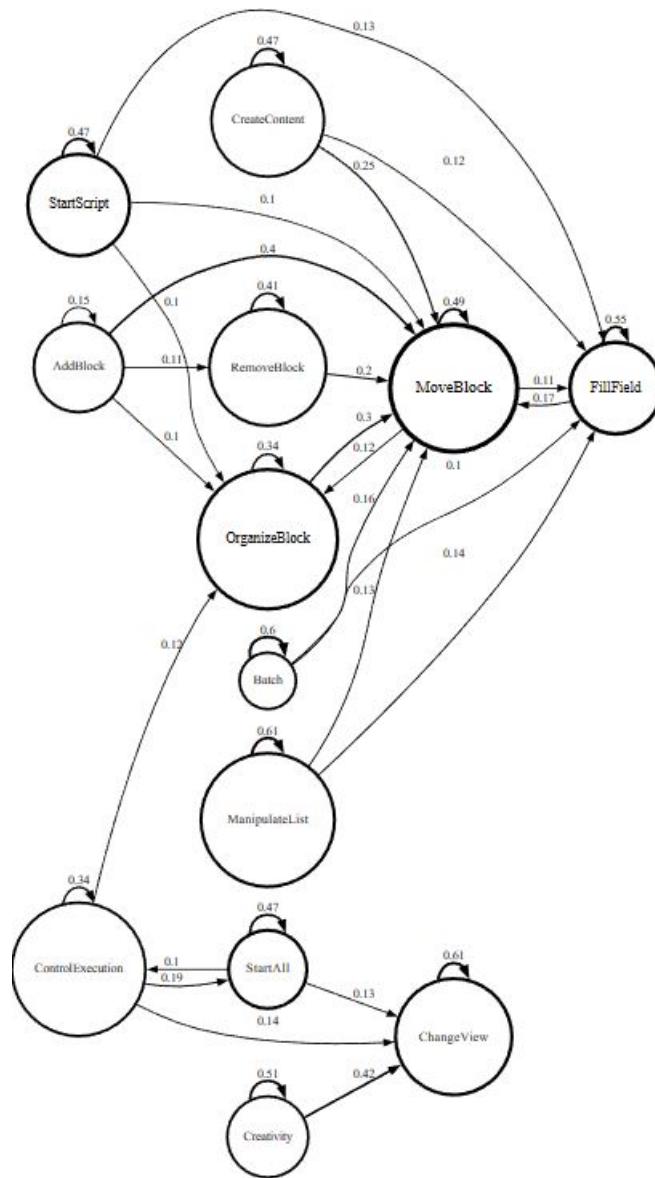


Figure 5.8: Markov chain model of students' NetsBlox actions during the Module 3 project.

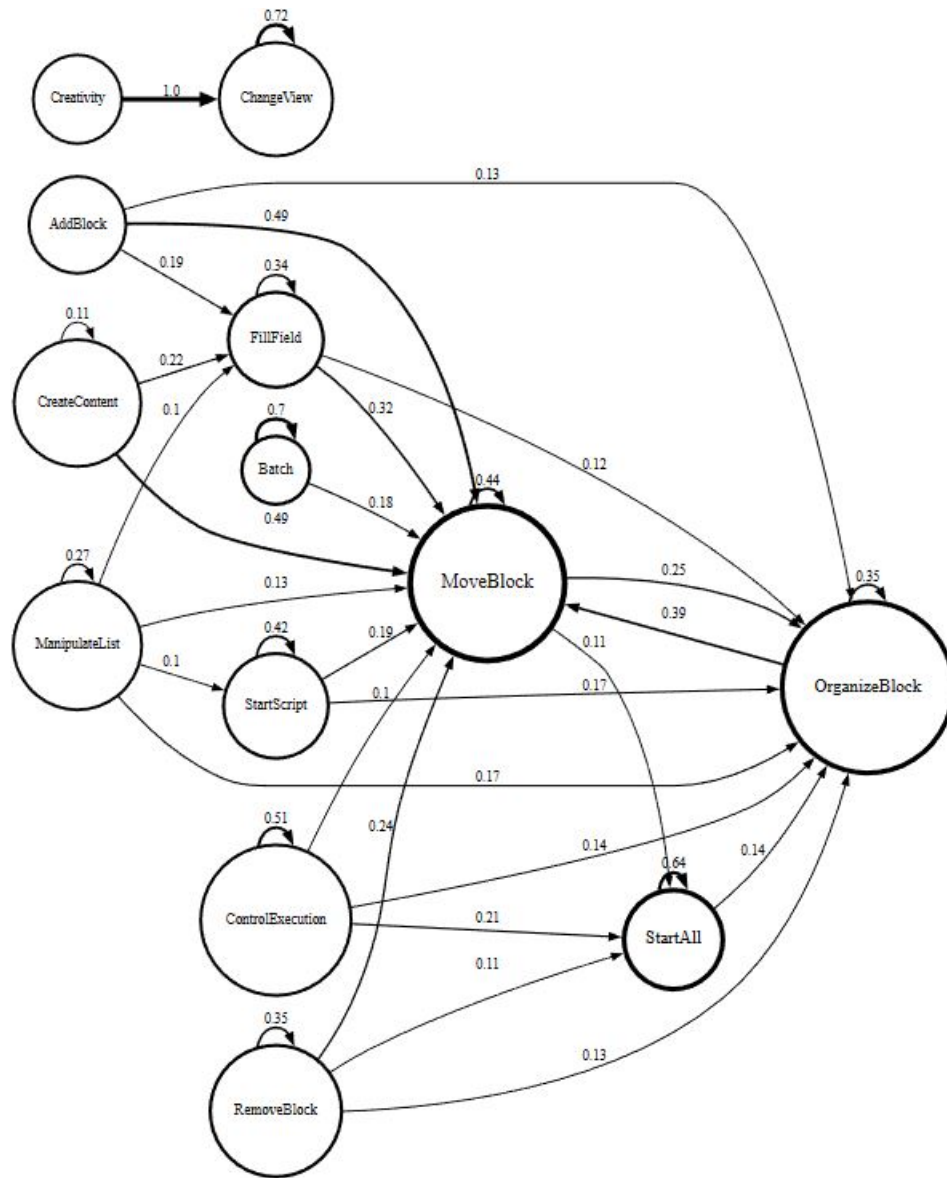


Figure 5.9: Markov chain model of students' NetsBlox actions during the Module 4 project.

Only single transitions are reliably able to be analyzed through the MC approach. It can also be difficult to track important progressions of programming strategies with just this approach. Therefore, we also apply Differential Sequence Mining (DSM) [Kinnebrew et al., 2013] to evaluate how student programming behaviors change over time. In particular, we present comparisons between Modules 1 and 2, Modules 2 and 3, and Modules 3 and 4. DSM works by comparing the instance support of different groups of students. In our case, we have the same set of students for each project, so we separate based off of project number. Instance support, often abbreviated *i*-support or I-S, is the average number of times a particular pattern of programming actions was used by students during the appropriate module. We highlight patterns that were significantly different from project to project as assessed via the I-S metric. We set our *s*-support threshold at 80%, which requires that at least 80% of students performed the sequence of actions in question on at least one of the two projects being compared. There was no gap allowed between actions.

We apply DSM analysis to track differences in sequences of NetsBlox actions from project to project. Each presented table has a number of patterns obtained from the results of the DSM implementation. These do not represent all possible sequences, but are generally representative of the largest absolute change in relative quantities of sequences from project to project. Each pattern (also referred to as sequence) is assigned a number for ease of referring to them. The numbers are unique across all of the tables of results.

This process starts with our first two projects, as shown in Table 5.25. Many of these sequences much more commonly appearing in project 2 (numbers 1, 2, and 5) are related to FillField and MoveBlock actions exclusively. In particular, MoveBlock -> MoveBlock and MoveBlock -> MoveBlock -> FillField both were significantly more likely to occur during Project 2 with very large effect sizes (2.14 and 2.13 respectively). As mentioned in the prior results related to MC analysis, the lack of Solution Assessment actions in between these and the other sequences of largely Solution Construction actions makes more sense within the context of this specific project. Still, the variation in actions shown through the FillField -> StartAll -> MoveBlock sequence (number 7) is more indicative of traditionally successful sequences. The ChangeView -> StartAll sequence (number 8) is only notable as it is the sequence with the highest effect size ($d = 0.93$) amongst DSM results where the sequence occurred more in project 1 than in project 2. This sequence may traditionally be considered ineffective, but fits somewhat within the context of project 1.

A wide variety of sequences are on display in the DSM results of Projects 2 and 3 (Table 5.26). Effect sizes are lower than the extremes of some Project 1 to 2 results, but all would still qualify as high. Patterns 9 and 10 each occurred much more often during project 2 compared to project 3, and each would generally be considered productive combinations of Solution Assessment and Solution Construction actions. Effect sizes are 1.45 and 1.40 for FillField -> StartAll and for StartAll -> MoveBlock -> FillField respectively. The first sequence which is more prevalent in project 3 (number 11) is a ChangeView -> AddBlock process. A

similarly high effect size ($d = 1.38$) accompanies this pattern. It is generally considered as a positive result of students switching between sprites of the game to then recognize that a new block is needed.

The pattern 14 combination of FillField -> StartAll -> FillField is a similarly productive sequence to those previously mentioned, and is once again much more likely to occur during project 2 work with effect size of 1.22. Pattern 15 (ManipulateList -> ManipulateList) is primarily notable for the fact that it is restricted to project 3, as lists had not yet been introduced while students were working on project 2. The final two sequences (16 and 17) are included to show more generally positive behaviors that occurred more often during project 3. StartAll -> ChangeView could indicate running the game to test new additions, realizing that they were working properly, and moving on to another sprite to make further improvements. Meanwhile, MoveBlock -> CreateContent would likely occur from students modifying their programs and realizing that a new variable was needed. Each of these sequences had effect sizes of 1.10 and 1.05 respectively. Less productive than most sequences mentioned here would be patterns 12 and 13 - the trio of ChangeView actions as well as the six repetitions of MoveBlock - that occur much more often in project 3 than project 2. Though somewhat unfair comparisons due to the nature of the two projects, these long sequences of actions should not occur during students' work. They are representative of depth-first programming [Grover et al., 2016a] as opposed to programming in parts.

When comparing projects 3 and 4, there are no sequences that are significantly more likely to occur in student work during project 4. Instead, Table 5.27 showcases the most relevant patterns that were completed much more often during project 3 than during project 4. The largest effect size ($d = 1.98$) was for a MoveBlock -> FillField sequence (number 18). Though a combination of two Solution Construction actions, these are still fairly compatible when taken together. For example, this could involve attaching a block to the end of a script and then typing in a numerical value. More traditional positive patterns can be seen as numbers 20 and 21. Numbers 20 and 21 are combinations of Solution Assessment and Solution Construction actions and were very prevalent amongst strategies students incorporated into their project 3 work. We explored similar patterns in past work [Yett et al., 2020b] as examples of ineffective trial-and-error approaches [Murphy et al., 2008]. ChangeView -> ChangeView -> ChangeView was previously mentioned as a common pattern for project 3, and is an example of several Solution Independent actions connected together. Pattern 22 ($d = 1.48$) is also a potentially negative situation of many repeated MoveBlock actions, though this type of behavior is sometimes necessary to reach a logical 'break-point' in the program prior to testing via Solution Assessment. It is also worth noting that, when comparing the total number of actions taken during each project, this pattern along with numbers 23 and 24 occurred at roughly the same relative frequency between the two projects. This abnormality will be discussed further in Chapter 6.

Table 5.25 Selected DSM Results - Project 1 to Project 2

Number	Pattern	I-S (P1)	I-S (P2)	p-score	Effect Size
1	MoveBlock, MoveBlock	5.61	22.68	< 0.01	2.14
2	MoveBlock, MoveBlock, FillField	1.36	7.57	< 0.01	2.13
3	MoveBlock x 4	0.61	3.36	< 0.01	1.56
4	OrganizeBlock, MoveBlock	4.07	15.25	< 0.01	1.52
5	FillField, FillField, MoveBlock	1.04	4.04	< 0.01	1.51
6	RemoveBlock, MoveBlock, MoveBlock	0.43	1.68	< 0.01	1.26
7	FillField, StartAll, MoveBlock	0.5	2.54	< 0.01	1.19
8	ChangeView, StartAll	2.64	0.76	< 0.01	0.93

Table 5.26 Selected DSM Results - Project 2 to Project 3

Number	Pattern	I-S (P2)	I-S (P3)	p-score	Effect Size
9	FillField, StartAll	24.18	4.68	< 0.01	1.45
10	StartAll, MoveBlock, FillField	2.14	0.14	< 0.01	1.40
11	ChangeView, AddBlock	0.5	2.57	< 0.01	1.38
12	ChangeView, ChangeView, ChangeView	1.54	7.36	< 0.01	1.31
13	MoveBlock x 6	0.82	3.43	< 0.01	1.24
14	FillField, StartAll, FillField	8.14	0.75	< 0.01	1.22
15	ManipulateList, ManipulateList	0	11.61	< 0.01	1.16
16	StartAll, ChangeView	1.25	5.14	< 0.01	1.10
17	MoveBlock, CreateContent	0.75	2.07	< 0.01	1.05

Table 5.27 Selected DSM Results - Project 3 to Project 4

Number	Pattern	I-S (P3)	I-S (P4)	p-score	Effect Size
18	MoveBlock, FillField	12.25	2.32	< 0.01	1.98
19	ChangeView, ChangeView, ChangeView	7.36	0.25	< 0.01	1.73
20	MoveBlock, StartScript	6.75	0.54	< 0.01	1.65
21	StartScript, FillField	6.07	0.25	< 0.01	1.59
22	MoveBlock x 4	8.5	1.89	< 0.01	1.40
23	AddBlock, MoveBlock, MoveBlock	3.75	0.21	< 0.01	1.35
23	CreateContent, MoveBlock	2.61	0.64	< 0.01	1.33
24	FillField, FillField, MoveBlock	3.11	0.54	< 0.01	1.23

Table 5.28 For each survey, we present the likelihood of a student’s response falling under the appropriate reasoning type. It was possible for one response to indicate more than one type of reasoning, so rows will not sum to 1. Dashes indicate that no responses were coded as that reasoning type for that survey.

Survey	Minimal Reasoning	Developing Reasoning	Visual Reasoning	Numerical Reasoning	Causal Reasoning	Mechanistic Reasoning
M2	0.05	0.54	0.37	0.12	0.1	0.1
M4	0.21	0.36	-	0.48	0.37	0.12
M5part1	0.44	0.25	-	0.14	0.15	-
M5part2	0.33	0.27	0.23	0.19	0.18	0.13

Table 5.29 p-score (Effect Size) shown for each type of reasoning as we transition from survey to survey throughout the intervention. Negative effect size corresponds with a decrease from the earlier survey to the later survey and is indicated by red text. However, for these reasoning types, a negative effect size would be desirable. Bold rows indicate a moderate ($d \geq |0.5|$) or greater effect size as measured by Cohen’s d.

Survey	Minimal Reasoning	Developing Reasoning
M2 to M4	< 0.01 (0.72)	< 0.01 (-0.54)
M4 to M5part1	< 0.01 (0.72)	0.06 (-0.33)
M5part1 to M5part2	0.06 (-0.08)	0.66 (0.07)
M2 to M5part2	< 0.01 (1.15)	< 0.01 (-0.83)

To conclude RQ3, we move away from programming actions and focus on student responses to a series of surveys. These surveys were part of various tasks during the modules of the curriculum. Survey M2 came from Module 2, specifically the previously described clock debugging task (Appendix C.1). M4 came from Module 4 and directly relates to the introductory project where students explored a message client project that was set up to communicate to a classroom-wide server. M5part1 was a survey students completed after creating a solution to and testing a square driving project with the robots. Finally, students filled out M5part2 after they explored two existing projects meant to demonstrate the bus and star topologies. As a reminder, the reasoning types themselves were described in Section 4.4.2.

Table 5.30 p-score (Effect Size) shown for each type of reasoning as we transition from survey to survey throughout the intervention. Negative effect size corresponds with a decrease from the earlier survey to the later survey and is indicated by red text. Bold rows indicate a moderate ($d \geq |0.5|$) or greater effect size as measured by Cohen’s d. “N/A” indicates that the survey combination in question was not applicable for that reasoning type.

