Educational Designer
Journal of the International Society for Design and Development in Education

Designing an Adaptive Assessment for Preschool Children's Robotics Knowledge

Amanda Strawhacker, Emily Relkin,
Marina Umaschi Bers

Abstract

Although Computer Science (CS) is gaining popularity in early education settings in the US and worldwide, there is a lack of agreement about how to assess learning in young children, particularly preschoolers. The current study presents the design and pilot of a developmentally appropriate assessment tool, the Coding Stages Assessment-KIBO (CSA-KIBO), to evaluate preschool children's coding skills with a robot kit (KIBO) designed for young children. Using a design-based research framework we developed evidence-based design criteria to inform our iteratively-tested assessment tool. In this paper, we address the following research question: How does the mode of administration of the CSA-KIBO robotics assessment impact performance among preschool students? We administered CSA-KIBO to 151 coding naive preschool students ages 3-5 years, from ethnically, socioeconomically, and linguistically diverse backgrounds. Results showed that shorter administration formats were more suitable for our preschool sample and yielded similar assessment results to the lengthier format. Across all formats of administration, a possible floor effect was present in our coding-inexperienced sample. We consider the major contribution from this study to be a focused exploration of assessment administration as a critical aspect of assessment design for preschool-aged learners.

Introduction

Computer science is gaining popularity in education settings in the US and worldwide. There is increasing interest in research, policy mandates, pedagogical approaches, learning standards, and commercial products to explore and support introducing computational skills as early as pre-kindergarten (Code.org, CSTA & ECEP Alliance, 2021; International Society for Technology in Education, 2007; NAEYC & Fred Rogers Center for Early Learning and Children's Media, 2012; Royal Society, 2017; U.S. Department of Education & U.S. Department of Health and Human Services, 2016; Vegas & Fowler, 2020). All 50 US states have voiced some level of support for integrating computer science into school curricula, with 23 currently adopting or implementing a policy to provide access at the high school level. Of these, 11 are further offering access at all K-12 levels (Code.org, CSTA & ECEP Alliance, 2021). Internationally, 219 countries have been identified as having CS frameworks in place in formal K-12 education settings (Vegas & Fowler, 2020). However, less work has focused on coding interventions in Pre-Kindergarten (PreK).

Research has shown that educational investment beginning in early childhood can have lasting gains, and that computer science interventions in PreK can improve children's confidence and positive attitudes toward learning with robotics and technology (Bers, 2020; Kazakoff & Bers, 2014; Zviel-Girshin, Luria & Shaham, 2020). Further, these experiences may have benefits beyond computer science content areas, supporting the development of foundational skills such as numeracy, literacy, problem solving, and sequencing (Bers, 2020; Kazakoff & Bers, 2014; Liao & Bright, 1991; Zviel-Girshin, Luria & Shaham, 2020). Computer-based play has even been suggested to support non-academic skills such as executive functioning (a broad skill set including mental flexibility, inhibitory control, and working memory), communication, and socio-emotional development (Clements & Sarama, 2004; Myers, 2021**;** Bers 2020).

Among educational computer tools, programmable robotic kits have emerged as an effective way to engage Preschool-aged learners in tangible computational experiences. Screen-free technologies such as the KIBO robotics kit (www.kinderlabrobotics.com), the BeeBot robot (www.terrapinlogo.com), and the Code-a-Pillar robot (www.fisher-price.com) engage children as young as 3 years old in tangible play with engineering construction and computer programming (also called coding). Young children become engineers by playing with motors and sensors as well as storytellers by creating and sharing personally meaningful projects that react in response to their environment (Bers, 2020). Thus, the use of robotic systems in early childhood starting in PreK can expand the range of computer science concepts and skills and include topics related to hardware and software, inputs and outputs. Because they facilitate cognitive as well as fine motor and social development, educational robotics kits are developmentally appropriate for early childhood education (Bers, 2020; Clements & Sarama, 2004; Svensson, 2000). Early intervention is even more critical for girls and children from minority backgrounds underrepresented in STEM, who have fewer opportunities to develop positive attitudes toward CS and STEM (Code.org, CSTA & ECEP Alliance, 2021). Research shows that early exposure to STEM curricula and computer programming may mitigate gender-based stereotypes regarding STEM careers, increase interest in engineering, and lower obstacles to entering these fields later in life for girls and minoritized children (Metz, 2007; Sullivan & Bers, 2018).

Despite the widespread popularity of coding, robotic, and computational tools, and the growing body of research supporting the educational and psychosocial benefits of these domains, there is currently a lack of consensus about the best approach to assessment of computational learning (So, Jong & Liu, 2020), especially for preschool-aged learners (Hsu, Chang & Hung, 2018). In order to design effective coding initiatives, we need to be able to assess children's learning about coding, and their understanding and abilities at baseline, prior to learning interventions (Webb et al., 2017). A growing number of organizations are working to address this gap through standardized computer science and computational thinking tests, such as the International Computer and Information Literacy Study (designed for 13-year-old students), the OECD's 2021 PISA Math and Science assessment (15-year-olds), and the Advanced Placement Computer Science exam (18-year-olds) (Fraillon et al., 2019; Hansen, Levesque, Valant & Quintero, 2018). However, many of these initiatives focus on adolescence, long after the critical period of early childhood when coding and robotic interventions may have formative impacts.

The current study explores the initial design of an assessment tool, Coding Stages Assessment-KIBO (CSA-KIBO), to evaluate children's computational fluency and understanding of a robot and coding language designed specifically for preschool through elementary aged children (the KIBO robot kit). In order to test how our assessment performs with the youngest and least-experienced cohort of our assessment population, our study focuses on preschool children with no prior coding experience. Bers (2020) proposed the Coding Stages Framework as part of her Coding as Another Language (CAL) pedagogical approach to introduce the idea that when children learn to code, they follow a developmental progression through recognizable milestones, similar to the development pathway to literacy, numeracy, and many other better-researched academic skills (Clements & Sarama, 2004). Because of the expressive and communicative opportunities that coding presents to young learners, Bers draws connections between developmental skills shared by coding and literacy, such as reading, composition, revision, and creative self-expression (Bers, 2020; Resnick & Robinson, 2017). This perspective, that coding should be taught as an expressive form, is in direct response to narrow conceptions of computer science education as an extension of logical reasoning, traditionally taught using sequence-based puzzles, abstract problem sets, and other approaches that do not leverage children's creativity and agency in the learning process (Bers, Govind & Relkin, 2021; Wing, 2011). By presenting coding as a language, complete with its own symbolic and syntactic systems, the CAL approach shifts the learning goal from technical proficiency with code, to fluency in conveying and interpreting ideas through original coded works (Hudson et al., 2011).

In the following sections, we outline in more depth the Coding Stages Framework used to design the assessment, describe findings from design research with related early childhood coding assessments, and explore the unique context of assessing children's knowledge within robotic coding environments. Following this, we describe the current study's CSA-KIBO design approach.

Background

Assessments of Young Children's Coding and Robotic Knowledge

Prior research has explored the development of assessments to demonstrate young children's learning with educational programming and robotic tools, a unique challenge given that written tests are not appropriate for this age group. In 2015, a research team piloted an assessment for 4-6 year old students called Solve-Its, story-based tasks with robotic main characters, which prompted children to use coding instructions to copy robot actions described in the story-contexts (Strawhacker & Bers, 2015; Sullivan & Bers, 2018). Solve-Its measured children's programming ability using two metrics: block recognition and sequencing ability. KIBO Mastery Challenges, a later iteration of Solve-Its, was designed to assess students' mastery of KIBO programming language and programming concepts introduced in a curriculum (Hassenfeld et al., 2020). While these assessments indicate proficiency in a range of algorithmic and logic-based skills, they were not able to capture children's ability to express themselves through programming, a key theoretical aspect of the Coding Stages Framework. Additionally, they do not incorporate a developmental perspective into the scoring system, making it difficult for educators and practitioners to know how to meaningfully interpret results and compare students along predicted developmental pathways.

