2024 LEAP Challenge

LEAP

Project Host:

ARUKAY

ARUKAY

Vicky Ricaurte

Vicky Ricaurte is the CEO and co-founder of Arukay, an edtech company focused on teaching coding and computational thinking to K-12 students across Latin America. With a background in Industrial Engineering, Marketing, and a Master's in Management from Harvard, Vicky has over 15 years of experience in technology and education. Under her leadership, Arukay has impacted over 100,000 students across several countries. Vicky's work has been recognized globally, including being named one of the Top 200 EdTech CEOs worldwide. Her mission is to bridge the digital divide by providing accessible, high-quality education.

Cristina Duarte

Cristina Duarte holds a Bachelor of Science in Industrial Engineering and a Bachelor’s degree in Psychology, combining a strong technical foundation with deep insight into human behavior. She currently serves as Head of Product & Customer Experience at an EdTech company, where she leads multidisciplinary teams in the development of innovative learning solutions for K-12 schools across Latin America. Cristina is passionate about educational transformation, academic research, behavioral economics, and the intersection between people and technology.

Fellows:

Laura Di Giunta, Research Fellow

Natalia Kucirkova, Team Lead, Research Fellow

Patricia Lockwood, Research Fellow

Jazib Zahir, Social Entrepreneur Fellow

TABLE OF CONTENTS

Executive Summary

Introduction

Arukay is an EdTech company dedicated to equipping students in Latin America with essential digital skills, computational thinking, and coding expertise to prepare them for the demands of a technology-driven world. By integrating digital literacy into K-12 education, Arukay aims to break cycles of poverty and create opportunities for future generations. For over a decade, Arukay, as a growing organization has impacted over 100,000 students across multiple Latin American countries, Arukay seeks to strengthen the evidence base of its educational model through a LEAP Project with MIT Solve.

Organisation's role & strength

Arukay operates as a for-profit EdTech company with a mission to transform education by making digital literacy accessible to students, regardless of socioeconomic background. The organization is led by CEO and co-founder Vicky Ricaurte, whose expertise in management and educational technology positions the company as a leader in the field. The team includes specialists in curriculum development, platform engineering, finance, and commercial outreach, ensuring cross-functional collaboration and strong implementation capabilities. Arukay's success is driven by its structured and dynamic instructional design, robust teacher training programs, and advanced data analytics for learning assessment.

Need summary

Latin America faces a critical digital skills gap, with only 17% of students advancing to higher education and a slow intergenerational economic progression. The lack of systematic, early digital education exacerbates social inequalities and limits economic mobility. Arukay seeks to address this challenge by integrating coding and computational thinking into primary and secondary education. However, to maximize impact, the organization requires a stronger evidence base to validate its effectiveness, refine assessment methodologies, and develop scalable measurement frameworks.

Solution summary & next steps

- **High-Quality Curriculum**: Age-appropriate, multilingual digital literacy courses.
- **Teacher Training & Support**: Equipping educators with the necessary tools to integrate computational thinking into their classrooms.
- **Advanced Reporting & Analytics**: Providing real-time insights into student progress.

The next steps include leveraging the LEAP Project to:

1. Improve the design of formative and summative assessments.
2. Enhance data collection methodologies for measuring soft and hard skill acquisition.
3. Develop a long-term framework to track computational thinking skills across primary and elementary education.

Deliverable 1

Validated Measurement Framework: Arukay aims to refine its assessment methods to ensure accurate measurement of student progress and digital skill acquisition. This includes:

- Establishing a structured methodology for measuring learning outcomes.
- Aligning assessment tools with international standards such as ISTE and CSTA.
- Ensuring that measurement strategies provide meaningful insights for educators and stakeholders.

Deliverable 2

Improved Data Collection & Analysis Methodologies: The LEAP Project will support the development of a research-backed approach to data gathering and analysis, leading to:

- Enhanced data accuracy and reliability through systematic collection techniques.
- More effective teacher engagement in data-driven instructional decisions.
- A scalable model for tracking computational thinking skill development over time.