Survey	Visual Reasoning	Numerical Reasoning	Causal Reasoning	Mechanistic Reasoning
M2 to M4	< 0.01 (-0.46)	< 0.01 (1.19)	< 0.01 (0.98)	0.70 (0.10)
M4 to M5part1	N/A	< 0.01 (-1.13)	< 0.01 (-0.76)	0.80 (0.06)
M5part1 to M5part2	N/A	0.23 (0.23)	0.29 (0.15)	N/A
M2 to M5part2	< 0.01 (-0.46)	0.06 (0.28)	0.09 (0.37)	0.38 (0.16)

Table 5.28 presents the initial percentages of each reasoning type for each of these surveys we have described. Mechanistic Reasoning is the most consistent reasoning type. It was expected and held true that this would be the most difficult for students to demonstrate. Ratios are 0.1, 0.12, and 0.13 at different parts of the curriculum, slowly increasing over time. There is a large jump in Numerical and Causal Reasoning during M4 (0.48 and 0.37 respectively), but otherwise these ratios are near the same level as those of Mechanistic Reasoning. Visual Reasoning is only possible for students to demonstrate on two of the surveys. Scores went down from 0.37 during M2 to 0.23 on M5part2. There is a generally downward trend for Developing Reasoning, starting from a high of 0.54 during M2 and settling at 0.25 and 0.27 for the two parts of M5. Minimal Reasoning ratios show no real trend, but rose from a very low initial ratio of 0.05 to higher ratios on the remaining surveys.

Tables 5.29 and 5.30 show the relationship between each logical pair of surveys. We progress from M2 to M4, M4 to M5part1, M5part1 to M5part2, and then conclude with M2 to M5part2 as a type of pre- to post-analysis. The significant ($p < 0.01$) increase with a large effect size ($d = 1.15$) in overall terms for Minimal Reasoning indicates that students struggled more in explaining responses related to the more complex networking and security content. However, there is a significant ($p < 0.01$) decrease with large effect size ($d = -0.83$) in Developing Reasoning responses, somewhat offsetting this situation. As noted from Table 5.28, there is an interesting progression for the Numerical and Causal Reasoning ratios. In particular, each had a significant ($p < 0.01$) increase with large effect size ($d = 1.19$ and $d = 0.98$) from M2 to M4. This is accompanied by a corresponding significant ($p < 0.01$) decrease from M4 to M5part1, with high effect size ($d = -1.13$) for Numerical and moderate ($d = -0.76$) for Causal. In an overall sense, the change is minimal for Numerical, Causal, and Mechanistic Reasoning. Visual Reasoning missed on our threshold for moderate effect size ($d = -0.46$), but did decrease significantly ($p < 0.01$).

5.4 Summary of Chapter

This Chapter presents the main results of our analysis of the data obtained from the interventions previously described in Chapter 4. We began with RQ1: *How did student performance, engagement, and attitudes measured using pre-post-assessments and surveys change as a result of our intervention?*. This was addressed through descriptive statistics of primarily pre- to post-test and survey responses from students. We saw positive trends in performance and confidence and noticed some concerns in engagement. Next was RQ2: *What was the learning progression of students in CT, networking, and security-related assessments, as well as key constructs that bridged these broader categories?*. Here, correlational analyses showed that projects, assessments, and surveys all related to results obtained at later time periods of the intervention. Additionally, the linear regression plots could successfully predict performance in areas such as CT and Conditionals. They

were less accurate for Messaging and Topology in particular.

We concluded the Chapter with RQ3: *What was the strategy progression of students based on their programming actions during projects and their open-ended questionnaire responses?*. The Markov chains aided our understanding of common transitions between categories of programming actions. Differential Sequence Mining then showed the large disparities between Project 1 and Project 2, and between Project 3 and Project 4. This technique was able to provide additional relevant context for changes in student behaviors between Projects 2 and 3 in particular. Finally, we finished this research question with results obtained from careful coding of student responses to selected surveys. We identified some difficulties students experienced when trying to progress beyond basic Minimal Reasoning skills. The main takeaways mentioned here from each research question will be discussed in more detail in the upcoming Discussion Chapter (Chapter 6).

CHAPTER 6

Discussion

We now discuss the significant results from Chapter 5. Overall, we did find an improvement in student performance in almost all of the areas of interest. Students started out with a reasonable baseline of CT knowledge as indicated by their pre-test scores. They showed significant gains from pre- to post-test but the effect size was small. Much of the improvement was concentrated in a few categories, primarily Loops and Evaluation. A similar result was seen in our networking test. Scores started lower (5.17 out of 13, compared to 15.25 out of 26 for CT) before reaching 7.35 on the post-test. The learning gains were statistically significant for Overall Networking, Topology, and Overall Security. We saw that students tended to perform better on mid-module assessments and the in-class final compared to the pre-post-tests. With the collected data, it is difficult to determine if this was primarily due to improved understanding, memorizing concepts that were then forgotten by the time the post-test was conducted, or differences in assessment difficulty.

Pre- to post-survey results were mixed. Many areas were not expected to change much as they measured innate attitudes students held about learning [Midgley et al., 2000]. These constructs did not show much change, e.g., in students' mastery-approach, performance-avoidance, or performance-approach goals. However, we hoped that our intervention would positively influence task values, behavioral and cognitive engagement, and the computing interest of students. As measured by these questions, this increase did not occur. There could be other factors at play, such as the small number of questions associated with most survey categories to minimize the overall length. Another possible explanation was the overall situation in the classroom by the end of the semester. There were many disruptions in the class schedule during the final module, and as stated by the teacher, "*The largest sustained struggle has been student attention to individual pieces, which has largely been complicated by the weird school schedule*".

This is not meant to avoid the fact that the curriculum and implementation need to be improved. At the end, the teacher recommended applying "*tighter timelines to sustain interest and interrupt with solution models more frequently using incomplete coding solution snippets*". It was also suggested to "*Structure a robot sharing pattern, e.g., 'now pass your robot to the right' between members of a group developing the robot task in the virtual environment, and then occasionally taking turns on the physical robot*". In short, better curricular structuring along with making use of the affordances of both the physical and virtual robotics solutions could potentially improve student engagement and interest.

There were some encouraging signs in the survey despite these mentioned difficulties. In particular, though not quite reaching a moderate effect size, there was a significant increase from pre- to post-survey in

students' computing confidence. This was measured across seven different questions - all of the statements beginning with "Right now, how confident are you in your ability to..." on the survey shown in Appendix B.3. This further validates the results of the pre-post-test that indicated student learning, as they also expressed increased confidence in their abilities through this self-assessment.

Our student group was quite varied (Table 4.3), with a mix of 7th-graders and 8th-graders, male and female students, and racial identities (primarily distributed among Asian, Black/African-American, and White students). We hoped to study the differences between these identities and how they impacted students' performance, attitudes, and interactions with the curriculum. The results of this process are shown in Tables D.3 and D.4 in Appendix D. Unfortunately, none of the findings were particularly novel or unexpected. There were some trends related to our underrepresented Female and Black/African-American populations, but they were generally negative associations with performance on tests or projects. The majority of these correlations were not significant and would require further study. On the other hand, outside of a difference in Module 1 actions taken, the 7th-grade students did not significantly differ from the 8th-grade students, supporting our decision to combine their data.

As part of the correlation analyses from RQ2 (Section 5.2), we noted several occasions where student performance on early CT assessments tended to correlate with successes later in the curriculum. For example, pre-test CT results tended to lead to higher results throughout the curriculum (see Tables 5.13 and 5.17). Likewise, projects 2 and 3 had moderate correlations to many of our other assessment metrics (see Table 5.15). These were results we were hoping to find in terms of higher CT competencies leading to better engagement with the curriculum and better performance in the related concepts. However, this finding comes with a caveat. We as researchers and curriculum developers need to prioritize enabling those students who are behind in CT to more effectively reach a similar level as their peers. This may be a challenge, but it presents potential avenues for future work to be discussed in Chapter 7.

Variables were a heavy focus throughout the curriculum. Starting from the first CT module (Module 2), they consistently related to many components of students' learning materials (Table 5.10) and assessments. There were also nine questions related to variables on the pre-post-test. Students performed well on Variables content during the mid-module assessments, with a fairly steady increase throughout (Table 5.11). This culminated in high scores on the classroom final. However, this progress did not show as a significant increase from pre- to post-test in this area (Table 5.1). This also led to fairly large errors when comparing our predicted result from linear regression to the actual performance of students (Figure 5.4). One consideration is the difficulty of the post-test when compared to some of the earlier assessments. Another possible explanation is the large variety of contexts and data types that can be grouped together under the variables umbrella. Storing lists or strings within variables can be difficult concepts for students to grasp, which may have led to

increased confusion.

We incorporated security in general and encryption in particular in Module 5. We went through the Caesar cipher process by hand on top of using encryption custom blocks in NetsBlox ([Lédeczi et al., 2019],[Broll et al., 2017]). There was also a significant increase in student performance from pre- to post-test in the broad Security area. However, the change was minimal for the Encryption topic, and students also struggled with the classroom final. For the pre-post-test, the research team noticed too late that one of the encryption questions (Question 5; Appendix B.2) was unintentionally misleading for students. An example of encryption was shown to the students to give them some chance of completing the task on the pre-test. This example used an encryption key that was also used in one of the incorrect answers provided. That particular incorrect answer is correct until the final letter of the encrypted word and was chosen by 69% of students on the pre-test and 77% of students on the post-test. Another cause could be the classroom implementation of this Module and the need to provide “*Different overall pacing [that] may help engagement with some of the more nuanced networking and cybersecurity activities.*”

Conditionals and Loops are generally thought of as complementary topics in programming. Most loops rely upon a condition for controlling either when to stop or how many iterations of the loop to complete. In NetsBlox, this second type of loop can be abstracted away using a “repeat” block that does not directly require any conditional knowledge. On top of this, although the two concepts are weighted approximately equally by the CSTA [Seehorn and Clayborn, 2017] and through the K-12 CS Framework [K-12, 2016], only loops are mentioned as part of the Tennessee CS standards that were our primary motivation [of Education, 2018]. This led to a much larger emphasis on loops when compared to conditionals in our curriculum. We started on loops in Module 2 compared to Module 3 for conditionals, and the usage counts were much higher for loops to place it on a similar level as variables (Table 5.10). This emphasis on loops did seem to resonate with the students, as they significantly improved ($p < 0.01$) with a small effect size ($d = 0.38$) from pre- to post-test. This potentially addressed some of the loops misconceptions of students learning to program [Grover and Basu, 2017]. Meanwhile, students performed fairly well on conditionals on the pre-test (9.79 out of 14, or 70%), but did not significantly change that performance on the post-test (9.98 out of 14, or 71%; Table 5.1 and Table 5.11).

Students started with essentially no knowledge of network topologies (Pre-Test average score of 28%; Table 5.11) but demonstrated their learning on the module assessments, classroom final, and through to the post-test (72%, 75%, and 57% respectively). This is still a complex topic and very theoretical in nature, including a set of questions on the pre-post-test that required students to critically think about the impact of one node failing for each of the core topology types that were covered (Questions 9 through 12; Appendix B.2). Our goal was to minimize future misconceptions that could evolve from initial simplistic scaffolding

techniques while successfully educating the students on an area that is emphasized in all of our standards of interest - CSTA [Seehorn and Clayborn, 2017], the K-12 CS Framework [K-12, 2016], and Tennessee CS Standards [of Education, 2018]. We appear to have accomplished this objective via the first embodied learning activities and the reinforcement through NetsBlox sprite-based projects.

Student skill at evaluation of existing programs is harder to quantify than many of our more traditional CT, networking, and security concepts. However, we made a concerted effort to incorporate learning opportunities in this area throughout the curriculum (Table 5.10). This included tasks such as determining different functionalities of starter programs and switching projects with other students to evaluate how they would execute. Evaluating existing programs was also incorporated into many of our module-based assessments. Students consistently improved their performance throughout the curriculum, leading to a moderately positive slope and a model that could somewhat accurately predict post-test performance from our linear regression analysis (Figure 5.5). There is still room to improve our materials to more systematically introduce other facets of evaluation, but it appears that we would be building from a strong foundation.

The project associated with Module 2 - a drawing project that required certain customizations and custom block creation but left some freedom for students to approach the problem in different ways - seemed to be successful. Students took a lot of actions during this project, indicating their engagement with the project and desire to successfully complete it. They demonstrated a number of positive transitions between action types (Section 5.2) during their work. We also discovered correlations between performance on this particular project and performance on many of the more difficult subjects of the curriculum. The Module 2 project showed $\rho \geq 0.4$ with nearly all computed metrics from Modules 4 and 5 as well as various post-test areas (Table 5.15). There were further relationships found with the concepts actually covered in the project (Variables and Loops; Table 5.16).

On the other hand, student work on the project for Module 4 indicates that the project may be in need of some improvement. This was an issue with the messaging features of NetsBlox in general due to the syntax-heavy nature of needing to perfectly type the correct names for projects and other minute details. Their teacher mentioned that “*They did not have the same struggles with syntax, especially of the IDs, as prior classes have*” but that students still “*tended to lose interest in their own messaging if they could not connect successfully*”. We attempted to scaffold this process as much as possible, but continuing to refine this approach could aid students in their persistence for this particular project. The project itself did not provide as many opportunities as desired for experimentation thanks to the scaffolding involved and very specific requirements to communicate successfully with the automated evaluator. Returning to a manual approach for student testing of their projects while giving them more flexibility in creating the various scripts could rectify some of these concerns.

For the Differential Sequence Mining, the comparison between projects 2 and 3 (Table 5.26) was likely the most interesting and relevant. These two projects had roughly the same number of combined Solution Assessment, Solution Construction, and Solution Independent actions taken during the appropriate time periods. 11056 actions were associated with all included project 2 files, and 13015 were associated with all included project 3 files. For comparison, there were 5419 actions during project 1 and only 3523 actions during project 4 programs. It follows that differences in the sequences applied in projects 2 and 3 are truly meaningful instead of potentially serving as a by-product of sheer action disparities. This is a potential limitation of the DSM technique that cannot easily be addressed, though one option could be to scale pattern counts to match the ratio of actions between projects.

From a big picture perspective, there are a few main patterns that emerge from all of these points. Student performance significantly improved from the beginning to the end of the intervention in CT, Networking, and Security. Prior knowledge of CT as well as the ability to learn those concepts quickly during the curriculum appeared to have a substantial relationship with learning the more advanced concepts. CT and the aggregated overall performance could be accurately tracked over time. Student computing confidence improved, but students' self-assessments did not indicate significant change in their computing interest or engagement. As an additional part of this discussion, we identified several areas that could be considered limitations. These would need to be addressed in future work. We dive further into such information in the next chapter (Chapter 7) to conclude this dissertation.

CHAPTER 7

Conclusions

7.1 Contributions

Our initial claims for research contributions were listed as follows:

- Item 1 - Creation, refinement, and implementation of a curriculum for introducing K-12 students to a variety of CT, networking, and cybersecurity topics. This also included providing tools to teachers such that they could implement the curriculum on their own.
- Item 2 - Extension of previously implemented assessments while contributing new ones to form a novel, comprehensive means of evaluating students' progress through the curriculum.
- Item 3 - Providing additional information to students, teachers, and researchers to ease their burdens during implementation, including insight gained from formative assessments, programming actions, and attitudinal surveys.
- Item 4 - Development of systematic methods to evaluate the effectiveness of our curriculum through learning analytics and results from assessments.
- Item 5 - Improved understanding of the progressions of student learning and strategy usage.

Through this work, we presented a completed curriculum (Item 1) that aimed to capture the right balance between instructional scaffolds and open-ended problems. Our approach was to develop a comprehensive curriculum with multiple Modules that covered CT and computing, networking, and cybersecurity concepts and practices. The structure of the development (Section 4.1) included co-design with a middle school CS teacher with expertise in the area who is now able to continue iterating upon the curriculum in the classroom. We successfully mapped materials and assessments to sources such as the Tennessee CS Standards [of Education, 2018] to validate the curriculum. There were of course areas of improvement mentioned in Chapter 6 and further addressed below in Section 7.2.

In addition, we developed and demonstrated the use of a comprehensive assessment framework that encompasses formative and summative approaches (Item 2; initially presented in Section 4.3). Surveys - as a form of self-assessment - and programming actions and projects were considered on top of the traditional tests and other assignments. Each of these were integrated within all Modules of the intervention as shown in Figure 3.1. We measured students' prior knowledge and motivation using pre-tests and detailed questionnaires.

Interspersed within modules were formative assessments that aided students and their instructor in tracking their learning progress (Item 3). We were also able to identify common errors and misconceptions that the teacher could address through instruction or that future iterations of the curriculum could improve upon. The robotics plus programming final Module provided the hands-on component of the intervention to motivate and engage students.

Results from all three research questions including the analysis techniques described throughout Chapter 5 related to Item 4. Though the techniques themselves were not truly novel, they have yet to be applied in this overall context. This approach relates to Item 5 as well. We successfully contributed an improved understanding of the progression of student learning from novice programming and CT skills through advanced security and networking concepts. This was presented throughout RQ2 and RQ3 (Sections 5.2 and 5.3). We had an opportunity to study this progression throughout an entire semester of the school year for a total of 48 7th-grade and 8th-grade students. Prior knowledge did make a difference in the performance of students moving throughout the remainder of the intervention. We were able to analyze students' progress in and measure their learning of CT, networking, and cybersecurity concepts and practices.