One exception is the Coding Stages Assessment, an assessment based on Bers' (2020) Coding Stages Framework. Researchers de Ruiter and Bers (2021) first developed this metric for use specifically with the ScratchJr coding environment, a tablet-based programming app for children ages 5-7 years. As with the current study, a focus of that design process was to create an assessment (the CSA-ScratchJr) to capture children's technical facility with the given programming language (e.g., the syntax and grammar), but also situate their coding-medium fluency in communicating and expressing complex ideas along a developmental pathway. This approach is aligned with the CAL curriculum and philosophy of coding as an expressive, progressive, and linguistic form, and also with prior research which showed that offering young children purposeful, goal-directed programming and debugging assessment questions (rather than rote tasks) helps them perform at a deeper and more engaged level (Lee & Ko, 2014). The purpose of exploring children's competence within a range of specific programming contexts, rather than focusing on general computational thinking ability (e.g., in an "unplugged" setting) is to allow children to leverage their context-dependent skills and fluency in order to best demonstrate their developmental coding stage.

Development of the CSA-KIBO followed a similar method to the development of the CSA-ScratchJr. Specifically, we applied a design-based research approach (e.g., Anderson & Shattuck, 2012; Edelson, 2002) by leveraging findings from earlier versions of coding and robotic assessments (e.g., Strawhacker & Bers, 2015; de Ruiter & Bers, 2021; Relkin et al., 2021) and honing our design goals to focus on a developmentally-informed KIBO robotic assessment for young children. As a methodology, design research emphasizes designing educational interventions according to intentional theoretical specifications to achieve a chosen learning outcome, iteratively implementing the intervention in real-world settings, and using evidence from implementations to inform future trials and design modifications (Edelson, 2002). In the following sections, we outline the design considerations, principles, and criteria guiding our development of the CSA-KIBO and share results from our early trials and iterative refinements as we work to approach a viable assessment prototype ready for validation and scaling to broader education communities.

Design Considerations for Assessing Young Children

Young children represent a unique population for studying computational thinking and computer science skills development (e.g., Chen, et al., 2017). Complex "if-then" conditionals, variables, and abstract logical statements are typically beyond their capacity (Janveau-Brennan & Markovits, 1999), and an assessment using these techniques would poorly reflect their true abilities on foundational computer science tasks. Young children are also prone to magical thinking as a way to explain the behavior of unknown or complex systems, including technology (Mioduser et al., 2009). However, some research suggests that this effect diminishes with increasing exposure to technology such as KIBO. Mioduser, Levy, and Tallis (2009) investigated Kindergarten children's utterances when observing a robotic program being executed. As children developed more sophisticated understandings of the robot's program, their explanations shifted from animistic descriptions of what the robot "wanted" to do, to explanations emphasizing patterns and rules in the robot's behavior. For this reason, any summative assessment should be situated within the context of a developmentally-appropriate programming language or computer environment that is familiar to the child.

Flannery and Bers (2013) and Vizner (2017) studied children's longitudinal progression through programming tasks, arguing that uptake of programming concepts such as symbol recognition and sequencing are tied to developmental readiness. Strawhacker and Bers (2019) examined child-made programs in a children's coding platform for evidence of mental representations related to computational concepts. In addition to finding age-related achievement differences (with older children showing generally higher understanding of the same programming environment over the same number of lessons), they concluded that coding knowledge may link to many other developing knowledge domains, including spatial reasoning and cause-and-effect logic. These studies echo findings from developmental research on achievement gains in a variety of subjects, which may be attributed to generalized, global growth as children progress through natural developmental stages (Campbell et al., 2001). Common findings across these early studies suggest that there may be a developmental or age-graded progression to children's performance on coding assessments, and thus, assessments should be designed with accommodations for these narrow developmental bands.

Design Principles of a Robotic Assessment for PreK

To share useful evidence that can inform learning interventions, we posit that robotic assessments for preschool aged children should capture the child's level of fluency within a coding language. Fluency with a coding language is rooted in an understanding of the language's syntax and grammar and indicates an ability to select and sequence commands in a way that the robot will execute. As with natural language development, fluency with a coding language involves change over time, with children progressively improving their command of the language. Furthermore, the end goal is to be able to use the programming language for creative expression, as another language to express a purposeful or meaningful idea (Bers, 2020). Thus, assessments should investigate this nuance in children's coding performance, as this suggests a shift from syntax and grammar exploration to actual expressive communication through technology.

Informed by prior research (e.g., Flannery and Bers, 2013; Mioduser, Levy, and Tallis, 2009; Relkin et al., 2020), we propose that assessments should be situated within the context of a developmentally appropriate robotic environment, and they should be repeatable to demonstrate change over time. Authentic assessment, a form of testing that uses items and prompts relevant to the learning process, addresses both of these points (Ming, Ming & Bumbacher, 2014; Shaffer & Resnick, 1999). Authentic assessment is inherently repeatable because it involves children judiciously applying their developing skills to negotiate a complex (but realistic) task (see, for example, Chapter 2 of Wiggins, 1998). Computers and new technologies, with their opportunities for connectedness and audience-oriented experience design, are uniquely suited to support authentic assessment (Shaffer & Resnick, 1999). Engaging learners in an authentic assessment experience is another design principle we applied in our current design work.

Design Criteria for the CSA-KIBO

The purpose of the current study is to develop a valid and reliable assessment measure to capture children's understanding of the KIBO robot. In addition to being a useful measure of the level of computational understanding that children can demonstrate through building and programming with KIBO, the assessment is meant to offer insights into children's developmental coding stage as they progress through a curriculum. Based on these needs, and informed by prior research on assessment development, we identified the following design criteria for our assessment.

1. Theoretical alignment with our learning framework of choice

Theoretically, the tool should align with the Coding Stages Framework in order to place children's progression along a predictable developmental trajectory of coding knowledge (Bers, 2020; de Ruiter & Bers, 2021). It should also be informed by the Coding as Another Language pedagogy, to explore children's fluency and expressiveness with the KIBO robotic coding environment (Bers, 2020). To this end, we decided to incorporate a mix of closed- and open-ended items, giving children an ability to demonstrate syntax understanding (closed-ended) and purposeful creation (open-ended) with KIBO.

2. Developmentally appropriate and responsive to children's learning needs

In order to be responsive to young children's limited attention spans, we decided on an ideal time limit of roughly 15-30 minutes per child for individualized assessment with a robotic tool (Mioduser & Levy, 2010; Mioduser, Levy & Talis, 2009; Moyer & Gilmer, 1955; Denner, J., Werner, L., Campe, S., & Ortiz, E. 2014). Additionally, the language and tasks should be appropriate for preschool aged children, to avoid children's limited language ability, developing manual dexterity, and unfamiliarity with jargon artificially limiting results (Sattler, 2014).

We wanted the assessment to be able to be re-administered to show change over time. We aimed to apply a "low floor, high ceiling, wide walls" (Resnick & Robinson, 2017) design model to ensure the test was sensitive enough for novices as well as experienced children (Sattler, 2014). This also led us to explore adaptive administration, in which the test stops after children have demonstrated their level capacity through enough incorrect answers.

3. Ease of administration and interpretation for educators

To best serve a classroom setting, the assessment should also be easy to administer. We aimed to design the CSA-KIBO assessment to be brief (no more than 30 minutes) and easy enough for relatively inexperienced teachers to administer successfully. Additionally, we aimed to keep the assessment simple to score, with a straightforward coding stage output corresponding to descriptive names (e.g., "Emergent Coder") to better convey the meaning of results to teachers and parents (Sattler, 2014).