Deliverable 3

Redesign of a Revised Computational Thinking Measurement Tool: Arukay will redesign a revised version of a vehicle to measure the acquisition of computational thinking skills across primary and elementary school years. This includes:

- Defining key performance indicators to track long-term student progress.

- Developing an adaptive measurement system to ensure accurate skill assessment over time.
- Creating a structured framework to integrate the tool within existing educational programs.

Deliverable 4

Linking Arukay's Mission with Global Partners: The LEAP team will identify partnerships that make sense for Arukay as it looks to grow its global footprint:

- Sharing the computational thinking framework followed by the Raspberry Pi Foundation as a reference
- Identifying partnership models that may align Arukay with the Raspberry Pi Foundation
- Reviewing academic literature on the significance of computational thinking in preparing students for robotics competitions as part of making Arukay think about its significance to schooling

Deliverable 1: Literature review

As a first step, we identified, in collaboration with Arukay, three key outstanding questions in the field of computational thinking. Fellows then reviewed the literature to provide an evidence base to answer these outstanding questions before evaluating the assessment tools currently used.

1. What are the different frameworks of computational thinking?

Computational thinking

Computational thinking is a structured and systematic approach to problem-solving that enables individuals to break down complex problems, recognize patterns, focus on relevant information, and develop algorithmic solutions. It is a fundamental skill that extends beyond computer science, integrating into various academic disciplines and real-world applications (Digital Promise, n.d.). Before a problem can be effectively addressed, it must first be thoroughly understood, including the potential methods by which it can be resolved. Computational thinking provides a framework to achieve this understanding and formulate solutions that can be executed by humans, computers, or both.