7.2 Limitations and Future Work

One potential area of future development is further scaffolding of student learning. NetLogo [Tisue and Wilensky, 2004] simulations could be an option to convey the scale of certain scenarios. Connecting many networked nodes, showing a large number of robots following commands, or combining the ideas to show robots conveying commands from one to the next would all be more feasible via that platform. An on-demand feedback system may now be possible directly in NetsBlox and could be extremely useful for the more linear projects throughout this curriculum in particular. This aspect would supplement Item 3 from our Contributions (Section 7.1), which was under-explored compared to the other items mentioned.

Additional use of the autograder and similar NetsBlox tools may also be appropriate to better scaffold student learning during the third component (the debugging or Parsons problem tasks) of each Module. An important consideration in all of this is the difficulties that students face when they have to learn about new tools. Instead of scaffolding student learning, we could inadvertently cause excessive cognitive load and detract from student learning. We may determine that the best way to reach the specific student population we are working with would be to reduce the amount of technology involved. Instead, we could re-introduce or improve upon existing embodied activities or other forms of unplugged activities.

Time constraints towards the end of the curriculum caused by standardized testing and other outside factors limited the impact of the final Module. More systematically and continuously leading students through the materials while providing impetus through engaging robot challenges could lead to improved understand-

ing by the students. The teacher recommended an “overall framing/previewing of the obstacle course” to “provide better context for what they’re attempting to piece together”. Re-framing past work from cybersecurity curricula into the networking-focused curriculum presented here or refining current networking tasks to tap into the competitive nature of the students could each be effective.

The curriculum itself needs revisions in a number of areas. Each student gained some understanding of the human factors of cybersecurity implicitly by observing the impact of other students interfering with their robot. They also completed a password activity to encourage them to think through securing their own accounts and data. However, additional time should be taken to incorporate these ideas further within the core curricular progression. Other areas were present at some level but the Tennessee CS standards laid more emphasis on them than their inclusion in our curriculum would indicate. For example, Encryption is related to four distinct standards, but was relegated to only some components of the final Module in our current version of the curriculum. Man-in-the-middle attacks and communication protocols likewise were featured in three Tennessee standards but were relatively underrepresented in our curriculum.

Relating back to our previous Discussion chapter, one highlight was the occasional lack of student engagement with the curriculum. This was found both from survey results and - at certain points - via feedback from interviews with the classroom teacher. Ultimately, though they did learn much of the expected content and gain confidence in their abilities, we also highly prioritize keeping students engaged and interested in persisting in STEM. There is no easy “fix” for this concern. Continuing to update existing materials to better take advantage of past work presented in our literature review is one potential route. Keeping up with the newest findings can also be productive. Ultimately, the networking features in particular were only tested in a classroom setting through the initial trial study prior to this implementation, so improvements are certainly possible.

Additionally, one area for future work would be related to the role of the robot in students’ learning and engagement. We surveyed this area heavily throughout the literature review of Chapter 2. However, it was and remains a difficult task to quantify the impact of the robot specifically. Due to time constraints, we were not able to systematically assess students to attempt to capture this information. The plan was to introduce concepts via embodied demonstrations or programming simulations, test students on their knowledge, then provide robotics tasks as reinforcement before another round of testing. This or another approach could be integrated within the curriculum.

Despite the comprehensive data analysis plan, there were some issues in this process as well. The environment occasionally crashed, impacting our ability to complete the curricular materials or collect data. Students would also fail to save their completed work quite often, leading to a need to recreate programs or causing us to lose project files that could have been included in our scoring system. On the assessment

side, the relatively few number of questions for important categories such as Testing and Debugging and Encryption made it difficult to implement our linear regression approach successfully. Likewise, the disparity in programming actions for different projects completed by students impacted our Differential Sequence Mining implementation. Better planning could rectify some of these issues. Additional environment features like autosaving projects or the ability to generate Abstract Syntax Trees or other models immediately from project files would also be extremely valuable. These tools could allow for automated feedback to teachers and students while aiding researchers in evaluating students after the fact.

The CT portion of the intervention appeared to be particularly impactful. Based on results and classroom observations, students were engaged with and learning from the curricular materials and activities. However, the latter modules related to networking and security were less successful. Part of the problem likely stemmed from the previously mentioned disruptions in the class schedule and difficulties in structuring the sharing of physical robots. We could also do a better job at making big picture connections between the CT, networking, and security concepts to allow students to understand the purpose of each step in the process. Still, other more specific areas could be addressed and improved.

Removing some of the roadblocks students encountered in the network communication side through additional scaffolding would be one option. This would involve additional custom variables and blocks built into the starter projects given to students plus further instruction on how to catch and fix any typos. Another option within the curriculum itself could be to better distinguish between the networking and security components in Module 5. By separating out into multiple Modules and better structuring the process to match the more successful CT Modules, we may see more student engagement and learning throughout those materials.

Making these improvements and repeating a similar classroom intervention would greatly progress this research. It would also continue to improve our understanding of student learning progression and strategy usage from both a programming and reasoning perspective. This process would additionally resolve a limitation of this study - the relatively small student population ($n = 48$). The limitation was somewhat addressed by the length and complexity of the curriculum and the data collected, but the larger population could lead to more confident predictions via linear regression and clearer patterns and correlations of student learning.

However, it is also possible that the previously mentioned disconnect between the initial CT component and the later networking and security materials is not something that can be addressed via classroom or curricular refinements. Instead, despite the success of CT as a conduit for similarly advanced concepts, there may be better approaches to the teaching of networking and security. Despite the presence in the Tennessee and CSTA standards, networking and security may be too complex for students to want to interact with even at the abstract level. A repeat study could reveal more evidence supporting or refuting this perspective.

Appendix A

Appendix A - List of Publications

- Yett, B., Hutchins, N., Stein, G., Zare, H., Snyder, C., Biswas, G., Metelko, M., and Lédeczi, Á. (2020). A hands-on cybersecurity curriculum using a robotics platform. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1040–1046.
- Yett, B., Snyder, C., Hutchins, N., and Biswas, G. (2020). Exploring the relationship between collaborative discourse, programming actions, and cybersecurity and computational thinking knowledge. In *2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 213–220. IEEE.
- Yett, B., Snyder, C., Zhang, N., Hutchins, N., Mishra, S., and Biswas, G. (2020). Using log and discourse analysis to improve understanding of collaborative programming. In *Proceedings of the 28th International Conference on Computers in Education*, volume 1, pages 137–146. Asia-Pacific Society for Computers in Education.
- Yett, B., Hutchins, N., Snyder, C., Zhang, N., Mishra, S., and Biswas, G. (2020). Evaluating student learning in a synchronous, collaborative programming environment through log-based analysis of projects. In *International Conference on Artificial Intelligence in Education*, pages 352–357. Springer.

Best Student Paper Award

- Lédeczi, Á, Maróti, M., Zare, H., Yett, B., Hutchins, N., Broll, B., Volgyesi, P., Smith, M. B., Darrah, T., Metelko, M., et al. (2019). Teaching cybersecurity with networked robots. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 885–891.
- Girma, D., Yett, B., Hutchins, N., and Biswas, G. (2020). Development of a Python-based Platform for Teaching Computer Science. *Young Scientist, A High School Research Journal*.
- Snyder, C., Hutchins, N. M., Biswas, G., Emara, M., Yett, B., and Mishra, S. (2020). Understanding collaborative question posing during computational modeling in science. In *International Conference on Artificial Intelligence in Education*, pages 296–300. Springer.
- Snyder, C., Narasimham, G., Hutchins, N. M., Biswas, G., and Yett, B. (2022). Examining how prior knowledge impacts students’ discussions and knowledge construction during computational model-building. In *2022 AERA Annual Meeting*

- Snyder, C., Hutchins, N. M., Biswas, G., Narasimham, G., Emara, M., and Yett, B. (2022). Instructor facilitation of STEM+CT discourse: engaging, prompting, and guiding student's computational modeling in physics. ISLS 2022.

Appendix B

Appendix B - Full Summative Assessments

B.1 CT Assessment

Page 1

Computational Thinking

Do your best to answer each question to the best of your abilities!

1) Enter your NetsBlox Username here: _____

For the next 3 questions, consider the following program and this scoring information. Bill gets a score of 9 points on the quiz while Janet scores 10 points and Kim scores 5 points.

If (quiz-score is equal to 10)
Then: Get the 'You're a pro' sticker

If (quiz-score is greater than 7)
Then: Get the 'Good job' sticker

2) What sticker or stickers should Bill receive? You're a pro
 Good job
 Study harder
 None of the above (only choose this option if you choose none of the others)

3) What sticker or stickers should Janet receive? You're a pro
 Good job
 Study harder
 None of the above (only choose this option if you choose none of the others)

4) What sticker or stickers should Kim receive? You're a pro
 Good job
 Study harder
 None of the above (only choose this option if you choose none of the others)

For the next 3 questions, suppose we keep the same scoring information (Bill = 9 points, Janet = 10 points, Kim = 5 points) but change to this program.

If (quiz-score is greater than 7)
Then: If (quiz-score is equal to 10)
Then: Get the 'You're a pro' sticker
Else: Get the 'Good job' sticker
Else: Get the 'Study harder' sticker

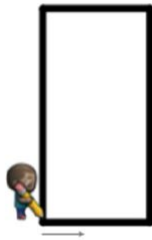
5) What sticker or stickers should Bill receive? You're a pro
 Good job
 Study harder
 None of the above (only choose this option if you choose none of the others)

Figure B.1

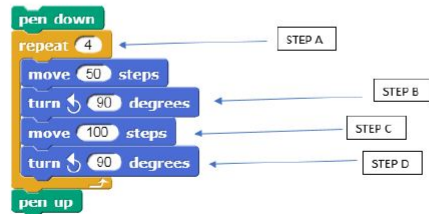
- 6) What sticker or stickers should Janet receive?
- You're a pro
 - Good job
 - Study harder
 - None of the above (only choose this option if you choose none of the others)
-
- 7) What sticker or stickers should Kim receive?
- You're a pro
 - Good job
 - Study harder
 - None of the above (only choose this option if you choose none of the others)

Use this information to answer the next question

The instructions should make the artist draw the following rectangle once:



The rectangle should be 50 steps wide and 100 steps high. In which step of the instructions is there a mistake?



- 8) In which step of the instructions is there a mistake?

- STEP A
- STEP B
- STEP C
- STEP D

Figure B.2

Descriptions of Gabriela's and Lucia's algorithms

For a class assignment, Gabriela and Lucia each created an algorithm that has a robot drive laps on the screen.

Gabriela's algorithm

- Step 1:** Ask how many laps the robot should drive. Go to Step 2.
- Step 2:** Check the number entered.
 - **Step 2a:** If the number entered is less than 2, then the robot says "Not enough laps." Then skip to Step 5.
 - **Step 2b:** If the number entered is greater than 200, then the robot says "Too many laps." Then skip to Step 5.
 - **Step 2c:** If the number entered is greater than or equal to 2 AND less than 100, then skip to Step 3.
 - **Step 2d:** If the number entered is between 100 and 200 (including 100 and 200), then skip to Step 4.
- Step 3:** The robot drives the number of laps entered. Skip to Step 5.
- Step 4:** The robot drives **half** of the number of laps entered. Go to Step 5.
- Step 5:** The program ends.

Lucia's algorithm

- Step 1:** Ask how many laps the robot should drive. Go to Step 2.
- Step 2:** Check the number entered.
 - **Step 2a:** If the number entered is less than 2, then the robot says "Not enough laps." Then skip to Step 4.
 - **Step 2b:** If the number entered is NOT less than 2, then move to Step 3.
- Step 3:** The robot drives the number of laps entered. Go to Step 4.
- Step 4:** The program ends.

9) In Gabriela's algorithm, what is shown on the screen if the number entered is 150?

- Robot says "Not enough laps."
- Robot says "Too many laps."
- Robot drives 150 laps.
- Robot drives 75 laps.

10) In Lucia's algorithm, what is shown on the screen if the number entered is 150?

- Robot says "Not enough laps."
- Robot says "Too many laps."
- Robot drives 150 laps.
- Robot drives 75 laps.

11) In Gabriela's algorithm, what is shown on the screen if the number entered is 2?

- Robot says "Not enough laps."
- Robot says "Too many laps."
- Robot drives 2 laps.
- Robot drives 1 laps.

12) In Lucia's algorithm, what is shown on the screen if the number entered is 2?

- Robot says "Not enough laps."
- Robot says "Too many laps."
- Robot drives 2 laps.
- Robot drives 1 laps.

Figure B.3

13) In Gabriela's algorithm, what is shown on the screen if the number entered is 250?

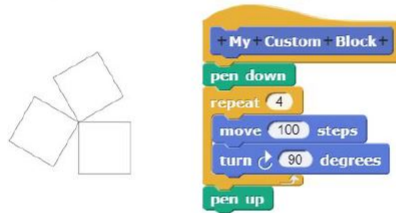
- Robot says "Not enough laps."
- Robot says "Too many laps."
- Robot drives 250 laps.
- Robot drives 125 laps.

14) In Lucia's algorithm, what is shown on the screen if the number entered is 250?

- Robot says "Not enough laps."
- Robot says "Too many laps."
- Robot drives 250 laps.
- Robot drives 125 laps.

Use this information to answer the next question

Which instructions should the sprite follow to draw the following design? Each side of each square measures 100 steps. You will also have this custom block to help you.



You may choose from the following options:

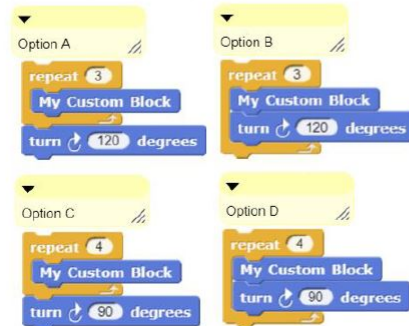


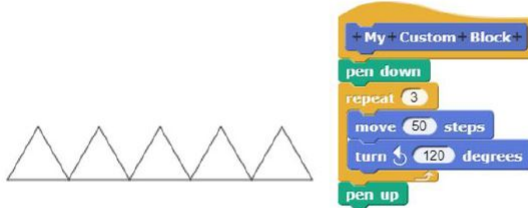
Figure B.4

15) Which option pictured above is correct?

- Option A
- Option B
- Option C
- Option D

Use this information to answer the next question

The instructions below should make the sprite draw the following design (starting from the left and ending on the right). Each side of each triangle measures 50 steps. You will also have this custom block to help you.



What is missing in the instructions? To put this another way, fill in the blank of this "repeat" loop with the number of times that the instruction needs to be repeated.



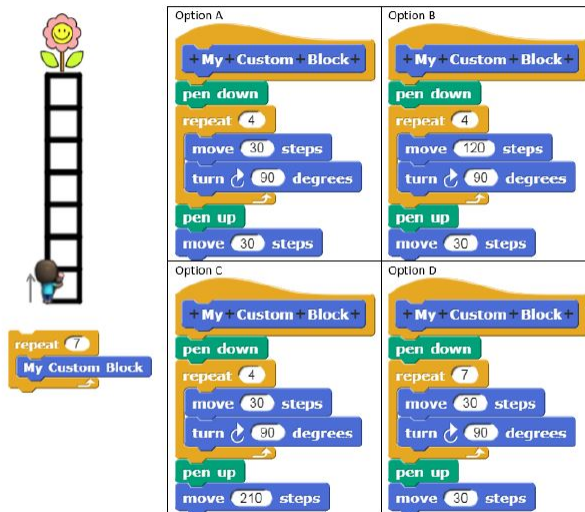
16) What number should go in the blank space above?

- 15
- 5
- 3
- 1

Figure B.5

Use this information to answer the next question

What custom block should the artist use to draw the ladder that reaches the flower? The custom block would go in place of "My Custom Block" in this "repeat" loop that is already provided. There are 30 steps between each rung of the ladder.

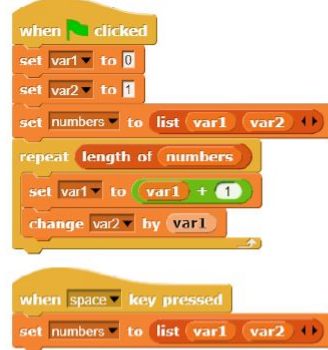


17) What option pictured above is correct?

- Option A
- Option B
- Option C
- Option D

Figure B.6

Use this program to help you with the following questions



```
when clicked
  set var1 to 0
  set var2 to 1
  set numbers to list (var1) (var2)
  repeat length of numbers
    set var1 to var1 + 1
    change var2 by var1
  end repeat
when space key pressed
  set numbers to list (var1) (var2)
```

-
- 18) How many times will the "repeat" loop be executed?
- 1
 - 2
 - 3
 - 4
-
- 19) What would the value of "var1" be after the green flag is clicked?
- 0
 - 1
 - 2
 - 3
 - 4
-
- 20) What would the value of "var2" be after the green flag is clicked?
- 0
 - 1
 - 2
 - 3
 - 4
-
- 21) The variable "numbers" is actually a list. What would the first item of the "numbers" list be after the green flag is clicked?
- 0
 - 1
 - 2
 - 3
 - 4

Figure B.7

22) What would the second item of the "numbers" list be after the green flag is clicked?