Coding Stages Framework

The Coding Stages Framework presented by Bers (2020) describes the developmental milestones exhibited by young children ages 4-7 years as they learn and gain mastery with coding, progressing from simple and exploratory behaviors (e.g., naming and recognizing coding command symbols) to more advanced skills and practices (e.g., sequencing commands to create repeating or interactive computational behaviors). Along their developmental pathway, children explore a range of powerful ideas from computer science, including the distinction between hardware and software, the power of sequencing, and the cause-and-effect relationships of conditional statements (e.g., "if-then" or sensor-activated commands). Table 1 summarizes the coding stages, and what behaviors might be observed in children at each stage. As with literacy development, children progress individually and dynamically through these stages, independent of age but related to their general development and readiness to explore coding tasks. Children may exhibit different levels within the course of one activity, or show different mastery depending on the learning context, etc. As de Ruiter and Bers explain, "the developmental progression between coding stages is not always sequential, orderly, and cumulative, [although] the coding stages have some hierarchy based on syntax and grammar – mastery of simpler structures/commands (e.g., start/end) occurs before mastery of more complex structures (e.g. if statement or repeat loops). [...] In line with the Coding as Another Language (CAL) approach, the Coding Stages framework also includes meaningfulness and expression at every stage, as well as in the final stage Purposefulness. Thus, at each stage it is observed if and how children are able to use coding in expressive ways" (de Ruiter & Bers, 2021, p. 5).

The term "stages" is inspired by the universal stages of development posited by Piaget (Piaget, 1963). Coding stages refer to learned skills, and are indicators of growth in mastery of creative, expressive coding that occur when children learn a developmentally-appropriate coding language.

Table 1 – Coding Stages and Corresponding Computational Concepts Mastered by Children (Reprinted with author permission from de Ruiter & Bers, 2020)

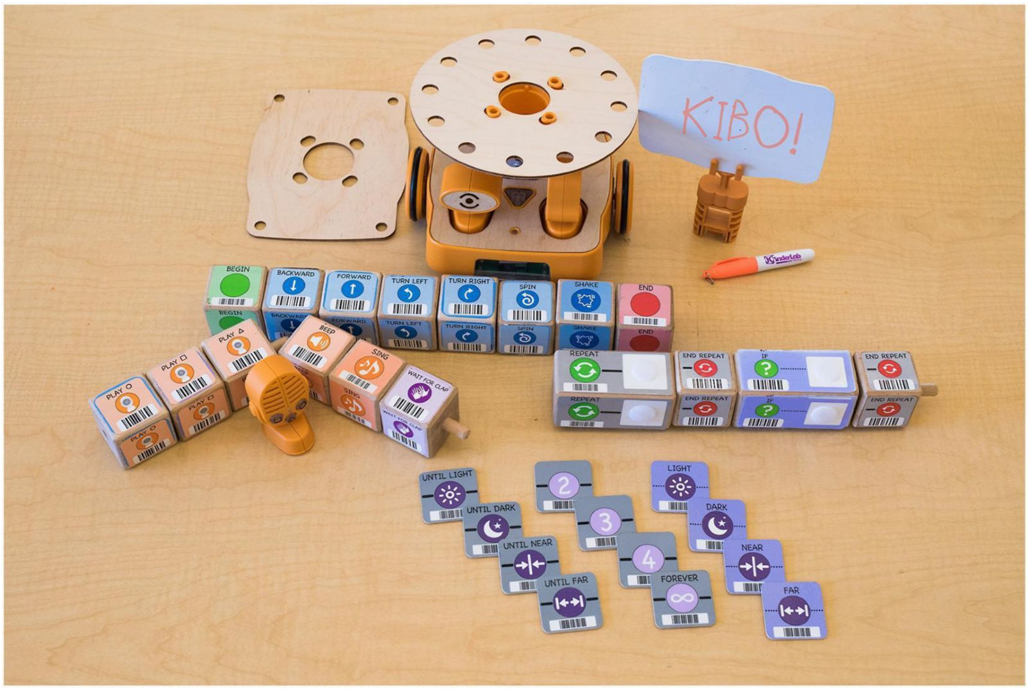
Coding Stage	Description
1. **Emergent**	• The child recognizes that technologies are human-engineered and are designed with a variety of purposes. • The child understands the concept of symbolization and representation (i.e., a command is not the behavior, but represents the behavior). • The child understands what a programming language and the purpose of its use is (knows that a basic sequence and control structure exists). • The child is familiar with the basics of the interface (turn the tool on and off and correctly interact). • This is a beginner's stage.
2. **Coding and Decoding**	• The child understands that sequencing matters and that the order in which commands are put together generates different behaviors. • The child has learned a limited set of symbols and grammar rules to create a simple project. • The child can correctly create simple programs with simple cause and effect commands. • The child can identify and fix grammatical errors in the code. • The child performs simple debugging through trial and error. • The child engages in goal-oriented command exploration. • The most growth can be seen at this stage. Children learn the basics of the programming language and understand it can serve to create projects of their choice.
3. **Fluency**	• The child has mastered the syntax of the programming language and can correctly create programs. • The child is personally motivated to create complex programs. • The child understands how to distinguish and fix logical errors in the code. • The child is beginning to be strategic in debugging. • This stage is characterized by the child moving from a "learning to code" to a "coding to learn" creative stance.

Coding Stage	Description
4. New Knowledge	• The child understands how to combine multiple control structures and create nested programs that achieve complex sequencing. • The child engages in more goal oriented logical exploration with their programs. • The child is personally motivated to create complex programs. • The child is strategic in debugging and has developed strategies. • The child learns how to learn new commands or novel uses of the interface. • This stage is characterized by the child's ability to use their knowledge to create a personally meaningful project and if needed, acquire new knowledge on her own to meet the demands of the project.
5. Purposefulness	• The child can skillfully create complex programs for their needs and purposes. • The child understands how to analyze, synthesize, and translate abstract concepts into code and vice versa. • The child is able to identify multiple ways to translate abstract concepts into code. • The child understands how to create programs that involve user's input. • The child can create multiple programs that interact with one another. • The child can debug multiple control structures. • This stage is characterized by the child being able to code in a rapid and efficient manner at high levels of abstraction, requiring skill and flexibility and applying those skills to create a personally meaningful project. A child who reaches this stage has mastered all of the commands, grammar and syntax, of the programming language and has the ability to express herself through the project they create.

KIBO Robot

The KIBO robot kit is a screen-free educational platform that is designed for young children in preschool through early elementary school. It allows children to explore coding, engineering, and robotics in a playful, developmentally appropriate way (Figure 1). KIBO was initially designed in 2011 as a research prototype (NSF DRL-1118897) and was launched as a commercially-available product in 2013. Over a decade of research has been conducted on KIBO, involving thousands of children and teachers and families from around the US and the world (Albo-Canals et al., 2018; Bers 2020). KIBO was iteratively designed, informed by numerous focus groups and empirical experiments with children, caregivers, and teachers. The robot is intuitively easy to physically build and creatively program with coding blocks that encourage syntactically correct connections (e.g., no blocks can be added before a BEGIN block or after an END block), and hardware that can attach crafts, recycled materials, and other familiar early childhood building materials to the robot (KinderLab Robotics, 2021).