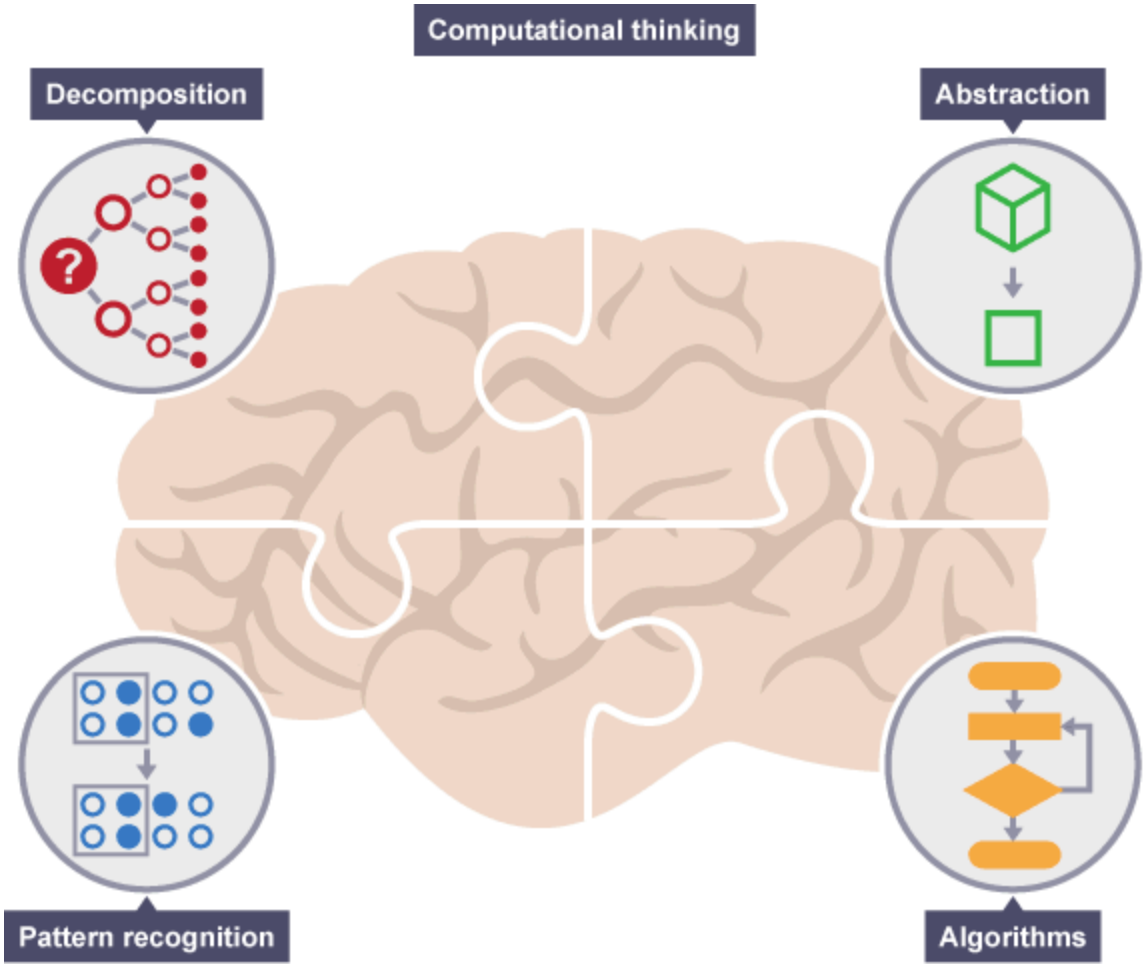


Figure 1: Image from BBC Bitesize, What is computational thinking? Available from: https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1

The Four Cornerstones of Computational Thinking

Computational thinking is underpinned by four essential techniques, often referred to as its cornerstones: decomposition, pattern recognition, abstraction, and algorithms. Each of these components plays a vital role in structuring and solving problems efficiently.

- **Decomposition** involves breaking down a complex problem or system into smaller, more manageable parts. This process makes intricate challenges more approachable by dividing them into simpler sub-problems, each of which can be analyzed and resolved individually before integrating the solutions into a cohesive whole.
- **Pattern Recognition** is the practice of identifying similarities and trends within problems. By recognizing recurring themes or structures, individuals can apply previous solutions to new problems, thereby increasing efficiency and reducing redundant effort.

- **Abstraction** focuses on isolating the most critical information while disregarding irrelevant details. This technique ensures that problem solvers concentrate on essential elements, making the process of finding a solution more streamlined and effective.
- **Algorithms** involve formulating a step-by-step procedure or set of rules to solve a problem. These procedures can be expressed in a manner that is comprehensible to both humans and computers, ensuring clear and repeatable execution.

Each of these techniques is equally important, functioning as interconnected skills that collectively enhance the ability to solve problems systematically. The absence of any one component weakens the overall problem-solving process, much like a table missing a leg. When applied correctly, these techniques are particularly beneficial in programming, as they help structure logical and efficient code.

The Broader Scope of Computational Thinking

Computational thinking extends beyond the realm of computer science. It is an interrelated set of skills and practices that facilitate the resolution of complex problems across various disciplines, including mathematics, science, social studies, and the arts. While computing encompasses both computer science and computational thinking, the latter serves as a cross-disciplinary approach to problem-solving. Unlike programming, which involves writing and debugging code to be executed by a computer, computational thinking focuses on the conceptual strategies that underpin effective problem-solving in diverse contexts.

For educators seeking to integrate computational thinking into their classrooms, it is best understood as a continuum of interrelated skills and competencies rather than a singular, isolated ability. By fostering these skills in students, educators can prepare them not only for careers in technology but also for a world increasingly driven by computational processes. Computational thinking is, therefore, not just a technical skill but a fundamental cognitive approach that enhances critical thinking and problem-solving capabilities across multiple domains.

The following rapid evidence review examines key CT frameworks, highlighting their key components and key contributions to the field.