- 0
- 1
- 2
- 3
- 4

23) Which items of the "numbers" list would change after the spacebar is pressed?

- Neither of the items
- The first item
- The second item
- Both of the items

Use these rules to help you answer the remaining questions

You are programming a maze game.

- The screen shows a robot, maze walls, and gold coins.
- You move the robot around the screen, trying to find the coins and avoid the walls of the maze
- The maze walls and the coins stay in the same location every time you play the game.

- **You start with a score of 0.**
- **You have 5 lives at the start of each game.**
- If the robot runs over a coin:
 - You **gain** a life AND your score **increases** by 100.
- If the robot runs into a wall:
 - You **lose** a life AND your score **decreases** by 100.

The game ends in one of two ways:

- You reach a score of at least 500
- You have 0 lives remaining

If your score is 500 or more when the game ends, you win! If your score is less than 500 when the game ends, you lose.

24) Which of the following expressions correctly represent what happens when the robot runs over a gold coin? Select all that apply

- Change lives by 1
- Change lives by -1
- Change lives by 100
- Change lives by -100
- Change score by 1
- Change score by -1
- Change score by 100
- Change score by -100

Figure B.8

25) Which of the following expressions correctly represent what happens when the robot runs into a maze wall? Select all that apply

- Change lives by 1
- Change lives by -1
- Change lives by 100
- Change lives by -100
- Change score by 1
- Change score by -1
- Change score by 100
- Change score by -100

Which of the following conditions would determine if the game is over? It would go into the "if" block pictured here.



26) Choose your answer here:

- lives = 0
- score = 500
- lives > 0
- score > 499
- lives = 0 AND score = 500
- lives = 0 OR score = 500
- lives = 0 AND score > 499
- lives = 0 OR score > 499

Which of the following conditions would determine if you won the game after the game has already ended? It would go into the "if" block pictured here.



27) Choose your answer here:

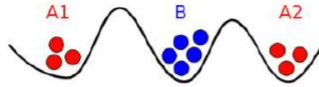
- lives = 0
- score = 500
- lives > 0
- score > 499
- lives = 0 AND score = 500
- lives = 0 OR score = 500
- lives = 0 AND score > 499
- lives = 0 OR score > 499

Figure B.9

B.2 Networking + Security Assessment

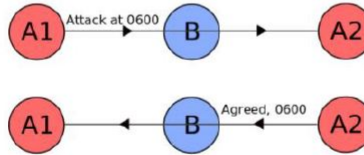
1) Enter your NetsBlox username here:

Two armies (A1 and A2) located on opposite sides of a city (B) are planning on attacking that city:



- These armies would like to agree on the time of their attack using messengers, but they are communicating unreliably as any messenger could be captured and may not deliver the message.
- For example, if A1 decides to attack at 6:00, it may send the message "We should attack at 6:00" to A2 but if the messenger gets captured, A2 will not attack at that time.
- If either army attacks alone, that army will be defeated - we are trying to prevent this.
- We will assume that the city cannot tamper with the content of the messages - the city will either stop the message from being sent or will not impact the message at all.

In order to solve this problem, suppose we require a single confirmation message. A1 will send a time to attack and wait for a confirmation from A2. Upon receiving the confirmation, A1 will attack at the proposed time. That is, A1 will attack if it receives an agreement message about the proposed time and A2 will attack if it receives a proposed attack time from A1:



2) Using this messaging technique, are the two armies guaranteed to attack at the same time?

- Yes
 No

3) If you answered "No", which army might attack alone? Select "N/A" if you answered "Yes".

- A1
 A2
 N/A

4) Explain your answers to the two previous questions

Figure B.10

The Caesar Cipher is an older (and since outdated) encryption technique used by Julius Caesar to "hide" messages he sent to his troops. Using the Caesar Cipher, each letter in the alphabet is translated to a new letter by a process known as shifting. For example, starting from the usual alphabet and shifting by 1 would result in the second row of letters here:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A

- Notice that for "Z" and other letters at the end of the alphabet, it is necessary to wrap around back to the start of the alphabet.
- As a practical example, if the message is "HI" and the shift value (also known as a key) is 2, then the encrypted message is "JK". This is because the letter "J" is two letters after "H" in the alphabet, and "K" is two letters after "I".
- It can also be helpful to keep in mind that a certain letter encrypted with a certain shift value will always be the same. For example, "A" will always become "B" with a shift value of 1.

5) Suppose that you want to encrypt the following message to communicate from army A2 back to army A1: ROBOTS. Of the following, which would be a correct encrypted message using Caesar Cipher?

- TODKVU
- URERWV
- SPCPUA
- VEFSXW

6) What is the core technology that allows digital devices to connect and transmit data with each other?

- Sensors
- Networks
- Smart Phones
- A Global Positioning Sensor (GPS)

7) Which of the following best explains what happens when a new device is connected to the internet?

- A device driver is assigned to the device.
- An Internet Protocol (IP) address is assigned to the device.
- A packet number is assigned to the device.
- A Web site is assigned to the device.

8) Which network topology requires a central controller or hub?

- Star
- Mesh
- Ring
- Bus

Figure B.11

Rank these network topologies based on how they would be impacted by a single failure between two nodes in the network.

	Most Impacted	Somewhat Impacted	Slightly Impacted	Least Impacted
9) Star	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10) Mesh	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
11) Ring	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
12) Bus	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

- 13) Which of these situations describes a computer network?
- An airport provides power outlets next to the seats in the boarding waiting areas. The outlets are constantly in use by passengers charging their laptops and smartphones
 - A fashion designer creates a glove that changes colors as your fingers move. The glove embeds flex sensors, a microcomputer, and LEDs. The embedded flex sensors are input sources for the microcomputer, relaying the current angle of a finger bend. The microcomputer then runs a program to determine the color and relays that to the LEDs
 - A programmer runs an algorithm on their computer that can play chess. The algorithm efficiently searches through a graph of all the possible moves and subsequently moves and picks the move with the highest probability of winning.
 - A video gaming club sets up six powerful computers in a room and connects them with Ethernet cables to a router. The club members play multi-player games together.

See the information provided below and answer the question:

The following table compares two data transport protocols from the suite of protocols powering the Internet.

Can it handle...	Protocol 1	Protocol 2
...lost packets?	Yes	No
...out of order packets?	Yes	No
...corrupt packets?	Yes	Yes

What are the most likely identities of Protocol 1 and Protocol 2?

- 14) Answer the question stated above
- Protocol 1 = User Datagram Protocol (UDP), Protocol 2 = Transmission Control Protocol (TCP)
 - Protocol 1 = TCP, Protocol 2 = Internet Routing Protocol (IP)
 - Protocol 1 = TCP, Protocol 2 = UDP
 - Protocol 1 = IP, Protocol 2 = UDP
- 15) Which of the following accurately describes the tradeoffs between the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP)?
- UDP can send a larger amount of data, but TCP is faster
 - UDP can guarantee the data is received in the correct order, but TCP is faster
 - TCP can send a larger amount of data, but UDP is faster
 - TCP can guarantee the data is received in the correct order, but UDP is faster

Figure B.12

-
- 16) Which of the following best describes symmetric encryption?
- An encryption scheme where the sender encrypts data with the receiver's public key and the receiver decrypts the data with their own private key
 - An encryption scheme where the sender and receiver use the same shared key for encrypting and decrypting data
 - An encryption scheme where the sender encrypts data with the sender's public key and the receiver decrypts the data with the sender's private key
 - An encryption scheme where the sender and receiver each have their own key for encrypting and decrypting data
-
- 17) What is a Distributed Denial of Service (DDoS) attack?
- An attempt by a government to deny Internet access to its citizens
 - An attempt to deny access to a website by flooding the website's servers with millions of requests from different computers.
 - An attempt by one user to deny service to another user by posting malicious material on a social network.
 - An attempt by an Internet user to access private information stored in a private database.
-
- Match the type of attack with the correct description of an attack**
- | | Eavesdropping | Man in the Middle | Replay | Brute Force |
|---|-----------------------|-----------------------|-----------------------|-----------------------|
| 18) Patricia intercepts the data being sent from her teacher to her principal, changing her grade from an "F" to an "A" | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| 19) Marco records the signal used to open a remote-controlled garage door, then sends that signal again to open the door himself | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| 20) Alice writes a program that can try all possible encryption values in an attempt to decrypt secure data and sell it to the highest bidder | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| 21) Charlie intercepts a private conversation sent from one of their friends to another without modifying the contents of the messages | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
-
- 22) Which of these best describes a Virtual Private Network (VPN)?
- A VPN is a protocol for establishing authenticated and encrypted links between networked computers
 - A VPN is the process of taking plain text and scrambling it into an unreadable format
 - A VPN is the technique to establish a protected network connection when using public networks
 - A VPN is built upon the Transmission Control Protocol (TCP) and allows packets of data to be transported across networks
-
- 23) What are some good rules to follow when creating a new password?

Figure B.13

B.3 Post-Survey

Self-Assessment

Page 1

Please complete the survey below.

Enter your NetsBlox Username here: _____

What is your gender identity? _____

Which of the following ethnic or racial categories best describes how you self-identify? Select all that apply

<input type="checkbox"/>	Pacific Islander/Native Hawaiian
<input type="checkbox"/>	Native American/Alaska Native/First Nations
<input type="checkbox"/>	Asian
<input type="checkbox"/>	Black or African-American
<input type="checkbox"/>	White
<input type="checkbox"/>	Hispanic/Latinx
<input type="checkbox"/>	Middle Eastern/North African
<input type="checkbox"/>	Other
<input type="checkbox"/>	Prefer Not to Say

Have you used robots in the past (before this class) in some way? Yes No

How did the robots we used in this class compare to that previous experience? What approach did you prefer? Why? _____

Have you used programming languages other than NetsBlox in the past (before this class)? Yes No

How did NetsBlox compare to that previous experience? What approach did you prefer? Why? _____

What did you like about NetsBlox? _____

What did you dislike about NetsBlox? _____

What did you like about the robots? _____

What did you dislike about the robots? _____

Figure B.14

For each row, please choose the column that most accurately describes your current attitudes and feelings. If the question is not applicable, select "Undecided". Thank you!					
	Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree
The things I learn in school help me in my daily life outside of school.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Being good at computer science is an important part of who I am.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It is important to me to understand cybersecurity.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I think the things I learned in this class were useful.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
One of my goals is to master a lot of new skills over next school year.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It's important to me that I thoroughly understand my coding class work.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Even if my school work is hard, I can learn and apply the concepts successfully.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I can do almost all the work in my coding classes if I don't give up.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Even if the work in my coding classes is hard, I can learn it	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It's important to me that other students in my classes think I am good at my classwork.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
One of my goals is to look smart in comparison to the other students in my coding classes.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I force myself to finish my class work even when there are other things I'd rather be doing.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Even if I don't see the importance of a particular coding assignment, I still complete it.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It's important that I don't look stupid in class.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
One of my goals in my coding classes is to avoid looking like I have trouble doing the work.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure B.15

When I do school work, I check over my work for mistakes.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I ask myself questions to make sure I understand the coding material I've been learning or applying.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
When I become confused about something I'm learning in my coding classes, I go back and try to figure it out.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I prefer working as part of a team to working alone.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I enjoy seeing my teammates be successful.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I find that teams make better decisions than individuals.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I find that teamwork raises my own efficiency.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I learned a lot about cybersecurity in this class.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I learned a lot about networking in this class.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I learned a lot about computer science in this class.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Regardless of whether or not you have actually tried it, how interested are you in...				
	Very interested	Interested	A little interested	Not at all interested
Computer networking (For example, the internet)?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Thinking of new ways to apply computer science (For example, new apps or software)?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Programming computers to create new apps (in other words, writing code)?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Finding technological solutions to world problems using computer science?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Designing computer games?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Learning computer science concepts?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Learning coding concepts?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure B.16

Page 4

Right now, how confident are you in your ability to...				
	Very confident	Confident	A little confident	Not at all confident
Do computer networking (For example, the internet)?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Think of new ways to apply computer science (For example, new apps or software)?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Program computers to create new apps (in other words, writing code)?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Find technological solutions to world problems using computer science?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Design computer games?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Learn computer science concepts?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Learn coding concepts?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure B.17

Appendix C

Appendix C - Module Examples

C.1 Module 2

This assessment would accompany the second module, which makes use of the square art project to teach loops, variables, and custom blocks:

1. Complete the empty spaces in the code below for the block `Draw Polygon`. Make sure that it draws a regular polygon with a length of each side equal to `length` and a number of sides equal to `sides`. It should start drawing once the block begins and stop drawing after the block completes its script.

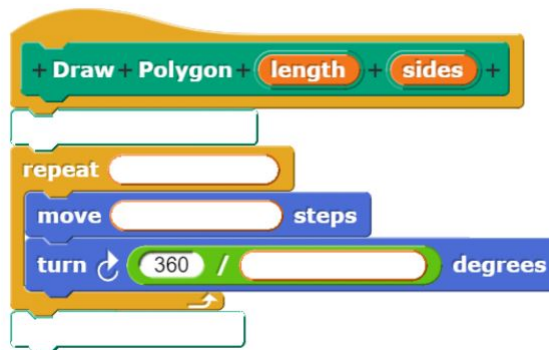


Figure C.1

2. Fill in the inputs to `Draw Polygon` below to draw the figure to the right. The first input is the length of each side and the second input is the number of sides.



3. Draw the polygon that `Draw Polygon` will produce with the given input values. The first input is the length of each side and the second input is the number of sides.



Figure C.2

Consider the following script that uses Draw Polygon with the first input as the length of each side and the second input as the number of sides.

```

when clicked
clear
hide
set pen color to red
go to x: 0 y: -80
point in direction 90
repeat 72
change pen hue by 1
Draw Polygon 100 5
move 5 steps
turn 5 degrees
  
```

Figure C.3

4. What pattern will this script draw when the green flag is clicked?
- A. 72 pentagons nested inside of each other
 - B. 72 squares nested inside of each other
 - C. A circle of 72 squares
 - D. A circle of 72 pentagons


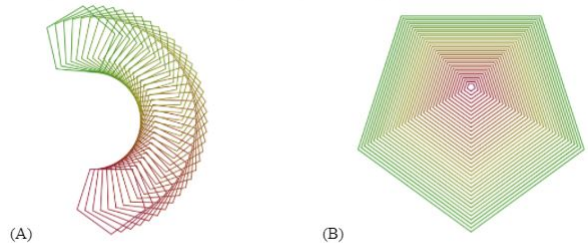
5. What will happen to the pattern drawn by the script if the block  is removed?
- A. It will draw the same pattern, but all shapes will have the same color.
 - B. It will draw the same pattern, but all shapes will have thicker lines.
 - C. It will draw a different pattern, but each shape will have a different color.
 - D. It will draw a different pattern, and all shapes will have the same color.

Figure C.4

6. Which pattern will be drawn if the input to repeat is changed to 36?

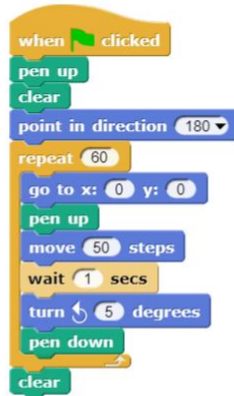


- (C) It will draw the same pattern as the one drawn in problem 4 but with smaller polygons.
- (D) It will not draw a pattern at all.

Figure C.5

Debugging Task - A Broken Clock

There is a saying that even a broken clock is correct twice a day (I guess only once if it's a broken 24-hour clock, like military time). But a functioning clock would be more useful anyway. You are attempting to build a script that makes the second hand of a clock tick around one full revolution every 60 seconds. Unfortunately, what your initial code looks like is this:



You will also have these blocks provided to help you fix the code:

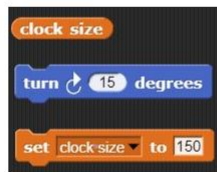


Figure C.6: This Debugging task was included with Module 2 (Part 1 of 2)

Download [that code from here](#).

Run the code and see what it actually does. Discuss with your workstation neighbor what you think is making the clock tick so weirdly.

Reorder the blocks so that the clock works in about the same way as the video. You may also need to replace one or two blocks with the additional blocks already placed to the side of this script in the downloaded code.

Here are some **guiding questions** to help you think through the possible bugs - you and your partner should write down answers to these:

1. Does what is happening with the pen in the video match up with what is happening in this project? If not, why?
2. Does the turning of the sprite in the video match up with what is happening in this project? If not, what is different?
3. Does the sprite start in the right location? If not, where does it start?
4. How can we control the size of the clock?