Figure 1 – The programmable KIBO robot



The programmable KIBO robot (top center), wooden programming blocks to give coding instructions, the KIBO (middle), parameter stickers (e.g., Until Near, Forever) to define repeat and conditional statements (bottom), module attachments (e.g., sound recorder) and sensors (light, sound, distance) to make KIBO interactive (attached to KIBO and near coding blocks), and attachable art platforms to customize physical aesthetic of robot (top left, top right, and attached to KIBO) (Reprinted with author permission from Relkin et al., 2021)

KIBO is also screen-free, in line with research that tangible interfaces may be more beneficial for engaging young children in early coding experiences (e.g., Pugnali, Sullivan & Bers, 2017; Strawhacker & Bers, 2019). KIBO's screen-free approach has made it accessible for audiences beyond the U.S. K-2 classroom. Researchers have found that children diagnosed with autism spectrum disorder, students in various countries, and children coding with family members in informal learning spaces outside of school are all able to learn to code and explore computational thinking concepts like modularity, and control structures with the KIBO robot (Albo-Canals et al., 2018; Elkin et al., 2018; Relkin et al., 2020; Strawhacker & Bers, 2019; Sullivan, et al., 2018). KIBO has been empirically shown to increase coding and computational thinking skills. For example, a study was conducted in which 848 students were given an intergraded KIBO robotics coding and literacy curriculum called CAL-KIBO (Coding as Another Language - KIBO Robotics). Children's computational thinking skills as measured by the *TechCheck* assessment improved significantly after seven weeks of engagement with KIBO whereas a control group who participated in typical classroom activities without coding did not improve significantly (Relkin et al., 2021; Bers et al., 2021). Research has confirmed that KIBO supports PreK-2nd grade children's learning of discipline-specific content in foundational areas of math and literacy, in addition to computational thinking skills (Albo-Canals et al., 2018; Bers, 2020). Beyond coding, engaging with KIBO has been shown to have positive impacts on children's social-emotional development under the Positive Technological Development and Palette of Virtues Frameworks (Bers, 2020, 2021; Relkin et al., 2020).

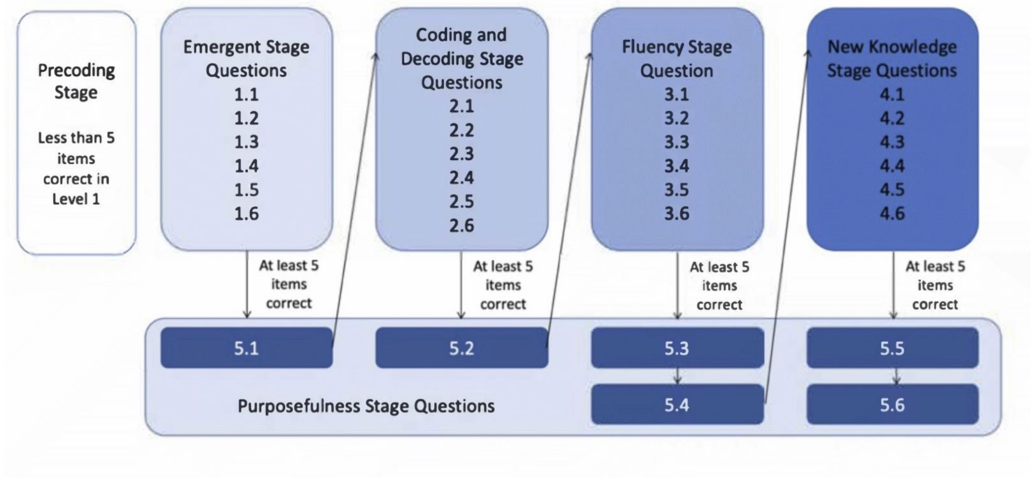
The KIBO Coding Stages Assessment

The KIBO Coding Stages Assessment (CSA-KIBO) is a 30-item assessment designed to determine children's level of performance with the KIBO robotic kit. After formal validation, CSA-KIBO training, certification, and the downloadable assessment will be available to request at the following website: https://sites.tufts.edu/devtech/csa-kibo/. Assessments are administered one-on-one, with a researcher voicing questions and recording answers, and a child using a KIBO robot and a selection of parts and coding blocks to physically demonstrate answers. Researchers provide a predetermined selection of parts at the beginning of each level of the assessment, gradually increasing the amount and variety of parts available to the child.

In keeping with our design criteria of theoretical alignment with the Coding Stages Framework, each item assesses an aspect of one of the coding stages described by Bers (2020). (See de Ruiter & Bers, 2021, for further discussion of how the stage hierarchy impacts the CSA-KIBO stage item progression and scoring system.) Figure 2 shows the progression of items throughout the assessment, and how items are related to elements of the Coding Stages Framework. Children answer questions within a single stage of the assessment at once, with the exception of the Purposefulness stage. Because of the theoretical perspective within the Coding Stages Framework that children can exhibit purpose in their coded creations regardless of coding fluency, open-ended Purposefulness items were administered after each stage. For example, Purposefulness item 5.1, asks children to choose the KIBO block that they think represents a dance movement, and requires only an Emergent level of coding knowledge to correctly respond. Although children respond to Purposefulness items after each successful stage, scoring a Purposefulness stage is only possible for children who surpass the New Knowledge stage.

In the context of this assessment study, children who test below the Emergent stage (meaning they have little to no understanding of technologies as human created and engineered, or how a system of coded symbols could control a machine) are classified as "Precoders" (see Figure 2). While not technically a stage in the Coding Stages Framework, Precoder is a useful category to indicate a very naive level of technological and programming awareness.

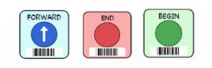
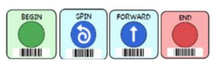
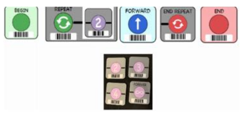
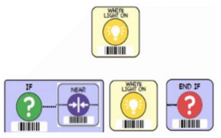
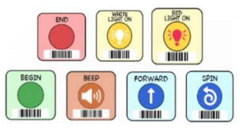
Figure 2 – CSA-KIBO Test Structure and Item Alignment with Coding Stages Framework



The CSA-KIBO is also designed to respond to children's developmental needs in terms of attention, focus, and challenge. Questions are written for PreK comprehension levels, with simple prompts that can be answered using language or physical demonstration (example question: "Can you point to the coding block that will make KIBO move forward?"). Researchers can repeat questions up to three times and may name the parts that children point to throughout the assessment. Total length of time to conduct the assessment is limited to focus children's attention, and occasional breaks are available throughout the assessment.

Our final design criteria required that the assessment be easy to administer and results easy to interpret. Administration is simplified using a script for questions with visuals to show which parts should be displayed in front of children, and brief explanations of satisfactory and unsatisfactory answers (all items are scored as a binary "correct or incorrect"). Results show the child's threshold stage (i.e., the highest stage where they score at least 5 out of 6 correct answers). A child must correctly answer 5 out of 6 questions in a stage to progress to the next one. Correct and incorrect response definitions were developed for each item, as well as acceptable paraphrased prompts if children did not understand the question. For example, in answer to the question "*How do you turn KIBO on?*", an acceptable paraphrase prompt was "*I want to turn on KIBO. What do I need to do?*". Acceptable correct answers included pressing or describing the triangle-shaped ON button, and incorrect responses included pressing or describing somewhere else other than the triangle button, saying "I don't know", requesting to skip the question, or offering no answer. For a more complete illustration of sample items with correct and incorrect responses, see Table 2. A student who scores less than 5 correct in the first Emergent stage of questions is automatically categorized as a Precoder. In order to operationalize stages for teachers, each stage corresponds with the coding stage captured by most of its questions, so stage 1 denotes an Emergent coder, stage 2 indicates Coding and Decoding, etc. These titles are self-descriptive, and teachers can also seek more information by referring to the original Coding Stages Framework, developed for practitioners as well as researchers (Bers, 2020).

Table 2 – Sample CSA-KIBO Items with Correct and Incorrect Response Criteria.