Early Foundations: Seymour Papert's Model (1980)

Seymour Papert was a pioneer in computational thinking, introducing the concept through the LOGO programming language. Papert's work emphasized the role of computers in fostering exploratory learning, particularly in mathematics and artistic expression (Papert, 1980). By engaging children in programming, Papert aimed to cultivate higher-order thinking skills, reinforcing logical reasoning and problem-solving abilities. His model laid the foundation for integrating coding as an essential pedagogical tool, influencing subsequent CT frameworks.

Jeannette Wing's Computational Thinking Framework (2006)

Jeannette Wing (2006) significantly advanced the discourse on CT by formalizing it as a universal skill applicable beyond computer science. Wing conceptualized CT as involving problem-solving, system design, and an understanding of human behavior. Her seminal work argued that CT should be a fundamental competency akin to literacy and numeracy, prompting a global movement to integrate CT into K-12 education (Wing, 2006). This framework underscored the interdisciplinary relevance of CT, advocating for its inclusion in diverse curricula.

Brennan and Resnick's Three-Dimensional Framework (2012)

Brennan and Resnick (2012) expanded upon existing CT models by introducing a three-dimensional framework encompassing concepts, practices, and perspectives. The concepts dimension includes fundamental programming constructs such as loops, conditionals, and sequences.They are the concepts that users need to master to understand the mechanics of programming, including sequences, loops, parallelism, events, conditionals, operators, and data. The practice dimension includes strategies that learners apply while solving problems, including during programming, expressing, connecting, and questioning concepts. The dimension draws on four main strategies: being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing. The perspectives dimension accounts for the socio-emotional aspects of CT, such as collaboration and self-awareness.

This model provides a structured methodology for assessing and developing CT skills in educational contexts.

Tikva and Tambouris's Five-Area Model (2021)

Tikva and Tambouris (2021) introduced a comprehensive framework for CT that categorizes its components into five interconnected areas: knowledge base, learning strategies, tools, assessment, and capacity building. This model offers a structured approach to integrating CT into K-12 education, emphasizing both theoretical knowledge and practical application. By incorporating assessment methods, the framework ensures that CT skills are systematically developed and evaluated.

Zapata-Cáceres's Beginners Computational Thinking Test (2020)

Zapata-Cáceres et al. (2020) designed the Beginners Computational Thinking Test (BCTt) to assess students' proficiency in CT concepts across different educational stages. The test evaluates fundamental computational constructs, including sequences, simple loops, nested

loops, and conditionals. While the BCTt provides a structured approach to measuring CT skills, it does not integrate socio-emotional components, leaving room for future research on the interplay between computational skills and broader cognitive abilities.

In sum, from Papert's pioneering work in exploratory programming to Wing's conceptualization of CT as a universal literacy, each model provides unique insights into the development and assessment of computational skills. Future research should focus on integrating socio-emotional aspects with core computational skills to create a holistic CT framework that supports both cognitive and collaborative learning.

2. Is any framework of computational thinking more reliable/evidence-based than any other framework?

As reviewed in the previous section, many different definitions of computational thinking (CT) exist. Some definitions have been inspired by the culture and the technicalities of professional computer science, whereas others have been developed by educators and researchers. These issues surrounding definition have made it hard to assess whether there is a suitable evidence base supporting the teaching of CT to students. Similarly, the ongoing discussion and confusion surrounding the definition of CT have led to varied interpretations and operationalization in implementing CT in teachers' professional learning. If we converge on a **general definition as the ability to analyze and solve various problems based on cognitive competencies and dispositions** (Wing, 2010) we can next ask whether there is an evidence base that supports specific components being essential to these factors. However, it is too early to answer whether one framework has a stronger evidence base than any other framework.