Figure C.7: This Debugging task was included with Module 2 (Part 2 of 2)


This survey would also accompany the debugging task:

- Does what is happening with the pen in the video match up with what is happening in this project? If not, why?
- Does the turning of the sprite in the video match up with what is happening in this project? If not, what is different?
- Does the sprite start in the right location? If not, where does it start?
- How can we control the size of the clock?

Drawing Build

Now that you have a wealth of artistic skills to employ within Netsblox, you will **draw the following art**:

- A set of 5 nested pentagons (5 sides). Each pentagon must be a different color than the other pentagons (5 pts.).
- A circular wheel of squares (think of button 2 in the Added Features) (5 pts.) that all have the same side length as the smallest pentagon in the nested pentagons (5 pts.).
- A hexagon (6 sides) that is twice as big as the squares or smallest pentagon (5 pts.) that also has lines that are at least 10 in size and colored blue (5 pts.).

All three sets of shapes should start drawing when the green flag  is clicked. They should not be directly on top of each other. All shapes must be drawn using the **Draw Polygon** block with appropriate inputs (5 pts.)

Save and Share

Save your drawing to your cloud.

In the file menu, **click "Save as...."** In the pop-window, click on the game file you just saved, and then click "Share." It will ask you if you are sure you want to publish your project. Choose "Yes."

Submit

Right-click on the green flag script you created and choose "script pic." Save the pic as "Drawing".

Upload your screenshot to the following assignment "2.4.2A Drawing Build." Copy and paste the URL (the web address) for your shared code to the comment option.

Submit your script pic and URL comment.

Figure C.8: This is the project task associated with Module 2

```

when green flag clicked
pen up
clear
hide
set pen size to 1
go to x: 60 y: 75
point in direction 90
set length to 15
set difference to 10
repeat 5
  draw polygon 5 length
  change length by difference
  change x by (-1 * difference / 2)
  change y by (1 * difference / 2)
  change pen hue by 30
go to x: -100 y: -50
point in direction 90
repeat 72
  set pen color to purple
  draw polygon 4 15
  move 5 steps
  turn 5 degrees
go to x: 30 y: -40
set pen size to 10
set pen color to blue
pen down
draw polygon 6 73

```

Figure C.9: This is an example of a strong implementation of the Module 2 project

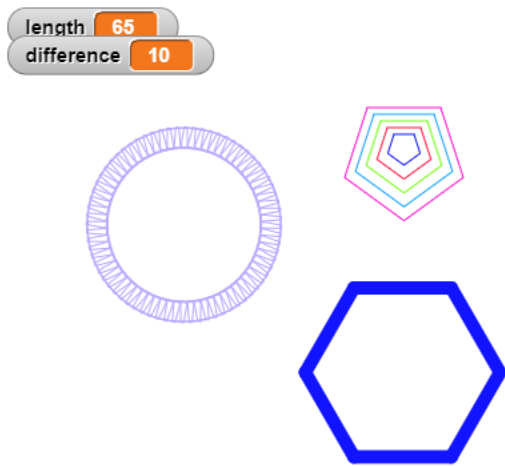


Figure C.10: This is the stage output of a strong implementation of the Module 2 project

```

when clicked
pen up
clear
hide
go to x: 0 y: -60
point in direction 90
repeat 72
  set pen hue to 100
  draw polygon 4 35
  move 5 steps
  turn 5 degrees
move 5 steps
turn 5 degrees
pen up
hide
go to x: 0 y: 5
point in direction 90
set length to 20
set difference to 10
repeat 5
  draw polygon 5 20
  change pen hue by 100
  change length by difference
  change x by  $-1 * \text{difference} / 2$ 
  change y by  $-1 * \text{difference} / 2$ 
go to x: 0 y: 0
draw polygon 6 100

```

Figure C.11: This is an example of a weaker implementation of the Module 2 project

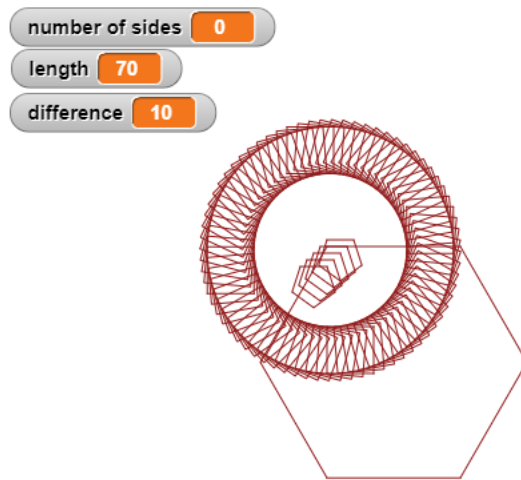


Figure C.12: This is the stage output of a weaker implementation of the Module 2 project

C.2 Module 3

Conditionals

Make sure you are starting from the template or your "Music 3-2-6" project, and that you have your headphones on.

Introducing Conditionals

If you've been scrolling through the "Control" category in Netsblox or looking into some of the pre-existing code on the game projects, you've probably noticed **if** blocks already:



Basically, we'll check if something is true or not - an example could be **if 2 > 1**, which looks like this:



In this case, the condition is true, so we would run any other blocks that are inside the **if** statement. If the condition was instead false, we would not run the blocks.

These are joined by **if else** blocks as the primary conditionals we will be using:



The difference here is we can include code that we want to run if the condition is false. For example, if we compared **2 < 1** (instead of the **2 > 1** from earlier), we would run only the blocks in the **else** part of the conditional:

Figure C.13: This is an example student-facing document for Module 3 (Part 1 of 2)

```
if 2 < 1
else
```

Applying Conditionals

Let's see how we can apply these ideas to the music program:

```
set music to list D D D rest E G G
set beats to list 0.5 0.5 0.5 0.5 0.5 0.5 0.5
for i = 1 to length of music
  if item i of music = rest
    rest for item i of beats beats
  else
    play note item i of music for item i of beats beats
```

We expand to 3 notes followed by a rest followed by 3 more notes to show off the new features.

Then, for each of the **items** of the **music** list, we check to see if the word "rest" was entered in. If it was, **the music will pause for the number of beats that were typed in.**

If the word "rest" was not entered, then we assume that a note was entered instead, and we play that note.

Try this for yourself to make sure it works as you expect.

Save Your Code

Save your code to your cloud as "Music 3-2-7"

Figure C.14: This is an example student-facing document for Module 3 (Part 2 of 2)

C.3 Module 4

Chat Build

You are now tasked with operating the server for a messaging app. You will be managing messages between two clients: Monte and Mr. M.

Open [this starter code](#).

Save it as "4-4-3 Chat Build Server". Save it exactly like that.

Connecting

When the chatters first try to **connect**, the code should first check that they are not already connected. If the **address** is already in the list **clients**, your script must send a **chat** message that says the first error message from: " is already logged in to the chat. Send a message". Otherwise, the server should manage connecting and send a **chat** message that says the confirmation message: ", you have successfully connected". (10 points)

connect now has two fields: **address** and **handle**. The **address** information will contain each chatter's Public Role ID. The **handle** will contain the name of the chatter. Both of these need to be stored in respective **lists** called **clients** (for **address**) and **handles** (for **handle**). (10 points)

Messaging

When the connected clients send messages, they will come in as a **chat** type message, which has fields **sender** and **text**. **sender** contains the Public Role ID, and **text** contains the message you will relay to every member of the **clients** list.

If the **sender** is already on the **clients** list, meaning they already connected, then you will send that same **sender** and **text** to each **item** in the **client** list as a **chat** type message. (5 points)

If the **sender** is not already on the list, your script must send the second error message from " is not connected to the chat. Please connect" as a **chat** type message. (5 points)

Figure C.15: This is the project task associated with Module 4



Figure C.16: This is an example of a strong implementation of the Module 4 project

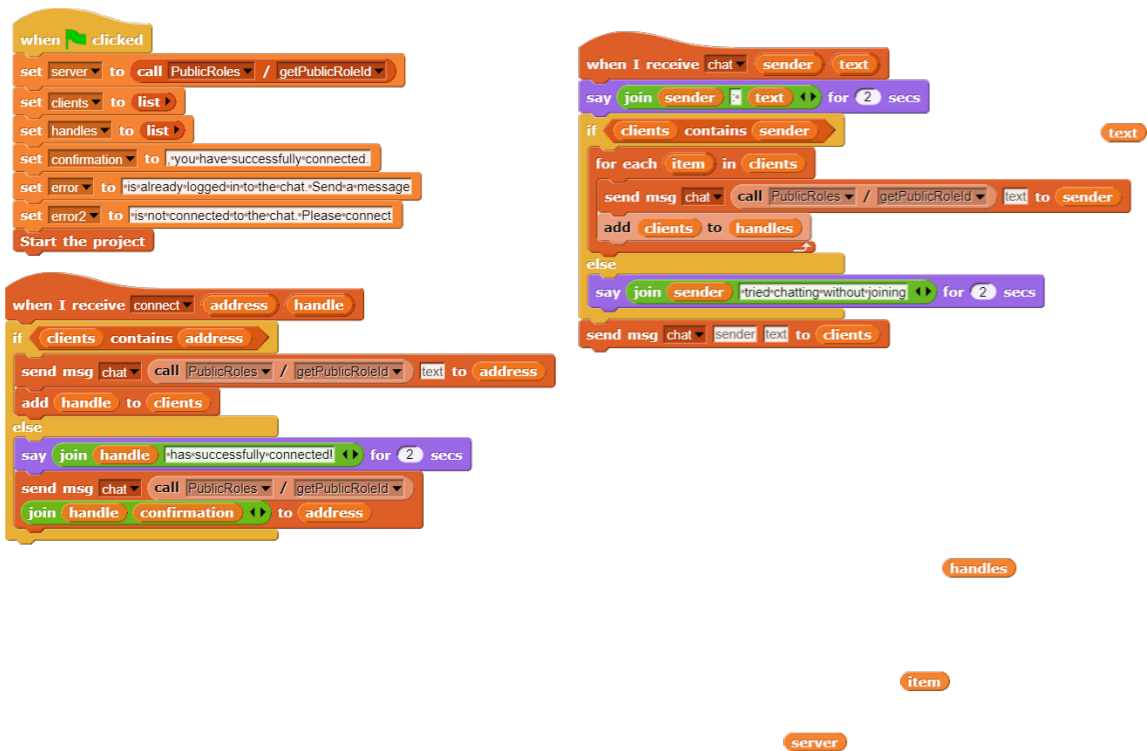


Figure C.17: This is an example of a weaker implementation of the Module 4 project

Appendix D

Appendix D - Additional Results

Table D.1 Assessment performance over time broken down by key concepts. Results are presented as “Mean (SD) / Max”. “N/A” indicates that the concept was not present during that particular assessment. This table contains the pre-test through module 3 - see Table D.2 for the remaining results.

Concept	Pre-Test	Module 1	Module 2	Module 3
Variables	3.56 (1.55) / 9	N/A	2.57 (1.17) / 5	5.3 (2.01) / 10
Conditionals	9.79 (2.08) / 14	N/A	N/A	1.85 (0.75) / 3
Loops	2.48 (1.19) / 7	N/A	5.98 (1.58) / 9	3 (1.35) / 6
Encryption	0.5 (0.65) / 2	N/A	N/A	N/A
Messaging	0.75 (0.83) / 2	N/A	N/A	N/A
Topology	1.4 (0.99) / 5	N/A	N/A	N/A
Evaluation	0.94 (0.69) / 2	0.58 (0.49) / 1	4 (1.12) / 6	N/A
Testing & debugging	0.75 (0.43) / 1	0.79 (0.76) / 2	N/A	N/A
Total	24.33 (5.22) / 46	7.09 (1.69) / 10	6.47 (1.69) / 10	5.3 (2.01) / 10

Table D.2 Assessment performance over time broken down by key concepts. Results are presented as “Mean (SD) / Max”. “N/A” indicates that the concept was not present during that particular assessment. This table contains module 4 through the post-test - see Table D.1 for the initial results.

Concept	Module 4	Module 5	Class Final	Post-Test
Variables	3.28 (1.13) / 5	1.54 (0.76) / 3	11.25 (2.12) / 14	3.85 (1.46) / 9
Conditionals	N/A	N/A	2.6 (0.64) / 3	9.98 (2.14) / 14
Loops	N/A	1.54 (0.76) / 3	9.35 (2.05) / 12	3.79 (1.77) / 7
Encryption	N/A	1.09 (0.78) / 2	1.06 (0.97) / 4	0.54 (0.58) / 2
Messaging	6.09 (1.27) / 8	0.52 (0.5) / 1	6.04 (1.62) / 8	0.9 (0.85) / 2
Topology	N/A	2.86 (1.02) / 4	2.25 (0.9) / 3	2.85 (1.68) / 5
Evaluation	N/A	1.78 (0.47) / 2	2.63 (0.9) / 4	1.46 (0.64) / 2
Testing & debugging	N/A	N/A	0.56 (0.5) / 1	0.67 (0.47) / 1
Total	7.84 (1.52) / 10	17.47 (4.4) / 25	26.19 (4.52) / 36	29.6 (6.74) / 46

Table D.3 Spearman correlations are displayed as one measure of the relationship between a student’s grade (7th or 8th), gender (male or *female*), or race (only listing those with $\geq 16.67\%$ of the student population) and their performance throughout the remainder of the intervention. Positive correlations ≥ 0.4 are in bold and shown in blue text, while negative correlations ≤ 0.4 are in bold and shown in red text. Each column corresponds to a separate grade, gender, or race, and each row corresponds to a metric from the intervention.

Metric	Grade	Gender	Asian	Black/African-American	White
Pre-Test Total	0.12	-0.07	0.15	-0.36	0.3
Pre-Test CT	0.26	0.14	0.12	-0.17	0.16
Pre-Test Networking	-0.13	-0.13	0.21	-0.29	0.06
Pre-Test Security	0	-0.25	0.03	-0.33	0.4
Module 1 Average	0.31	-0.24	-0.06	-0.32	0.27
Module 2 Average	0.29	-0.34	0.21	-0.37	0.31
Module 3 Average	0.25	-0.19	-0.04	-0.18	0.19
Module 4 Average	0.26	-0.26	0.07	-0.22	0.15
Module 5 Average	-0.08	-0.4	0.25	-0.56	0.26
Module 1 Project	0.1	-0.39	0.11	-0.37	0.37
Module 2 Project	0.18	-0.32	-0.08	-0.39	0.3
Module 3 Project	0.39	-0.03	0.08	-0.47	0.36
Module 4 Project	0.39	0.06	0.11	-0.38	0.15
Class Final Total	0.26	-0.33	0.15	-0.33	0.26
Class Final CT	0.3	-0.27	0.1	-0.31	0.25
Class Final Networking	0.27	-0.26	0.18	-0.22	0.13
Class Final Security	0.06	-0.33	0.12	-0.36	0.29
Post-Test Total	0.07	-0.09	0.1	-0.37	0.32
Post-Test CT	0.23	-0.03	-0.01	-0.31	0.18
Post-Test Networking	-0.13	-0.17	0.15	-0.37	0.34
Post-Test Security	-0.03	-0.17	-0.01	-0.3	0.35

Table D.4 Spearman correlations are displayed as one measure of relationship between a student’s grade (7th or 8th), gender (male or *female*), or race (only listing those with $\geq 16.67\%$ of the student population) and their attitudes (see Table 5.6), actions, and attendance. “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. Negative correlations ≤ 0.4 (or in the case of Tardies, a positive correlation that indicated more tardies and is generally considered to be a negative situation) are in bold and shown in red text. Each column corresponds to a separate grade, gender, or race, and each row corresponds to a metric from the intervention.