Stage	Sample Item	Visual Prompt (KIBO blocks on table in front of child)	Rephrase	Correct Responses	Incorrect Responses
Emergent	"Take a look at these blocks. Which block do you always need to start your program?"		"Which block goes first?"	Child points to or describes green Begin block	Child does not point to or describe green Begin block, says I don't know/skip, or no answer
Coding and Decoding	"Look at this program. I want KIBO to first move forward and then spin. How can you make this program?"		"How can you change this program so that KIBO moves forward before it spins?"	Child switches order of Spin and Forward blocks, or scans Forward before Spin	Child does not change order of program or scanning, says I don't know/skip, or no answer
Fluency	"Look at this program. Please make KIBO move forward 3 times instead of 2."		"Please fix my program so KIBO moves forward 3 times."	Child replaces 2 parameter with 3 parameter in Repeat Loop, or scans Forward 3 times	Child does not change or scan program correctly, says I don't know/skip, or no answer
New Knowledge	Point to light block and say "What is the difference between this block.....," Put if and end if around light block and say "...and this set of blocks?"		"What changes when the light block has these blocks around it?"	Child verbally answers that one block makes KIBO light up and the group of blocks makes KIBO light up only if it is near something	Child gives another explanation as to why they're different, says there is no difference, says I don't know/skip, or no answer
Purposeful-ness	"Pretend KIBO is a car. Make a program that makes KIBO drive and honk at other cars."		"Make a program to make KIBO drive and honk at other cars."	Child either uses a motion and a Beep block, or uses other blocks and explains their idea	Child arranges blocks with no explanation, says I don't know/skip, or no answer

Investigating Assessment Administration

We conducted a study to address the following research question: How does the mode of administration of the CSA-KIBO robotics assessment impact performance among preschool students?

By observing students' behavior when carrying out this investigation, we incidentally gathered important information related to the developmental appropriateness of the assessment and student engagement during administration, as is reported below.

Method

Participants & Context

The 151 preschool students who participated in this study were recruited from three sites: five non-profit Head Start Preschools in Missouri (Missouri Head Starts), one non-profit preschool in the Greater Boston area serving children from families experiencing or at risk of homelessness (Boston-area non-profit), and one tuition-based lab school in the Greater Boston Area following a local public school district curriculum (Boston-area lab school). Our sample includes organizations serving preschool children from families of diverse ethnic, socioeconomic, and linguistic backgrounds. All children were between the ages of 2.9 - 5.3 years and had little or no prior experience with the KIBO robotics kit.

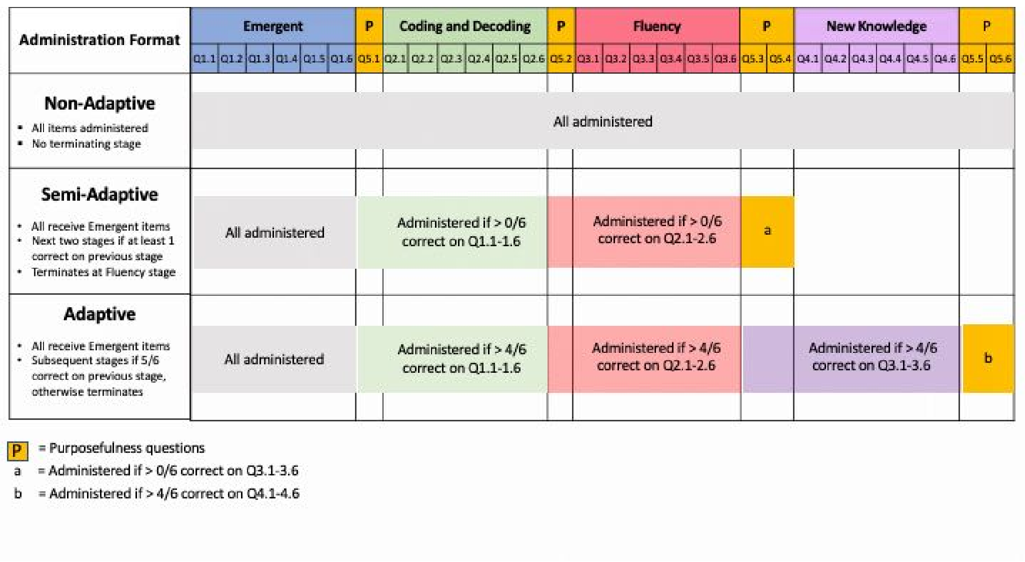
Procedure

After obtaining informed consent from families, children were invited to leave their classroom during free-play time to play coding games one-on-one with a trained researcher who administered the CSA-KIBO. If children opted not to participate, or opted to leave the assessment partway through, they were allowed to leave with no consequences, and were invited back on at most one other occasion on a different day within the month.

Three forms of administration were used to explore administration types. At one site, we administered our theory-informed *adaptive* version of the assessment, which required 5 out of 6 correct answers to progress to the next stage, or automatically terminated the assessment (in other words, the assessment terminated at the stage beyond threshold performance). We predicted that, according to the Coding Stages Framework, PreK children with no (or limited) prior robotic experience would test in the Precoding stage (0-4 correct items) and so the computer-based test was programmed to terminate once children reached their threshold stage. At two separate sites, we simultaneously explored a *non-adaptive* version of the CSA-KIBO (terminating after completion of all possible questions), to assess whether at baseline, PreK children could perform on any of the questions we categorized as more advanced (based on prior research with early childhood coding curricula and performance). We subsequently altered our non-adaptive administration to be *semi-adaptive* (terminating at the Fluency stage conditional on scoring at least one question correct on each of the previous levels), as a more developmentally appropriate response to children's patience and behavioral tolerance with advanced KIBO questions beyond their coding stage. In this semi-adaptive version of the assessment, the highest attainable stage was artificially capped at the Fluency stage.

Although the semi-adaptive test continues to ask items beyond the child's threshold performance stage (the highest stage on which they score >4 correct), children are still classified at their threshold. In scoring all administration formats, students were classified at the highest stage when they correctly answered 5 out of 6 questions, with students earning a score of Precoder for less than 5 correct items in the Emergent stage. Figure 3 shows the highest possible stage and testing constraints of each CSA-KIBO administration format.

Figure 3 – Highest Attainable Stage for CSA-KIBO Administration Formats



Administration Format	Emergent	P	Coding and Decoding	P	Fluency	P	P	New Knowledge	P
	Q1.1 Q1.2 Q1.3 Q1.4 Q1.5 Q1.6	Q5.1	Q2.1 Q2.2 Q2.3 Q2.4 Q2.5 Q2.6	Q5.2	Q3.1 Q3.2 Q3.3 Q3.4 Q3.5 Q3.6	Q5.3	Q5.4	Q4.1 Q4.2 Q4.3 Q4.4 Q4.5 Q4.6	Q5.5 Q5.6
Non-Adaptive ▪ All items administered ▪ No terminating stage	All administered								
Semi-Adaptive • All receive Emergent items • Next two stages if at least 1 correct on previous stage • Terminates at Fluency stage	All administered		Administered if > 0/6 correct on Q1.1-1.6		Administered if > 0/6 correct on Q2.1-2.6	a			
Adaptive • All receive Emergent items • Subsequent stages if 5/6 correct on previous stage, otherwise terminates	All administered		Administered if > 4/6 correct on Q1.1-1.6		Administered if > 4/6 correct on Q2.1-2.6			Administered if > 4/6 correct on Q3.1-3.6	b

P = Purposefulness questions
a = Administered if > 0/6 correct on Q3.1-3.6
b = Administered if > 4/6 correct on Q4.1-4.6

A total of 151 preschool students were administered the CSA-KIBO assessment prior to formal coding instruction, with different formats explored at various sites (see Table 3). 29 students were administered the non-adaptive version and received all questions on the assessment regardless of performance. 65 students received the semi-adaptive version of the assessment, which allowed 5 incorrect answers per stage and was capped at the Fluency stage. Finally, 57 students received the adaptive version of the assessment, which stopped once children gave 2 or more wrong answers on a given stage of the assessment.

Table 3 – Distribution of Administration Formats across Sites and Students.

Administration Format	Sites (Students)
Adaptive	Boston-area non-profit (*n* = 57)
Semi-adaptive	Missouri Head Starts (*n*=65)
Non-adaptive	Missouri Head Starts (*n* = 18), Boston-area lab school (*n* = 11)

Based on research with earlier assessment tools to measure young children's programming skills, and the fact that our sample comprised children at the youngest end of our intended age range, with limited prior exposure to coding tools, we hypothesized that our sample would demonstrate the Precoding stage of proficiency.