The evidence base is rapidly expanding given the sharp increase in the impact of information technologies in our lives. However, rigorous research studies are still in their infancy. One of the best types of evidence comes in the form of systematic reviews that can take a 'gold standard' empirical approach to surveying the literature and drawing conclusions. Liu et al., (2024) conducted a systematic review on supporting teachers in integrating computational thinking into K-12 classrooms. They surveyed the literature with key terms including the topic word "computational thinking", a set of synonyms for professional development (i.e., "professional development", "teacher development", "training", "intervention", and "workshop") and a set of synonyms for the potential outcomes of professional development (i.e., "teacher thinking", "teacher perception", and "teacher knowledge"). They concluded that future research should investigate what constitutes "good CT instruction" and how it can be effectively measured. This includes developing robust assessment tools that can capture teachers' professional growth longitudinally and its impact on student performance.

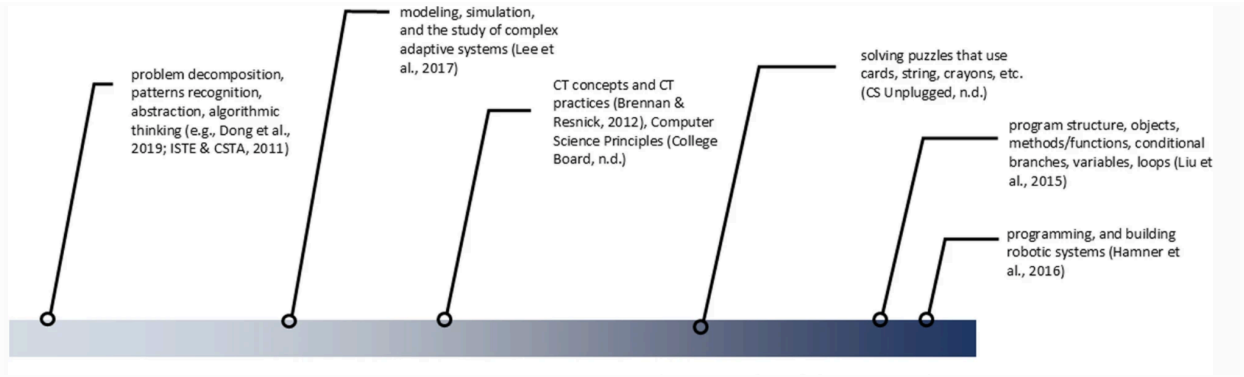


Figure 2. *Schematic showing operationalisation of computational thinking in light of how it impacts professional development.*

Those with a stronger emphasis on computing tool usage on the right side of the spectrum while the operationalizations with a heavier emphasis on thinking abilities on the left side of the spectrum. None are more valuable than another, but they are useful for contextualising the complex landscape of computational thinking for professional development for teachers. From Liu et al., (2024).

Evidence base for the benefits of computational thinking for students

Despite widespread agreement that teaching CT is beneficial for students, there remain very few empirical studies with rigorous research methods supporting a benefit. The National Research Council (United States) concluded that teaching programming improves interpersonal, self-regulatory, and metacognitive thinking skills (NRC, 2010). However, another study found a negative relationship between cooperativity and academic performance, and no association with algorithmic thinking, critical thinking, creativity, and problem-solving (Doleck et al., 2017). There is some evidence that teaching CT improves performance on a computer science course (Gouws et al., 2013). As mentioned, one of the best types of evidence at the early stage of developing an evidence base are systematic reviews. A systematic review of K-12 CT research found that teaching coding influenced a range of educational outcomes, including problem-solving, critical thinking, social skills, self-regulation, and other academic skills, such as reading and spelling (Popat & Starkey, 2019).

Evidence base for assessing computational thinking and how to teach CT

Another important question is whether there is an evidence base for how to assess computational thinking and how to teach it, and whether particular methods are more reliable than others. Vihavainen et al. (2014) found that teaching interventions focused on CT can improve programming pass rates by nearly one-third when compared to a traditional lecture and lab-based approach. These included peer-led team learning, pair-programming, peer-teaching as well as designing relatable content adapted to student interests and needs, including effective media design, real-world projects, and gamification.