Metric	Grade	Gender	Asian	Black/African-American	White
General TV - Pre	-0.11	0.2	-0.13	0.14	-0.06
Specific TV - Pre	0.01	-0.37	-0.21	-0.23	0.28
General MA - Pre	0.16	0.11	-0.29	0	0.31
Specific MA - Pre	0.03	-0.36	0.16	-0.39	0.14
General PC - Pre	-0.37	0.03	-0.11	-0.14	0.14
Specific PC - Pre	-0.19	-0.1	-0.27	-0.21	0.12
General AP - Pre	0.09	-0.09	0.12	-0.15	0.03
Specific AP - Pre	0.07	0.04	0.19	-0.09	0.04
General BE - Pre	-0.32	0.03	0	-0.19	0.28
Specific BE - Pre	-0.28	0.07	-0.29	-0.09	0.34
General AV - Pre	-0.21	-0.02	0.01	-0.05	0.2
Specific AV - Pre	-0.06	-0.05	0.21	-0.14	0.11
General CE - Pre	-0.12	0.03	0.01	-0.07	-0.04
Specific CE - Pre	-0.23	-0.28	0.08	-0.3	0.01
Specific CI - Pre	0.12	-0.39	-0.19	-0.22	0.36
Specific CC - Pre	0.13	-0.1	-0.05	-0.17	0.21
Collaboration - Pre	-0.07	-0.19	0.09	0.07	-0.15
Module 1 Actions	-0.43	-0.18	-0.24	-0.02	0.19
Module 2 Actions	-0.07	-0.19	-0.05	-0.2	0.17
Module 3 Actions	0.04	-0.28	0.13	-0.3	0.25
Module 4 Actions	0.25	-0.21	0.05	-0.22	0.15
Module 5 Actions	0.01	-0.3	-0.06	-0.33	0.36
Total Actions	-0.02	-0.33	0	-0.34	0.34
Total Absences	0.34	0.29	0.03	0.32	-0.19
Total Tardies	0.05	0.45	-0.02	0.56	-0.38
General TV - Post	0.15	-0.15	-0.24	-0.06	0.18
Specific TV - Post	0.26	-0.43	-0.16	-0.13	0.32
General MA - Post	0.13	-0.04	-0.23	-0.01	0.19
Specific MA - Post	0.06	-0.32	-0.02	-0.21	0.31
General PC - Post	-0.05	-0.04	-0.07	0.11	-0.08
Specific PC - Post	0.08	-0.15	-0.25	0.02	0.27
General AP - Post	0.04	-0.03	0.12	-0.05	0.2
Specific AP - Post	-0.09	0.22	0.19	-0.07	0.12
General BE - Post	0	0.22	0.12	-0.17	0.18
Specific BE - Post	0.06	-0.12	0.03	-0.08	0.15
General AV - Post	-0.2	0.06	0.15	-0.01	0.04
Specific AV - Post	-0.03	0.16	0.16	0.1	0.07
General CE - Post	-0.13	0.07	-0.22	-0.03	0.19
Specific CE - Post	0.2	0.06	0.03	-0.15	0.19
Learning - Post	0.03	-0.16	-0.14	-0.13	0.28
Specific CI - Post	0.15	-0.24	-0.15	-0.11	0.34
Specific CC - Post	-0.18	-0.18	-0.03	-0.05	0.18
Collaboration - Post	0.09	-0.16	0.01	-0.13	0.17

Table D.5 Spearman correlations are displayed as one measure of the relationship between a student’s pre-test performance and their post-test attitudes (see Table 5.6). “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. Each column corresponds to separate component of the pre-test and each row corresponds to a post-survey result.

Metric	Pre-Test Total	Pre-Test CT	Pre-Test Net-working	Pre-Test Security
General TV - Post	0.1	0.07	0.22	0
Specific TV - Post	0.21	0.07	0.04	0.32
General MA - Post	0.09	0.02	0.03	0.16
Specific MA - Post	0.19	0.05	0.04	0.28
General PC - Post	-0.17	-0.17	-0.11	-0.02
Specific PC - Post	0.15	0.18	-0.08	0.15
General AP - Post	0.14	0.02	-0.03	0.26
Specific AP - Post	0.17	0.12	0.04	0.13
General AV - Post	0.2	0.09	0.25	0.17
Specific AV - Post	0.2	0.19	0	0.16
General CE - Post	0.24	0.16	0.04	0.3
Specific CE - Post	0.12	0.13	0.14	0
Learning - Post	0.3	0.21	0.21	0.26
Specific CI - Post	0.08	0.09	-0.16	0.13
Specific CC - Post	0.21	0.05	0.05	0.33
Collaboration - Post	-0.03	-0.1	0	0.05

Table D.6 Spearman correlations are displayed as one measure of the relationship between a student’s pre-survey responses (see Table 5.6) and their performance throughout the remainder of the intervention. Only selected survey categories with at least one relevant correlation $\geq |0.4|$ are presented for the sake of space. “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. Positive correlations ≥ 0.4 are in bold and shown in blue text. Each column corresponds to separate component of the pre-survey and each row corresponds to a metric from throughout the intervention.

Metric	Specific MA - Pre	Specific PC - Pre	General BE - Pre
Module 1 Average	0.26	0.24	-0.02
Module 2 Average	0.15	0.2	0.05
Module 3 Average	0.12	0.06	0.1
Module 4 Average	0.34	0.25	0.12
Module 5 Average	0.33	0.23	0.27
Class Final Total	0.24	0.16	-0.04
Class Final CT	0.3	0.23	-0.04
Class Final Networking	0.1	0.15	-0.09
Class Final Security	0.28	0.12	0.06

Table D.7 Spearman correlations are displayed as one measure of the relationship between a student’s pre-survey performance and their attitudes (see Table 5.6). “General” refers to attitudes towards school in general, while “Specific” refers to attitudes towards this course or STEM specifically. Each column corresponds to a separate pre-survey category, and each row corresponds to a metric from later in the intervention.

Metric	Specific MA - Pre	General BE - Pre	General AV - Pre	Specific AV - Pre	Specific CI - Pre	Specific CC - Pre
General PC - Post	0.25	0.01	-0.02	0.01	0.14	0.39
Specific PC - Post	0.13	0.12	0	-0.16	0.27	0.33
General CE - Post	0.18	0.3	0.1	0	0.06	0.21
Specific CE - Post	0.25	0.13	-0.03	0.07	0.24	0.19
Learning - Post	0.27	0.2	0.05	-0.09	0.23	0.29
Specific CC - Post	0.09	0.2	0.27	0.23	0.24	0.27
Collaboration - Post	0.34	0.25	0.13	0.09	0.17	0.14

Table D.8 Spearman correlations are displayed as one measure of the relationship between a student’s early CT performance and their performance and participation during networking and security modules and assessments. Each column corresponds to a separate CT-related assessment, and each row corresponds to a metric from later in the intervention. This early CT performance is based on the CT components of Module 2 and 3 assessments. Positive correlations ≥ 0.4 are in bold and shown in blue text.

Metric	Module 2 CT	Module 3 CT
Module 4 Actions	0.19	0.44
Module 4 Networking	0.45	0.5
Module 4 Project	0.3	0.32
Module 5 Actions	0.3	0.42
Module 5 Networking	0.53	0.5
Module 5 Security	0.2	0.45
Justin Final Networking	0.39	0.37
Justin Final Security	0.29	0.32
Post-Test Networking	0.38	0.26
Post-Test Security	0.46	0.25

References

- [act, 2021] (2021). ActivityBot 360. <https://www.parallax.com/activitybot-360/>. Cited 2021 September 19.
- [Afari and Khine, 2017] Afari, E. and Khine, M. (2017). Robotics as an educational tool: impact of lego mindstorms. *International Journal of Information and Education Technology*, 7(6):437–442.
- [Akoglu, 2018] Akoglu, H. (2018). User’s guide to correlation coefficients. *Turkish journal of emergency medicine*, 18(3):91–93.
- [Al-Jarrah and Pontelli, 2014] Al-Jarrah, A. and Pontelli, E. (2014). ”alice-village” alice as a collaborative virtual learning environment. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–9. IEEE.
- [Amo et al., 2021] Amo, D., Fox, P., Fonseca, D., and Poyatos, C. (2021). Systematic review on which analytics and learning methodologies are applied in primary and secondary education in the learning of robotics sensors. *Sensors*, 21(1):153.
- [Anderson et al., 2015] Anderson, B. R., Nauer, K. S., Lee, W. K., McClain, J. T., and Abbott, R. G. (2015). Tracer fire cyberforensic training platform. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
- [Anderson et al., 1996] Anderson, J. R., Reder, L. M., and Simon, H. A. (1996). Situated learning and education. *Educational researcher*, 25(4):5–11.
- [Angeli and Valanides, 2020] Angeli, C. and Valanides, N. (2020). Developing young children’s computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, 105:105954.
- [Araujo et al., 2019] Araujo, A. L. S. O., Andrade, W. L., Guerrero, D. D. S., and Melo, M. R. A. (2019). How many abilities can we measure in computational thinking? a study on bebras challenge. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 545–551.
- [Arhippainen et al., 2011] Arhippainen, L., Pakanen, M., Hickey, S., and Mattila, P. (2011). User experiences of 3d virtual learning environment. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 222–227.
- [Armoni and Gal-Ezer, 2014] Armoni, M. and Gal-Ezer, J. (2014). Early computing education: why? what? when? who? *ACM Inroads*, 5(4):54–59.
- [Ayala et al., 2020] Ayala, A., Cruz, F., Campos, D., Rubio, R., Fernandes, B., and Dazeley, R. (2020). A comparison of humanoid robot simulators: A quantitative approach. In *2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 1–6. IEEE.
- [Barak and Zadok, 2009] Barak, M. and Zadok, Y. (2009). Robotics projects and learning concepts in science, technology and problem solving. *International Journal of Technology and Design Education*, 19(3):289–307.
- [Barr and Stephenson, 2011] Barr, V. and 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.
- [Barth-Cohen et al., 2018] Barth-Cohen, L. A., Jiang, S., Shen, J., Chen, G., and Eltoukhy, M. (2018). Interpreting and navigating multiple representations for computational thinking in a robotics programming environment. *Journal for STEM Education Research*, 1(1-2):119–147.
- [Basawapatna et al., 2015] Basawapatna, A. R., Repenning, A., and Koh, K. H. (2015). Closing the cyber-learning loop: Enabling teachers to formatively assess student programming projects. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 12–17.

- [Basu, 2019] Basu, S. (2019). Using rubrics integrating design and coding to assess middle school students' open-ended block-based programming projects. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 1211–1217.
- [Belland, 2017] Belland, B. R. (2017). *Instructional scaffolding in STEM education: Strategies and efficacy evidence*. Springer Nature.
- [Belland et al., 2017] Belland, B. R., Walker, A. E., Kim, N. J., and Lefler, M. (2017). Synthesizing results from empirical research on computer-based scaffolding in stem education: A meta-analysis. *Review of Educational Research*, 87(2):309–344.
- [Benitti, 2012] Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3):978–988.
- [Bergman et al., 2003] Bergman, L. R., Magnusson, D., and El Khouri, B. M. (2003). *Studying individual development in an interindividual context: A person-oriented approach*, volume 4. Psychology Press.
- [Berland et al., 2014] Berland, M., Baker, R. S., and Blikstein, P. (2014). Educational data mining and learning analytics: Applications to constructionist research. *Technology, Knowledge and Learning*, 19(1-2):205–220.
- [Berland et al., 2013] Berland, M., Martin, T., Benton, T., Petrick Smith, C., and Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, 22(4):564–599.
- [Biswas et al., 2005] Biswas, G., Leelawong, K., Schwartz, D., Vye, N., and at Vanderbilt, T. T. A. G. (2005). Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence*, 19(3-4):363–392.
- [Biswas et al., 2016] Biswas, G., Segedy, J. R., and Bunchongchit, K. (2016). From design to implementation to practice a learning by teaching system: Betty's brain. *International Journal of Artificial Intelligence in Education*, 26(1):350–364.
- [Biswas and Sulcer, 2010] Biswas, G. and Sulcer, B. (2010). Visual exploratory data analysis methods to characterize student progress in intelligent learning environments. In *2010 International Conference on Technology for Education*, pages 114–121. IEEE.
- [Blikstein, 2011] Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st international conference on learning analytics and knowledge*, pages 110–116.
- [Blikstein, 2013] Blikstein, P. (2013). Multimodal learning analytics. In *Proceedings of the third international conference on learning analytics and knowledge*, pages 102–106.
- [Brady et al., 2020] Brady, C., Gresalfi, M., Steinberg, S., and Knowe, M. (2020). Debugging for art's sake: Beginning programmers' debugging activity in an expressive coding context.
- [Braught et al., 2010] Braught, G., MacCormick, J., and Wahls, T. (2010). The benefits of pairing by ability. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 249–253.
- [Broll et al., 2017] Broll, B., Lédeczi, A., Volgyesi, P., Sallai, J., Maroti, M., Carrillo, A., Weeden-Wright, S. L., Vanags, C., Swartz, J. D., and Lu, M. (2017). A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 81–86.
- [Buitrago Flórez et al., 2017] Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., and Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87(4):834–860.

- [Burlleson et al., 2017] Burlleson, W. S., Harlow, D. B., Nilsen, K. J., Perlin, K., Freed, N., Jensen, C. N., Lahey, B., Lu, P., and Muldner, K. (2017). Active learning environments with robotic tangibles: Children’s physical and virtual spatial programming experiences. *IEEE Transactions on Learning Technologies*, 11(1):96–106.
- [Cabada et al., 2018] Cabada, R. Z., Estrada, M. L. B., Hernández, F. G., Bustillos, R. O., and Reyes-García, C. A. (2018). An affective and web 3.0-based learning environment for a programming language. *Telematics and Informatics*, 35(3):611–628.
- [Cabo, 2019] Cabo, C. (2019). Student progress in learning computer programming: Insights from association analysis. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE.
- [Catlin and Blamires, 2010] Catlin, D. and Blamires, M. (2010). The principles of educational robotic applications (era). In *Constructionism Conference, Paris*.
- [Catlin and Woollard, 2014] Catlin, D. and Woollard, J. (2014). Educational robots and computational thinking. In *Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education*, pages 144–151.
- [Chattopadhyay et al., 2020] Chattopadhyay, A., Azhar, M. Q., Everson, T., and Ruska Jr, R. (2020). Integrated cybersecurity plus robotics lesson using nao. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 397–402.
- [Chen, 2003] Chen, C. (2003). A constructivist approach to teaching: Implications in teaching computer networking implications in teaching computer networking. *Information Technology, Learning, and Performance Journal*, 21(2):17.
- [Chou and Liu, 2005] Chou, S.-W. and Liu, C.-H. (2005). Learning effectiveness in a web-based virtual learning environment: a learner control perspective. *Journal of computer assisted learning*, 21(1):65–76.
- [Clow, 2013] Clow, D. (2013). An overview of learning analytics. *Teaching in Higher Education*, 18(6):683–695.
- [Coffman-Wolph and Gray, 2019] Coffman-Wolph, S. and Gray, K. (2019). Computer security activities for a middle school classroom or outreach event (p12 resource/curriculum exchange). In *2019 ASEE Annual Conference & Exposition*.
- [Cohen, 1960] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.
- [Cohen, 1992] Cohen, J. (1992). Statistical power analysis. *Current directions in psychological science*, 1(3):98–101.
- [Cooper et al., 2000] Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3-d tool for introductory programming concepts. *Journal of computing sciences in colleges*, 15(5):107–116.
- [Crumpler and Lewis, 2019] Crumpler, W. and Lewis, J. A. (2019). The Cybersecurity Workforce Gap. *Center for Strategic and International Studies (CSIS)*, (January):1–10.
- [Curricula, 2017] Curricula, C. (2017). Curriculum guidelines for post-secondary degree programs in cybersecurity. *A Report in the Computing Curricula Series Joint Task Force on Cybersecurity Education*. URL: <https://www.slideshare.net/MatthewRosenquist/cybersecurity-curricula-guidelines-for-postsecondarydegree-programs> (accessed 09.03. 2020).
- [Damaševicius et al., 2017] Damaševicius, R., Narbutaite, L., Plauska, I., and Blažauskas, T. (2017). Advances in the use of educational robots in project-based teaching. *TEM Journal*, 6(2):342.
- [Das et al., 2020] Das, M., Marghitu, D., Jamshidi, F., Mandala, M., and Howard, A. (2020). Accessible computer science for k-12 students with hearing impairments. In *International Conference on Human-Computer Interaction*, pages 173–183. Springer.

- [Denner et al., 2014] Denner, J., Werner, L., Campe, S., and Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education*, 46(3):277–296.
- [Denning et al., 2013] Denning, T., Lerner, A., Shostack, A., and Kohno, T. (2013). Control-alt-hack: the design and evaluation of a card game for computer security awareness and education. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 915–928.
- [Denny et al., 2008] Denny, P., Luxton-Reilly, A., and Simon, B. (2008). Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*, pages 113–124.
- [Deshpande et al., 2019] Deshpande, P., Lee, C. B., and Ahmed, I. (2019). Evaluation of peer instruction for cybersecurity education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 720–725.
- [Dillenbourg et al., 2002] Dillenbourg, P., Schneider, D., and Synteta, P. (2002). Virtual learning environments.
- [Djambong and Freiman, 2016] Djambong, T. and Freiman, V. (2016). Task-based assessment of students' computational thinking skills developed through visual programming or tangible coding environments. *International Association for Development of the Information Society*.
- [Dorling and White, 2015] Dorling, M. and White, D. (2015). Scratch: A way to logo and python. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 191–196.
- [Earl et al., 2006] Earl, L., Katz, S., et al. (2006). Rethinking classroom assessment with purpose in mind. *Winnipeg, Manitoba: Western Northern Canadian Protocol*.
- [Eccles, 1983] Eccles, J. (1983). Expectancies, values and academic behaviors. *Achievement and achievement motives*.
- [Elias, 2011] Elias, T. (2011). Learning analytics.
- [Ertmer and Glazewski, 2019] Ertmer, P. A. and Glazewski, K. D. (2019). Scaffolding in pbl environments: Structuring and problematizing relevant task features. *The Wiley Handbook of Problem-Based Learning*, pages 321–342.
- [Feaster et al., 2013] Feaster, Y., Ali, F., Zhai, J., and Hallstrom, J. O. (2013). Serious toys ii: teaching networks, protocols, and algorithms. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 273–278.
- [Fisher, 1936] Fisher, R. A. (1936). Design of experiments. *British Medical Journal*, 1(3923):554.
- [for Women & Information Technology,] for Women & Information Technology, N. C. Computing interest-confidence-perception survey.
- [Fredricks et al., 2005] Fredricks, J. A., Blumenfeld, P., Friedel, J., and Paris, A. (2005). School engagement. *What do children need to flourish? Conceptualizing and measuring indicators of positive development*, pages 305–321.
- [Gansner and North, 2000] Gansner, E. R. and North, S. C. (2000). An open graph visualization system and its applications to software engineering. *Software: practice and experience*, 30(11):1203–1233.
- [Garneli et al., 2015] Garneli, V., Giannakos, M. N., and Chorianopoulos, K. (2015). Computing education in k-12 schools: A review of the literature. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, pages 543–551. IEEE.
- [Geyamallika and Reddy, 2019] Geyamallika, K. and Reddy, E. K. (2019). Survey of emerging threats in cyber security. *International Journal of Engineering Research & Technology (IJERT)*.