Results

CSA-KIBO Administration

For the non-adaptive format of the assessment, administration was not completed in 12 out of the 29 cases (41.37%) because the child either indicated they wanted to stop or skip questions or time ran out. In comparison, none of the children administered the adaptive format and only 1 child who received the semi-adaptive version indicated that they wanted to stop. The average time it took children to complete the CSA-KIBO was *M* = 25.42 minutes, *mdn*= 26.18 minutes for the non-adaptive assessment, *M* = 18.03 minutes, *mdn*= 12.93 minutes for the semi-adaptive assessment and *M* = 11.51, *mdn*= 6.63 minutes for the adaptive version of the assessment. Note this time included some children taking breaks during testing for up to 5 minutes.

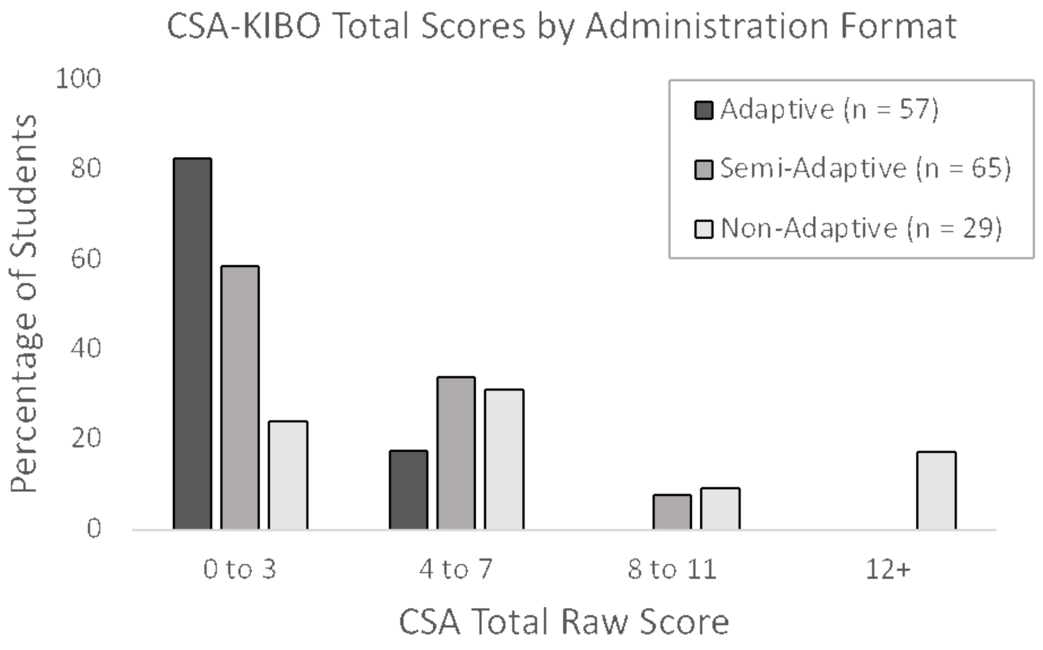
Descriptives

For all formats of administration, each question answered correctly scored 1 point. There was a maximum of 30 questions asked for 30 possible points when children were administered the non-adaptive and adaptive formats and a maximum of 20 questions for 20 possible points when children were administered the semi-adaptive format. Table 4 shows descriptive statistics for each type of CSA-KIBO administration. Scores ranged from 0-19 points. Figure 4 shows the percentage of students' raw CSA-KIBO total scores by each format of the assessment. The non-adaptive administration format does provide a greater range of raw-score outcomes and higher stage classifications. However, this greater discrimination comes at the price of much greater attrition than the other two administration formats.

Descriptive Statistics for students' scores by administration type

CSA-KIBO type	N	Mean	SD	Mode	Median	Min	Max	Skewness
Adaptive	57	2.37	1.28	2	2	0	7	0.62
Semi-adaptive	65	3.54	2.63	3	3	0	9	0.2
Non-adaptive	29	7.41	4.63	4, 6	7	0	19	0.64

Figure 4 – Percentage of participants by total CSA-KIBO score for the three administration formats



Coding Stages Assignments

Figure 2 shows how the coding stage that each child is assigned to is derived from the students' performance. As Table 5 shows, regardless of the format administered, the majority of students were placed in the Precoding stage. This is because most students responded correctly to less than 5 questions on the first stage (see Table 5). A majority of participants were placed at the floor of the assessment regardless of the mode of administration. This indicates that the level of difficulty was too high (as predicted). We explored whether children could answer any of the more advanced questions by administering every question to the children (non-adaptive format). All questions in the "New Knowledge" (Q4.1 through Q4.6) and the more advanced "Purposefulness" (Q5.3 through Q5.6) stages were answered correctly by 2 (5%) or less of children. The assessment is therefore performing as expected for a coding naïve sample.

Table 5 – Percentage of students assigned to each Coding Stage

Format	Precoding	Emergent	Coding and Decoding	Fluency	New Knowledge	Purposeful-ness
Adaptive (*n* = 57)	98.24%	1.75%	0	0	0	0
Semi-adaptive (*n* = 65)	96.92%	3.07%	0	0	0	0
Non-adaptive (*n* = 29)	79.31%	10.34%	10.34%	0	0	0

Discussion

This exploratory study using the CSA-KIBO in preschool aged coding-naïve children showed differences in outcome related to the format of administration. Completion of the assessment was more successful when adaptive or semi-adaptive formats were implemented. A possible floor effect was present with all formats of administration, which is not unexpected given the focus of the instrument on coding skills and the lack of coding experience in this design study cohort.

Impact of Administration Format on Assessment outcomes

Our research question asked, "How does the mode of administration (adaptive, semi-adaptive or non-adaptive) impact CSA-KIBO results in preschool students?"

We found the non-adaptive assessment took significantly longer (about 25 minutes on average) and nearly half of the children requested to end the assessment early, compared with no drop-outs from the adaptive version. The semi-adaptive administration (average 12-18 minutes) cut down on time of administration and only had a single child request to end early. In comparison, the adaptive version of the assessment significantly reduced the average test time (7 minutes) and showed very similar Coding Stage placement and total raw scores to children who took the slightly longer semi-adaptive assessment.

It is likely that if more questions are administered to children, they have a greater opportunity of getting a higher score which is what the results indicated. However, in this study we cannot say with certainty whether differences in the mean scores represent actual differences associated with the three administration methods or factors such as the differences in the schools from which the respective samples were drawn.

In deciding which administration format to recommend, rather than base our conclusion on score distributions, we have been persuaded by the advantages of the adaptive version of the assessment in terms of its brevity and ease of administration. There may be other contexts in which the non-adaptive or semi-adaptive versions offer advantages (such as with older and/or children with prior coding experience). While this study illustrates the utility of the adaptive version of the assessment for coding-naive preschoolers, additional studies will be needed to establish how the instrument performs with more advanced and older children.

Developmental Appropriateness of the CSA-KIBO

In addition to addressing our question about administration format, we examined incidental observations about student and assessor experience while taking the assessment.

Children showed engagement and enthusiasm during early items on the assessment, although attention spans waned during later items, particularly during the non-adaptive administration (longest assessment format). We attribute this to the lengthy administration time and inappropriateness of the advanced questions for novice robotics students. In the shortened adaptive form, the assessment was sufficiently engaging and appropriate for all children in the sample to complete the required questions.

From an anecdotal perspective, researchers noted that the adaptive format also took less time per child to administer and required less physical set-up of the materials and computerized form to collect responses.

Children in our sample performed predominantly at a low “Precoding” stage. This was consistent with our prediction, based on the sample’s low level of prior experience with KIBO. We interpret this to mean that the CSA-KIBO was unlikely to register “false positives,” that is, to incorrectly categorize inexperienced children at a higher coding stage. Thus, our empirical results are consistent with what we would expect based on our theoretical framework and intended design.

Limitations and Future Work

None of the students in the present study surpassed the “Coding and Decoding” stage in our scoring system. This finding is positive for establishing that the CSA-KIBO is appropriate for the youngest and least-experienced members of our intended audience, but many questions remain about how the assessment fares with children who belong in more advanced coding stages. Future work will need to explore the CSA-KIBO with older children, who are likely to score higher than younger peers based on general developmental growth, as in other assessment research (e.g., Campbell et al., 2001). Additionally, we will seek to pilot with children who have more diverse experiences with robotics beyond pre-exposure and more diverse background of sample.

There is also no data from this study on whether results can be interpreted with ease by families and teachers. This limitation will be explored in future work.

A long-term goal of this project is to create a valid, reliable, and robust assessment metric for easily evaluating a student’s developmental coding level. This would ideally include longitudinal assessments of the same children at various time points, for example pre- and post-test data from a robotics intervention, or multiple time-point assessments of children over a school year. With these longitudinal data, we plan in future to conduct validation investigations involving item response theory and classical test theory analyses to determine the psychometric properties of the CSA-KIBO assessment.

Conclusion

The research sought to investigate the viability of a novel KIBO coding assessment tool, aligned with Bers' (2020) Coding Stages Framework, for use with our intended audience of very young children. In the current study, we applied a design research approach to pilot this assessment with children at the youngest and least experienced end of our intended audience of children aged 3-9 years and explored a range of administration formats to learn the relative impact on implementation ease and evaluative success.

Of the three administration formats explored, the adaptive format, the shortest and most convenient method, showed the best combination of effectively maintaining children's interest and engagement for the duration of the session and keeping administration time brief, while accurately capturing the expected stage for preschool children with little or no coding experience. Our results using the adaptive method were consistent with our predictions based on the coding stages framework that coding naïve children should be in the Precoding stage. The non-adaptive and semi-adaptive formats may be useful in certain contexts with the understanding that attrition may be higher than with the adaptive format.

Design research is often described with expressions like "flying the plane while building it," to indicate the iterative and responsive nature of this type of research method. While the purpose of the design research method is to develop new methods, models, and tools to further learning and research, an equally important and often overlooked aspect of design research is the process of refining the implementation approach used to test designs, informed by evidence from successive trials (Edelson, 2002). From a design perspective, we consider the major contribution from this study to be a focused exploration of assessment administration as a critical aspect of assessment design. Based on our findings, a simple adaptive administration format showed sufficient success to be used in future trials of the CSA-KIBO with a larger, more diverse sample. In future work, as we work to validate the assessment CSA-KIBO with data collected from a wide array of learners before and after they engage in an implementation of the CAL curriculum, we feel confident in applying our research-informed adaptive administration format.

Acknowledgements

The research team gratefully acknowledges generous funding support from the LEGO Foundation for the Playful Engineering-Based Learning project (LEGO Foundation COVID Recovery Grant). We also thank all the members of the DevTech Research Group, especially Madhu Govind and Apittha Unahalekhaka, who contributed to the collection and analysis of these data. Finally, we are grateful to the St. Louis Missouri Urban League, especially Dr. Gwendolyn Diggs and Dr. Miranda Avant-Elliot, as well as the Eliot-Pearson Children's School, the Horizons for Homeless Children preschool center, the Head Start preschool centers, and all the children, families, teachers, and administrators who hosted us and made this study possible.

References

Albo-Canals, J., Barco, A., Relkin, E., Hannon, D., Heerink, M., Heinemann, M., Leidl, K. & Bers, M. (2018). A Pilot Study of the KIBO Robot in Children with Severe ASD. *International Journal of Social Robotics*, 10(3), 371-383. Advance online publication. doi:10.1007/s12369-018-0479-2

Anderson, T. & Shattuck, J. (2012). Design-based research: A decade of progress in education research? *Educational researcher*, 41(1), 16-25.

Bers, M. (2020). *Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom*, Second Edition. New York, NY: Routledge Press.

Bers, M. (2021). Coding, robotics and socio-emotional learning: developing a palette of virtues PIXEL-BIT. *Revista de Medios y Educación*, 62, 309-322.

Bers, M., Govind, M. & Relkin, E. (2021) Coding as Another Language: Computational Thinking, Robotics, and Literacy in First and Second Grade. In Ottenbreit-Leftwich, A. & Yadav, A. (Eds.) *Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions*. ACM and the Robin Hood Learning + Technology Fund, New York, NY

Campbell, F. A., Pungello, E. P., Miller-Johnson, S., Burchinal, M. & Ramey, C. T. (2001). The development of cognitive and academic abilities: Growth curves from an early childhood educational experiment. *Developmental Psychology*, 37(2), 231–242. https://doi.org/10.1037/0012-1649.37.2.231

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X. & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers and Education*, 109, 162-175.

Clements, D. H. & Sarama, J. (2004). Learning trajectories in mathematics education. *Mathematical thinking and learning*, 6(2), 81-89.

Code.org, CSTA & ECEP Alliance. (2021). *2021 State of computer science education: Accelerating action through advocacy*. Retrieved from https://advocacy.code.org/stateofcs

Denner, J., Werner, L., Campe, S., & 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.

de Ruiter, L. E. & Bers, M. U. (2021). The Coding Stages Assessment: development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language. *Computer Science Education*, 1-30.

Edelson, D. C. (2002). Design research: What we learn when we engage in design. *The Journal of the Learning sciences,* 11(1), 105-121.

Elkin, M., Sullivan, A. & Bers, M. U. (2018). Books, Butterflies, and 'Bots: Integrating Engineering and Robotics into Early Childhood Curricula. In L. English and T. Moore (Eds.), *Early Engineering Learning* (225-248). Singapore: Springer. doi:10.1007/978-981-10-8621-2_11

Flannery, L. P. & Bers., M. U. (2013). Let's dance the "robot hokey-pokey!": children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education*, 46(1), 81–101. http://ase.tufts.edu/DevTech/publications/JRTE-robot-hokey-pokey.pdf

Fraillon, J., Ainley, J., Schulz, W., Duckworth, D. & Friedman, T. (2019). *IEA international computer and information literacy study 2018 assessment framework* (p. 74). Springer Nature.

Hansen, M., Levesque, E., Valant, J. & Quintero, D. (2018). *The 2018 Brown Center report on American education: How well are American students learning*. Washington, DC: The Brookings Institution.

Hassenfeld, Z. R., Govind, M., de Ruiter, L. E. & Bers, M. U. (2020). If You Can Program, You Can Write: Learning Introductory Programming Across Literacy Levels. *Journal of Information Technology Education: Research*, 19, 65-85 https://doi.org/10.28945/4509

Hsu, T. C., Chang, S. C. & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, 126, 296-310.

Hudson, R. F., Isakson, C., Richman, T., Lane, H. B. & Arriaza-Allen, S. (2011). An examination of a small-group decoding intervention for struggling readers: Comparing accuracy and automaticity criteria. *Learning Disabilities Research & Practice*, 26(1), 15–27. https://doi.org/10.1111/j.1540-5826.2010.00321.x

International Society for Technology in Education. (2007). *Standards for technological literacy*. https://www.iteea.org/File.aspx?id=67767&v=b26b7852

Janveau-Brennan, G. & Markovits, H. (1999). The development of reasoning with causal conditionals. *Developmental Psychology,* 35(4), 904.

Kazakoff, E. R. & Bers, M. U. (2014). Put your robot in, put your robot out: Sequencing through programming robots in early childhood. *Journal of Educational Computing Research*, 50(4), 553-573.

KinderLab Robotics, (2021). *Awards*. Retrieved from: https://kinderlabrobotics.com/awards/

Lee, M. J. & Ko, A. J. (2014, July). A demonstration of gidget, a debugging game for computing education. In 2014 *IEEE Symposium on Visual Languages and Human-Centric Computing* (VL/HCC) (pp. 211-212). IEEE.

Liao, Y. K. C. & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: A meta-analysis. *Journal of educational computing research*, 7(3), 251-268.