- [Giroto et al., 2016] Giroto, V., Lozano, C., Muldner, K., Burleson, W., and Walker, E. (2016). Lessons learned from in-school use of rtag: A robo-tangible learning environment. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 919–930.
- [Giroto et al., 2014] Giroto, V., Thomas, E., Lozano, C., Muldner, K., Burleson, W., and Walker, E. (2014). A tool for integrating log and video data for exploratory analysis and model generation. In *International Conference on Intelligent Tutoring Systems*, pages 69–74. Springer.
- [Grover, 2011] Grover, S. (2011). Robotics and engineering for middle and high school students to develop computational thinking. In *annual meeting of the American educational research association, New Orleans, LA*.
- [Grover, 2017] Grover, S. (2017). Assessing algorithmic and computational thinking in k-12: Lessons from a middle school classroom. In *Emerging research, practice, and policy on computational thinking*, pages 269–288. Springer.
- [Grover and Basu, 2017] Grover, S. and Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*, pages 267–272.
- [Grover et al., 2018] Grover, S., Basu, S., and Schank, P. (2018). What we can learn about student learning from open-ended programming projects in middle school computer science. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 999–1004.
- [Grover et al., 2016a] Grover, S., Bienkowski, M., Niekrasz, J., and Hauswirth, M. (2016a). Assessing problem-solving process at scale. In *Proceedings of the third (2016) ACM conference on learning@ scale*, pages 245–248.
- [Grover et al., 2016b] Grover, S., Bienkowski, M., Tamrakar, A., Siddiquie, B., Salter, D., and Divakaran, A. (2016b). Multimodal analytics to study collaborative problem solving in pair programming. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, pages 516–517.
- [Grover et al., 2020] Grover, S., Cateté, V., Barnes, T., Hill, M., Ledeczi, A., and Broll, B. (2020). First principles to design for online, synchronous high school cs teacher training and curriculum co-design. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, pages 1–5.
- [Grover and Pea, 2013] Grover, S. and Pea, R. (2013). Computational thinking in k–12: A review of the state of the field. *Educational researcher*, 42(1):38–43.
- [Grover and Pea, 2018] Grover, S. and Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school*, 19.
- [Grover et al., 2015] Grover, S., Pea, R., and Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer science education*, 25(2):199–237.
- [Grubbs, 2013] Grubbs, M. (2013). Robotics intrigue middle school students and build stem skills. *Technology and engineering Teacher*, 72(6):12.
- [Guzdial et al., 2014] Guzdial, M., Ericson, B., Mcklin, T., and Engelman, S. (2014). Georgia computes! an intervention in a us state, with formal and informal education in a policy context. *ACM Transactions on Computing Education (TOCE)*, 14(2):1–29.
- [Hamza et al., 2020] Hamza, A., Gharakheili, H. H., and Sivaraman, V. (2020). Iot network security: Requirements, threats, and countermeasures. *arXiv preprint arXiv:2008.09339*.
- [Hannafin et al., 1994] Hannafin, M. J., Hall, C., Land, S., and Hill, J. (1994). Learning in open-ended environments: Assumptions, methods, and implications. *Educational Technology*, 34(8):48–55.

- [Harel and Papert, 1990] Harel, I. and Papert, S. (1990). Software design as a learning environment. *Interactive learning environments*, 1(1):1–32.
- [Harris et al., 2019] Harris, P. A., Taylor, R., Minor, B. L., Elliott, V., Fernandez, M., O’Neal, L., McLeod, L., Delacqua, G., Delacqua, F., Kirby, J., et al. (2019). The redcap consortium: Building an international community of software platform partners. *Journal of biomedical informatics*, 95:103208.
- [Harris et al., 2009] Harris, P. A., Taylor, R., Thielke, R., Payne, J., Gonzalez, N., Conde, J. G., et al. (2009). A metadata-driven methodology and workflow process for providing translational research informatics support. *J Biomed Inform*, 42(2):377–81.
- [Harvey et al., 2013] Harvey, B., Garcia, D. D., Barnes, T., Titterton, N., Armendariz, D., Segars, L., Lemon, E., Morris, S., and Paley, J. (2013). Snap!(build your own blocks). In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 759–759.
- [Haynie and Packman, 2017] Haynie, K. and Packman, S. (2017). Ap cs principles phase ii: Broadening participation in computer science final evaluation report. prepared for the college board and the national science foundation.
- [Hirsch et al., 2007] Hirsch, L. S., Carpinelli, J. D., Kimmel, H., Rockland, R., and Bloom, J. (2007). The differential effects of pre-engineering curricula on middle school students’ attitudes to and knowledge of engineering careers. In *2007 37th Annual Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, pages S2B–17. IEEE.
- [Hoffman et al., 2011] Hoffman, L., Burley, D., and Toregas, C. (2011). Holistically building the cybersecurity workforce. *IEEE Security & Privacy*, 10(2):33–39.
- [Holstein et al., 2019] Holstein, K., McLaren, B. M., and Aleven, V. (2019). Co-designing a real-time classroom orchestration tool to support teacher-ai complementarity. *Grantee Submission*.
- [Hubwieser et al., 2014] Hubwieser, P., Armoni, M., Giannakos, M. N., and Mittermeir, R. T. (2014). Perspectives and visions of computer science education in primary and secondary (k-12) schools. *ACM Transactions on Computing Education (TOCE)*, 14(2):1–9.
- [Hutchins et al., 2018] Hutchins, N., Biswas, G., Conlin, L., Emara, M., Grover, S., Basu, S., and McELHANEY, K. (2018). Studying synergistic learning of physics and computational thinking in a learning by modeling environment. In *26th International Conference on Computers in Education (ICCE), Manila, Philippines*.
- [Hutchins et al., 2020a] Hutchins, N., Biswas, G., Wolf, R., Chin, D., Grover, S., and Blair, K. (2020a). Computational thinking in support of learning and transfer.
- [Hutchins, 2022] Hutchins, N. M. (2022). *Co-Designing Teaching Augmentation Tools to Support the Integration of Problem-Based Learning in K-12 Science*. PhD thesis, Vanderbilt University.
- [Hutchins et al., 2021] Hutchins, N. M., Basu, S., McElhaney, K., Chiu, J., Fick, S., Zhang, N., and Biswas, G. (2021). Coherence across conceptual and computational representations of students’ scientific models. *Computersupported collaborative learning*.
- [Hutchins et al., 2020b] Hutchins, N. M., Biswas, G., Maróti, M., Lédeczi, Á., Grover, S., Wolf, R., Blair, K. P., Chin, D., Conlin, L., Basu, S., et al. (2020b). C2stem: a system for synergistic learning of physics and computational thinking. *Journal of Science Education and Technology*, 29(1):83–100.
- [Irgens et al., 2020] Irgens, G. A., Dabholkar, S., Bain, C., Woods, P., Hall, K., Swanson, H., Horn, M., and Wilensky, U. (2020). Modeling and measuring high school students’ computational thinking practices in science. *Journal of Science Education and Technology*, 29(1):137–161.
- [Jang-Jaccard and Nepal, 2014] Jang-Jaccard, J. and Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5):973–993.

- [Javidi and Sheybani, 2018] Javidi, G. and Sheybani, E. (2018). K-12 cybersecurity education, research, and outreach. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE.
- [Jin et al., 2018] Jin, G., Tu, M., Kim, T.-H., Heffron, J., and White, J. (2018). Evaluation of game-based learning in cybersecurity education for high school students. *Journal of Education and Learning (Ed-uLearn)*, 12(1):150–158.
- [Jin et al., 2017] Jin, G., Tu, M., Kim, T.-H., Heffron, J. D., and White, J. K. (2017). Pnw gencyber summer camp: Game based cybersecurity education for high school students. In *2017 Mid-Atlantic Section Fall Conference*.
- [Jose et al., 2016] Jose, M., Kurian, P. S., and Biju, V. (2016). Progression analysis of students in a higher education institution using big data open source predictive modeling tool. In *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pages 1–5. IEEE.
- [K-12, 2016] K-12 (2016). K12 Computer Science Framework. <http://www.k12cs.org>. Cited 2020 July 31.
- [K-12, 2020] K-12 (2020). AP Computer Science Principles Course and Exam Description. <https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf?course=ap-computer-science-principles>. Cited 2020 July 31.
- [K-12 Cybersecurity, 2021] K-12 Cybersecurity (2021). The State of Cybersecurity Education in K-12 Schools. <https://cyber.org/sites/default/files/2020-06/The%20State%20of%20Cybersecurity%20Education%20in%20K-12%20Schools.pdf>. Cited 2021 November 24.
- [Karaahmetoglu and Korkmaz, 2019] Karaahmetoglu, K. and Korkmaz, Ö. (2019). The effect of project-based arduino educational robot applications on students’ computational thinking skills and their perception of basic stem skill levels. *Online Submission*, 6(2):1–14.
- [Karaman et al., 2017] Karaman, S., Anders, A., Boulet, M., Connor, J., Gregson, K., Guerra, W., Guldner, O., Mohamoud, M., Plancher, B., Shin, R., et al. (2017). Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at mit. In *2017 IEEE integrated STEM education conference (ISEC)*, pages 195–203. IEEE.
- [Karim et al., 2015] Karim, M. E., Lemaignan, S., and Mondada, F. (2015). A review: Can robots reshape k-12 stem education? In *2015 IEEE international workshop on Advanced robotics and its social impacts (ARSO)*, pages 1–8. IEEE.
- [Kelly et al., 2019] Kelly, N., Wright, N., Dawes, L., Kerr, J., and Robertson, A. (2019). Co-design for curriculum planning: A model for professional development for high school teachers. *Australian Journal of Teacher Education (Online)*, 44(7):84–107.
- [Khanlari, 2013] Khanlari, A. (2013). Effects of educational robots on learning stem and on students’ attitude toward stem. In *2013 IEEE 5th Conference on Engineering Education (ICEED)*, pages 62–66. IEEE.
- [Kim et al., 2016] Kim, S., Kim, J. W., Park, J., and Oh, A. (2016). Elice: an online cs education platform to understand how students learn programming. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pages 225–228.
- [Kinnebrew et al., 2013] Kinnebrew, J. S., Loretz, K. M., and Biswas, G. (2013). A contextualized, differential sequence mining method to derive students’ learning behavior patterns. *Journal of Educational Data Mining*, 5(1):190–219.
- [Kinnebrew et al., 2017] Kinnebrew, J. S., Segedy, J. R., and Biswas, G. (2017). Integrating model-driven and data-driven techniques for analyzing learning behaviors in open-ended learning environments. *IEEE Transactions on Learning Technologies*, 10(2):140–153.
- [Knobelsdorf and Vahrenhold, 2013] Knobelsdorf, M. and Vahrenhold, J. (2013). Addressing the full range of students: challenges in k-12 computer science education. *Computer*, 46(9):32–37.

- [Koh et al., 2016] Koh, E., Shibani, A., Tan, J. P.-L., and Hong, H. (2016). A pedagogical framework for learning analytics in collaborative inquiry tasks: An example from a teamwork competency awareness program. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, pages 74–83.
- [Kurose et al., 2002] Kurose, J., Liebeherr, J., Ostermann, S., and Ott-Boisseau, T. (2002). Acm sigcomm workshop on computer networking: Curriculum designs and educational challenges. *ACM SIGCOMM Computer Communications Review*, 32(5).
- [Ladabouche and LaFountain, 2016] Ladabouche, T. and LaFountain, S. (2016). Gencyber: Inspiring the next generation of cyber stars. *IEEE Security & Privacy*, 14(5):84–86.
- [Lakanen and Isomöttönen, 2015] Lakanen, A.-J. and Isomöttönen, V. (2015). What does it take to do computer programming? surveying the k-12 students’ conceptions. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 458–463.
- [Land, 2000] Land, S. M. (2000). Cognitive requirements for learning with open-ended learning environments. *Educational Technology Research and Development*, 48(3):61–78.
- [Land and Hannafin, 1996] Land, S. M. and Hannafin, M. J. (1996). A conceptual framework for the development of theories-in-action with open-ended learning environments. *Educational Technology Research and Development*, 44(3):37–53.
- [Lawhead et al., 2002] Lawhead, P. B., Duncan, M. E., Bland, C. G., Goldweber, M., Schep, M., Barnes, D. J., and Hollingsworth, R. G. (2002). A road map for teaching introductory programming using lego® mindstorms robots. *Acm sigcse bulletin*, 35(2):191–201.
- [Lédeczi et al., 2019] Lédeczi, Á., MarÓti, M., Zare, H., Yett, B., Hutchins, N., Broll, B., Völgyesi, P., Smith, M. B., Darrah, T., Metelko, M., et al. (2019). Teaching cybersecurity with networked robots. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 885–891.
- [Leelawong and Biswas, 2008] Leelawong, K. and Biswas, G. (2008). Designing learning by teaching agents: The betty’s brain system. *International Journal of Artificial Intelligence in Education*, 18(3):181–208.
- [Lin and Van Brummelen, 2021] Lin, P. and Van Brummelen, J. (2021). Engaging teachers to co-design integrated ai curriculum for k-12 classrooms. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–12.
- [Linnenbrink, 2005] Linnenbrink, E. A. (2005). The dilemma of performance-approach goals: The use of multiple goal contexts to promote students’ motivation and learning. *Journal of educational psychology*, 97(2):197.
- [Linnenbrink-Garcia et al., 2018] Linnenbrink-Garcia, L., Wormington, S. V., Snyder, K. E., Riggsbee, J., Perez, T., Ben-Eliyahu, A., and Hill, N. E. (2018). Multiple pathways to success: An examination of integrative motivational profiles among upper elementary and college students. *Journal of educational psychology*, 110(7):1026.
- [Liu et al., 2013] Liu, A., Newsom, J., Schunn, C., and Shoop, R. (2013). Students learn programming faster through robotic simulation. *Tech Directions*, 72(8):16.
- [Liu et al., 2018] Liu, R., Stamper, J. C., and Davenport, J. (2018). A novel method for the in-depth multi-modal analysis of student learning trajectories in intelligent tutoring systems. *Journal of Learning Analytics*, 5(1):41–54.
- [Lockwood and Mooney, 2017] Lockwood, J. and Mooney, A. (2017). Computational thinking in education: Where does it fit? a systematic literary review. *arXiv preprint arXiv:1703.07659*.

- [Maennel, 2020] Maennel, K. (2020). Learning analytics perspective: Evidencing learning from digital datasets in cybersecurity exercises. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 27–36. IEEE.
- [Mahzoon et al., 2018] Mahzoon, M. J., Maher, M. L., Eltayeb, O., Dou, W., and Grace, K. (2018). A sequence data model for analyzing temporal patterns of student data. *Journal of Learning Analytics*, 5(1):55–74.
- [Maloney et al., 2004] Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. (2004). Scratch: a sneak preview [education]. In *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.*, pages 104–109. IEEE.
- [Mangaroska et al., 2020] Mangaroska, K., Sharma, K., Gašević, D., and Giannakos, M. (2020). Multimodal learning analytics to inform learning design: Lessons learned from computing education. *Journal of Learning Analytics*, 7(3):79–97.
- [Mao et al., 2021] Mao, Y., Shi, Y., Marwan, S., Price, T. W., Barnes, T., and Chi, M. (2021). Knowing when and where: Temporal-astnn for student learning progression in novice programming tasks. *International Educational Data Mining Society*.
- [Martin et al., 2012] Martin, A. J., Anderson, J., Bobis, J., Way, J., and Vellar, R. (2012). Switching on and switching off in mathematics: An ecological study of future intent and disengagement among middle school students. *Journal of Educational Psychology*, 104(1):1.
- [Matuk et al., 2016] Matuk, C., Gerard, L., Lim-Breitbart, J., and Linn, M. (2016). Gathering requirements for teacher tools: Strategies for empowering teachers through co-design. *Journal of Science Teacher Education*, 27(1):79–110.
- [McLurkin et al., 2013] McLurkin, J., Lynch, A. J., Rixner, S., Barr, T. W., Chou, A., Foster, K., and Bilstein, S. (2013). A low-cost multi-robot system for research, teaching, and outreach. In *Distributed Autonomous Robotic Systems*, pages 597–609. Springer.
- [Midgley et al., 2000] Midgley, C., Maehr, M. L., Hruda, L. Z., Anderman, E., Anderman, L., Freeman, K. E., Urdan, T., et al. (2000). Manual for the patterns of adaptive learning scales. *Ann Arbor: University of Michigan*, pages 734–763.
- [Min et al., 2014] Min, W., Mott, B., and Lester, J. (2014). Adaptive scaffolding in an intelligent game-based learning environment for computer science. In *Proceedings of the Workshop on AI-supported Education for Computer Science (AIEDCS) at the 12th International Conference on Intelligent Tutoring Systems*, pages 41–50.
- [Mirkovic et al., 2015] Mirkovic, J., Dark, M., Du, W., Vigna, G., and Denning, T. (2015). Evaluating cybersecurity education interventions: Three case studies. *IEEE Security & Privacy*, 13(3):63–69.
- [Mishra et al., 2017] Mishra, S., Raj, R. K., Tymann, P., Fagan, J., and Miller, S. (2017). Cybercsp: Integrating cybersecurity into the computer science principles course. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE.
- [Mislevy et al., 2003] Mislevy, R. J., Almond, R. G., and Lukas, J. F. (2003). A brief introduction to evidence-centered design. *ETS Research Report Series*, 2003(1):i–29.
- [Mislevy and Haertel, 2006] Mislevy, R. J. and Haertel, G. D. (2006). Implications of evidence-centered design for educational testing. *Educational Measurement: Issues and Practice*, 25(4):6–20.
- [Mislevy and Riconscente, 2005] Mislevy, R. J. and Riconscente, M. M. (2005). Evidence-centered assessment design: Layers, structures, and terminology.
- [Moons and De Backer, 2013] Moons, J. and De Backer, C. (2013). The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education*, 60(1):368–384.

- [Mouheeb et al., 2019] Mouheeb, D., Abbas, S., and Merabti, M. (2019). Cybersecurity curriculum design: A survey. In *Transactions on Edutainment XV*, pages 93–107. Springer.
- [Mountrouidou et al., 2019] Mountrouidou, X., Vosen, D., Kari, C., Azhar, M. Q., Bhatia, S., Gagne, G., Maguire, J., Tudor, L., and Yuen, T. T. (2019). Securing the human: a review of literature on broadening diversity in cybersecurity education. *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, pages 157–176.
- [Murphy et al., 2008] Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., and Zander, C. (2008). Debugging: the good, the bad, and the quirky—a qualitative analysis of novices’ strategies. *ACM SIGCSE Bulletin*, 40(1):163–167.
- [Nag et al., 2013] Nag, S., Katz, J. G., and Saenz-Otero, A. (2013). Collaborative gaming and competition for cs-stem education using spheres zero robotics. *Acta astronautica*, 83:145–174.
- [Namin et al., 2016] Namin, A. S., Aguirre-Muñoz, Z., and Jones, K. S. (2016). Teaching cyber security through competition: An experience report about a participatory training workshop. In *International Conference on Computer Science Education Innovation & Technology (CSEIT). Proceedings*, page 98. Global Science and Technology Forum.
- [Nasir et al., 2019] Nasir, J., Norman, U., Johal, W., Olsen, J. K., Shahmoradi, S., and Dillenbourg, P. (2019). Robot analytics: What do human-robot interaction traces tell us about learning? In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–7. IEEE.
- [Noroozi et al., 2019] Noroozi, O., Alikhani, I., Järvelä, S., Kirschner, P. A., Juuso, I., and Seppänen, T. (2019). Multimodal data to design visual learning analytics for understanding regulation of learning. *Computers in Human Behavior*, 100:298–304.
- [Nugent et al., 2009] Nugent, G., Barker, B., Grandgenett, N., and Adamchuk, V. (2009). The use of digital manipulatives in k-12: robotics, gps/gis and programming. In *2009 39th IEEE Frontiers in Education Conference*, pages 1–6. IEEE.
- [Nugent et al., 2010] Nugent, G., Barker, B., Grandgenett, N., and Adamchuk, V. I. (2010). Impact of robotics and geospatial technology interventions on youth stem learning and attitudes. *Journal of Research on Technology in Education*, 42(4):391–408.
- [Nugent et al., 2016] Nugent, G., Barker, B., Grandgenett, N., and Welch, G. (2016). Robotics camps, clubs, and competitions: Results from a us robotics project. *Robotics and Autonomous Systems*, 75:686–691.
- [of Education, 2018] of Education, T. D. (2018). Digital readiness: K-8 computer science standards. <https://www.tn.gov/education/instruction/academic-standards/computer-science.html>. Cited 2021 August 19.
- [Olano et al., 2014] Olano, M., Sherman, A., Oliva, L., Cox, R., Firestone, D., Kubik, O., Patil, M., Seymour, J., Sohn, I., and Thomas, D. (2014). Securityempire: Development and evaluation of a digital game to promote cybersecurity education. In *2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*.
- [Olsen et al., 2020] Olsen, J. K., Sharma, K., Rummel, N., and Aleven, V. (2020). Temporal analysis of multimodal data to predict collaborative learning outcomes. *British Journal of Educational Technology*, 51(5):1527–1547.
- [O’Malley and Fraser, 2004] O’Malley, C. and Fraser, D. S. (2004). Literature review in learning with tangible technologies.
- [O’Rourke et al., 2015] O’Rourke, E., Andersen, E., Gulwani, S., and Popović, Z. (2015). A framework for automatically generating interactive instructional scaffolding. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 1545–1554.
- [Özgür, 2018] Özgür, A. (2018). *Cellulo: Tangible Haptic Swarm Robots for Learning*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne.

- [Özgür et al., 2017] Özgür, A., Lemaignan, S., Johal, W., Beltran, M., Briod, M., Pereyre, L., Mondada, F., and Dillenbourg, P. (2017). Cellulo: Versatile handheld robots for education. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 119–127. IEEE.
- [Papert, 1980] Papert, S. A. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic books.
- [Parker and DeLyser, 2017] Parker, M. C. and DeLyser, L. A. (2017). Concepts and practices: Designing and developing a modern k-12 cs framework. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*, pages 453–458.
- [Payne et al., 2016] Payne, B. R., Abegaz, T., and Antonia, K. (2016). Planning and implementing a successful nsa-nsf gencyber summer cyber academy. *Journal of Cybersecurity Education, Research and Practice*, 2016(2):3.
- [Perlin et al., 2018] Perlin, K., He, Z., and Rosenberg, K. (2018). Chalktalk: A visualization and communication language—as a tool in the domain of computer science education. *arXiv preprint arXiv:1809.07166*.
- [Petre and Price, 2004] Petre, M. and Price, B. (2004). Using robotics to motivate ‘back door’ learning. *Education and information technologies*, 9(2):147–158.
- [Piech et al., 2012] Piech, C., Sahami, M., Koller, D., Cooper, S., and Blikstein, P. (2012). Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 153–160.
- [Pintrich et al., 1993] Pintrich, P., Smith, D., Garcia, T., and McKeachie, W. (1993). Predictive validity and reliability of the motivated strategies for learning questionnaire. *Educational and Psychological Measurement*, 53(3):801–813.
- [Rachmatullah et al., 2020] Rachmatullah, A., Akram, B., Boulden, D., Mott, B., Boyer, K., Lester, J., and Wiebe, E. (2020). Development and validation of the middle grades computer science concept inventory (mg-csci) assessment. *EURASIA Journal of Mathematics, Science and Technology Education*, 16(5):em1841.
- [Ramachandran et al., 2016] Ramachandran, A., Litoiu, A., and Scassellati, B. (2016). Shaping productive help-seeking behavior during robot-child tutoring interactions. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 247–254. IEEE.
- [Rand et al., 2018] Rand, K., Feil-Seifer, D., and Sengupta, S. (2018). Unplugged robotics as a platform for cybersecurity education in the elementary classroom. *Information systems education journal*.
- [Rodríguez et al., 2013] Rodríguez, F. J., Kerby, N. D., and Boyer, K. E. (2013). Informing the design of a game-based learning environment for computer science: A pilot study on engagement and collaborative dialogue. In *The Proceeding of the 1st Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, pages 30–39.
- [Román-González et al., 2017] Román-González, M., Moreno-León, J., and Robles, G. (2017). Complementary tools for computational thinking assessment. In *Proceedings of International Conference on Computational Thinking Education (CTE 2017)*, S. C Kong, J Sheldon, and K. Y Li (Eds.). *The Education University of Hong Kong*, pages 154–159.
- [Roschelle et al., 2006] Roschelle, J., Penuel, W., and Shechtman, N. (2006). Co-design of innovations with teachers: Definition and dynamics.
- [Russ et al., 2008] Russ, R. S., Scherr, R. E., Hammer, D., and Mikeska, J. (2008). Recognizing mechanistic reasoning in student scientific inquiry: A framework for discourse analysis developed from philosophy of science. *Science education*, 92(3):499–525.
- [Russell and Norvig, 2002] Russell, S. and Norvig, P. (2002). *Artificial intelligence: a modern approach*.

- [Sánchez et al., 2020] Sánchez, J., Mallorquí, A., Briones, A., Zaballos, A., and Corral, G. (2020). An integral pedagogical strategy for teaching and learning iot cybersecurity. *Sensors*, 20(14):3970.
- [Sarkar, 2006] Sarkar, N. I. (2006). Teaching computer networking fundamentals using practical laboratory exercises. *IEEE Transactions on education*, 49(2):285–291.
- [Scornavacco et al., 2022] Scornavacco, K., Kelly, M. R., and Boardman, A. (2022). Leading with curricular co-design: An exploration of teacher leadership through the co-design process. *Journal of Research on Leadership Education*, 17(4):333–357.
- [Seehorn and Clayborn, 2017] Seehorn, D. and Clayborn, L. (2017). Csta k-12 cs standards for all. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 730–730.
- [Segedy et al., 2011] Segedy, J. R., Kinnebrew, J. S., and Biswas, G. (2011). Modeling learner’s cognitive and metacognitive strategies in an open-ended learning environment. In *AAAI Fall Symposium: Advances in Cognitive Systems*.
- [Segedy et al., 2013] Segedy, J. R., Kinnebrew, J. S., and Biswas, G. (2013). The effect of contextualized conversational feedback in a complex open-ended learning environment. *Educational Technology Research and Development*, 61(1):71–89.
- [Sengupta et al., 2013] Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., and Clark, D. (2013). Integrating computational thinking with k-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, 18(2):351–380.
- [Sentance and Csizmadia, 2017] Sentance, S. and Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher’s perspective. *Education and Information Technologies*, 22(2):469–495.
- [Sipitakiat et al., 2002] Sipitakiat, A., Blikstein, P., and Cavallo, D. (2002). The gogo board: Moving towards highly available computational tools in learning environments. In *Proceedings of Interactive Computer Aided Learning International Workshop*.
- [Spikol et al., 2018] Spikol, D., Ruffaldi, E., Dabisias, G., and Cukurova, M. (2018). Supervised machine learning in multimodal learning analytics for estimating success in project-based learning. *Journal of Computer Assisted Learning*, 34(4):366–377.
- [Starr et al., 2018] Starr, E. L., Reilly, J. M., and Schneider, B. (2018). Toward using multi-modal learning analytics to support and measure collaboration in co-located dyads. International Society of the Learning Sciences, Inc.[ISLS].
- [Stein and Lédeczi, 2021] Stein, G. and Lédeczi, A. (2021). Enabling collaborative distance robotics education for novice programmers. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–5. IEEE.
- [Švábenskỳ et al., 2020] Švábenskỳ, V., Vykopal, J., and Čeleda, P. (2020). What are cybersecurity education papers about? a systematic literature review of sigcse and iticse conferences. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 2–8.
- [Tang et al., 2020] Tang, X., Yin, Y., Lin, Q., Hadad, R., and Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, page 103798.
- [Taub et al., 2012] Taub, R., Armoni, M., and Ben-Ari, M. (2012). Cs unplugged and middle-school students’ views, attitudes, and intentions regarding cs. *ACM Transactions on Computing Education (TOCE)*, 12(2):1–29.
- [Tisue and Wilensky, 2004] Tisue, S. and Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA.

- [Touretzky et al., 2013] Touretzky, D. S., Marghitu, D., Ludi, S., Bernstein, D., and Ni, L. (2013). Accelerating k-12 computational thinking using scaffolding, staging, and abstraction. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 609–614.
- [Trabelsi and Barka, 2019] Trabelsi, Z. and Barka, E. (2019). A basic course model on information security for high school it curriculum. In *2019 IEEE Global Engineering Education Conference (EDUCON)*, pages 63–70. IEEE.
- [Tselegkaridis and Sapounidis, 2021] Tselegkaridis, S. and Sapounidis, T. (2021). Simulators in Educational Robotics: A Review. *Education Sciences*, 11(1):11.
- [Tucker et al., 2003] Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., and Verno, A. (2003). *A Model Curriculum for K–12 Computer Science*. ACM/Association for Computing Machinery.
- [Vykopal et al., 2021] Vykopal, J., Čeleda, P., Seda, P., Švábenský, V., and Tovarňák, D. (2021). Scalable learning environments for teaching cybersecurity hands-on. *arXiv preprint arXiv:2110.10004*.
- [Walker et al., 2016] Walker, E., Giroto, V., Kim, Y., and Muldner, K. (2016). The effects of physical form and embodied action in a teachable robot for geometry learning. In *2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*, pages 381–385. IEEE.
- [Wang et al., 2016] Wang, J., Hong, H., Ravitz, J., and Hejazi Moghadam, S. (2016). Landscape of k-12 computer science education in the us: Perceptions, access, and barriers. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 645–650.
- [Watson et al., 2013] Watson, C., Li, F. W., and Godwin, J. L. (2013). Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *2013 IEEE 13th international conference on advanced learning technologies*, pages 319–323. IEEE.
- [Webb et al., 2017] Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., and Sysło, M. M. (2017). Computer science in k-12 school curricula of the 21st century: Why, what and when? *Education and Information Technologies*, 22(2):445–468.
- [Weese and Feldhausen, 2017] Weese, J. L. and Feldhausen, R. (2017). Stem outreach: Assessing computational thinking and problem solving. In *ASEE Annual Conference & Exposition*.
- [Weintrop et al., 2016] Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., and Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1):127–147.
- [Weintrop and Wilensky, 2017] Weintrop, D. and Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–25.
- [Wiebe et al., 2019] Wiebe, E., London, J., Aksit, O., Mott, B. W., Boyer, K. E., and Lester, J. C. (2019). Development of a lean computational thinking abilities assessment for middle grades students. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 456–461.
- [Wilson and Sloane, 2000] Wilson, M. and Sloane, K. (2000). From principles to practice: An embedded assessment system. *Applied measurement in education*, 13(2):181–208.
- [Wing, 2006] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.
- [Witherspoon et al., 2017] Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., and Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–20.
- [Wu, 2016] Wu, B. (2016). Robotics programming in support of computational thinking: Scaffolding from teacher, peer, and robotics agents. *Transforming Learning, Empowering Learners*.

- [Wu et al., 2020] Wu, S., Peel, A., Bain, C., Anton, G., Horn, M., and Wilensky, U. (2020). Workshops and co-design can help teachers integrate computational thinking into their k-12 stem classes. In *Proceedings of International Conference on Computational Thinking Education 2020*.
- [Yett et al., 2020a] Yett, B., Hutchins, N., Stein, G., Zare, H., Snyder, C., Biswas, G., Metelko, M., and Lédeczi, Á. (2020a). A hands-on cybersecurity curriculum using a robotics platform. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1040–1046.
- [Yett et al., 2020b] Yett, B., Snyder, C., Zhang, N., Hutchins, N., Mishra, S., and Biswas, G. (2020b). Using log and discourse analysis to improve understanding of collaborative programming. In *Proceedings of the 28th International Conference on Computers in Education*, volume 1, pages 137–146. Asia-Pacific Society for Computers in Education.
- [Yuan and Zhong, 2009] Yuan, D. and Zhong, J. (2009). An instructional design of open source networking laboratory and curriculum. In *Proceedings of the 10th ACM conference on SIG-information technology education*, pages 37–42.
- [Yuan et al., 2019] Yuan, X., Zhang, T., Shama, A. A., Xu, J., Yang, L., Ellis, J., He, W., and Waters, C. (2019). Teaching cybersecurity using guided inquiry collaborative learning. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–6. IEEE.
- [Zhang, 2020] Zhang, N. (2020). *Supporting the integrated learning of science, engineering, and computational thinking in an open-ended learning environment*. PhD thesis, Vanderbilt University.