Metz, S. S. (2007). Attracting the engineering of 2020 today. In R. J. Burke, M. C. Mattis & E. Elgar (Eds.), *Women and minorities in science, technology, engineering and mathematics: Upping the numbers* (pp. 184-209). Northampton, MA: Edward Elgar Publishing.

Ming, V., Ming, N. & Bumbacher, E. (2014). *Aligning learning with life outcomes through naturalistic assessment*. Socos LLC white paper.

Mioduser, D. & Levy, S. T. (2010). Making sense by building sense: Kindergarten children's construction and understanding of adaptive robot behaviors. *International Journal of Computers for Mathematical Learning*, 15(2), 99-127.

Mioduser, D., Levy, S. T. & Talis, V. (2009). Episodes to scripts to rules: Concrete-abstractions in kindergarten children's explanations of a robot's behavior. *International Journal of Technology and Design Education*, 19(1), 15-36.

Moyer, K. E. & Gilmer, B. V. H. (1955). Attention spans of children for experimentally designed toys. *The Journal of genetic psychology*, 87(2), 187-201.

Myers, E. K. (2021). The Role of Executive Function and Self-Regulation in the Development of Computational Thinking. In M. Bers (Ed.) *Teaching Computational Thinking and Coding to Young Children (pp. 64-83). IGI Global.* https://doi.org/10.4018/978-1-7998-7308-2.ch004.

NAEYC & Fred Rogers Center for Early Learning and Children's Media. (2012). *Technology and interactive media as tools in early childhood programs serving children from birth through age 8. Joint position statement.* www.naeyc.org/files/naeyc/file/positions/PS_technology_WEB2.pdf

Piaget, J. (1963). The attainment of invariants and reversible operations in the development of thinking. *Social research*, 283-299.

Pugnali, A., Sullivan, A. & Bers, M.U. (2017). The Impact of User Interface on Young Children's Computational Thinking. *Journal of Information Technology Education: Innovations in Practice*, 16, 172-193.

Relkin, E., de Ruiter., L., Bers, M. U. (2020). TechCheck: Development and Validation of an Unplugged Assessment of Computational Thinking in Early Childhood Education. *Journal of Science Education and Technology*. DOI: 10.1007/s10956-020-09831-x

Relkin, E., de Ruiter, L.E., Bers, M.U. (2021). Learning to Code and the Acquisition of Computational Thinking by Young Children. *Computers & Education, 169*. https://doi.org/10.1016/j.compedu.2021.104222

Resnick, M. & Robinson, K. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. MIT press.

Royal Society. (2017). *After the reboot: Computing education in UK schools*. Policy Report.

Sattler, J. M. (2014). *Foundations of behavioral, social and clinical assessment of children*. Jerome M. Sattler, Publisher

Shaffer, D. W. & Resnick, M. (1999). " Thick" authenticity: New media and authentic learning. *Journal of interactive learning research*, 10(2), 195-216.

So, H. J., Jong, M. S. Y. & Liu, C. C. (2020). Computational thinking education in the Asian Pacific region. *The Asia-Pacific Education Researcher*, 29(1), 1-8.

Strawhacker, A. L. & Bers, M. U. (2015). "I want my robot to look for food": Comparing children's programming comprehension using tangible, graphical, and hybrid user interfaces. *International Journal of Technology and Design Education*, 25(3), 293-319. doi:10.1007/s10798-014-9287-7

Strawhacker, A. & Bers, M. U. (2019). What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development,* 67(3), 541-575.

Sullivan, A. & Bers, M.U. (2018). Investigating the use of robotics to increase girls' interest in engineering during early elementary school. *International Journal of Technology and Design Education, 29,* 1033-1051. doi:10.1007/s10798-018-9483-y

Svensson, A. K. (2000). Computers in school: Socially isolating or a tool to promote collaboration? *Journal of Educational Computing Research*, 22(4), 437-453.

U.S. Department of Education & U.S. Department of Health and Human Services. (2016). Early learning and educational technology policy brief. https://tech.ed.gov/earlylearning

Vegas, E. & Fowler, B. (2020). *What do we know about the expansion of K-12 computer science education?* Brookings. Retrieved February 2, 2021.

Vizner, M. (2017). *Big Robots for Little Kids: Investigating the role of scale in early childhood robotics kits.* (Unpublished master's thesis). Tufts University, Medford, MA.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P. & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, 22(2), 445-468.

Wiggins, G. (1998). Educative Assessment: Designing Assessments to Inform and Improve Student Performance. San Francisco: Jossey-Bass.

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The link magazine*, 6, 20-23.

Zviel-Girshin, R., Luria, A. & Shaham, C. (2020). Robotics as a tool to enhance technological thinking in early childhood. *Journal of Science Education and Technology, 29*(2), 294-302.

About the Authors

Amanda Strawhacker (amanda.strawhacker@tufts.edu) is the Associate Director of the Early Childhood Technology (ECT) Graduate Certificate Program at Tufts University's Eliot-Pearson Department of Child Study and Human Development. Her work involves teaching, developing curriculum, and professional development around educational technology. Prior to her role at ECT, Amanda was a Ph.D. student at the DevTech Research Group. She has contributed to the research and development of several technologies including the ScratchJr programming app, the KIBO robotics kit, the Early Childhood Makerspace at Tufts, and most recently the CRISPEE bioengineering kit, a collaborative research project between DevTech and the

Human-Computer Interaction Lab at Wellesley College about engaging children in playful learning about bioengineering and bioethics. Amanda is a triple-Jumbo who finished her B.A. in Biological Anthropology at Tufts in 2011 and her Master's and Ph.D. in Child Study and Human Development at Eliot-Pearson in 2020. She is a two-time winner of the Eliot-Pearson Research-Practice Integration Award, and was a speaker with TEDxYouth@BeaconStreet on her research with bioengineering in early childhood. She is passionate about engaging young children in playful, positive, and developmentally appropriate STEAM experiences.

Emily Relkin (emily.relkin@tufts.edu) is a Ph.D. candidate at the Eliot-Pearson Department of Child Study and Human Development at Tufts University and is a member of the DevTech Research Group. Emily received her B.A. from Muhlenberg College in Psychology and her M.A. from Tufts University in Child Study and Human Development. Her research focuses on understanding and assessing the development of computational thinking abilities in young children. She developed and validated *TechCheck*, a novel unplugged computational thinking assessment for 3-9-year-olds that has been administered to thousands of children around the world.

Marina Umaschi Bers (marina.bers@tufts.edu) is professor at the Eliot-Pearson Department of Child Study and Human Development with a secondary appointment in the Computer Science Department at Tufts University. She heads the interdisciplinary Developmental Technologies research group. Her research involves the design and study of innovative learning technologies to promote children's positive development. She also developed and serves as director of the on-line blended graduate certificate program on Early Childhood Technology at Tufts University. Prof. Umaschi Bers is passionate about using the power of technology to promote positive development and learning for young children. Check out her 2014 TEDx talk "Young programmers — think playgrounds, not playpens". Bers' philosophy and theoretical approach as well as the curriculum and assessment methods can be found in her books "Beyond Coding: How Children Learn Human Values through Programming" (MIT Press, 2022); "Teaching Computational Thinking and Coding to Young Children" (IGI, 2021); "Coding as Playground: Programming and Computational Thinking in the Early Childhood Classroom" (Routledge, 2018; 2020); "The Official ScratchJr Book" (2015; No Starch Press); "Designing Digital Experiences for Positive Youth Development: From Playpen to Playground" (2012, Oxford University Press); and "Blocks to Robots: Learning with Technology in the Early Childhood Classroom" (2008; Teacher's College Press).

Strawhacker, A., Relkin, E., Bers, M.U. (2022) Designing an Adaptive Assessment for Preschool Children's Robotics Knowledge. *Educational Designer*, 4(15). ISSN 1759-1325
Retrieved from: http://www.educationaldesigner.org/ed/volume4/issue15/article60/