Self-report measures, where students are asked questions about how much they have developed CT skills, have also begun to be developed. The Computational Thinking Scale is a 29-item instrument composed of five factors, namely creativity, cooperativity, algorithmic thinking, critical thinking, and problem-solving (Korkmaz et al., 2017). They suggested that the psychometric properties of the scale were good, and the measures have been cited over 700 times. However, this scale was developed to assess CT skills in undergraduate degree students and is likely not suitable for younger students. Another scale, also called The Computational Thinking Scale (Tsai et al., 2020), but developed for young people ages 13 to 15, focused on 5 core CT components (abstraction, decomposition, algorithmic thinking, evaluation and generalization). They found good validity and reliability for the assessment. The items are shown below.

Abstraction1	I usually think of a problem from a whole point of view, rather than looking at the details.
Abstraction2	I usually think about the relations between different problems.
Abstraction3	I usually try to find the key points of a problem.
Abstraction4	I usually try to analyze the common patterns of different problems.
Decomposition1	I usually think if it is possible to decompose a problem.
Decomposition2	I usually think of the structure of a problem.
Decomposition3	I usually think about how to split a big problem into several small ones.
Algorithmic Thinking1	I am used to figuring out the procedures step-by-step for a solution.
Algorithmic Thinking2	I usually try to find effective solutions for a problem.
Algorithmic Thinking3	I usually try to lay out the steps of a solution.
Algorithmic Thinking4	I usually try to figure out how to execute a solution for a problem.
Evaluation1	I tend to find a correct solution for a problem.
Evaluation2	I usually think of the best solution for a program.
Evaluation3	I usually try to find the most effective solution for a problem.
Evaluation4	I usually think of the fast solution for a problem.
Generalization1	I tend to solve a new problem according to my experience.
Generalization2	I usually try to use a common way to solve different problems.
Generalization3	I usually think about how to apply a solution to other problems.
Generalization4	I usually try to apply a familiar solution for solving more problems.

Figure 3. The computational thinking scale. A validated measure for young people aged 13-15 (Tsai et al., 2020).

Another measure with programming items, The computational thinking assessment scale (Shen et al., 2024), is used to assess children ages 8-11. It consists of 10 items, 9 multiple-choice questions and 1 open-ended question. The questions focus on students' ability to think algorithmically and apply computational processes in different contexts. In a study of 222 children, they found the CTAS to be a strong tool for assessing computational thinking. In a systematic review of measures assessing computational thinking (Ocampo et al., 2024) concluded that:

- Validity is the most reported psychometric property in CT measurement
- Abstraction is the most evaluated skill in computational thinking measurement instruments.
- In Latin America, there is an absence of instruments for measuring CT
- Türkiye, USA, and China lead the publication of articles that evaluate CT

			Validity					Reliability
Year	Article	Instrument Used and/or Adjusted	Content	Construct	Criterion	Other	No evidence	Coefficient
2023	A Normative Analysis of the TechCheck Computational Thinking Assessment	TechCheck					X	Cronbach's alpha
2023	Computational Literacy: Unplugged musical activities around Bebras International Challenge	Thinking Test using Bebras Problems					X	Cronbach's alpha
2023	Computational thinking in primary school: effects of student and school characteristics	TechCheck					X	Cronbach's alpha
2023	Developing and Testing a Design-Based Learning Approach to Enhance Elementary Students' Self-Perceived Computational Thinking	CTS		X				Cronbach's alpha
2023	Development and validation of a computational thinking test for lower primary school students	CTtLP	X	X	X			Cronbach's alpha
2023	Effect of Reverse Engineering Pedagogy on Primary School Students' Computational Thinking Skills in STEM Learning Activities	CTS		X				Cronbach's alpha
2023	Effects of robotics STEM camps on rural elementary students' self-efficacy and computational thinking	Survey adapted from CTS					X	Cronbach's alpha
2023	Effects of Scratch-Based Activities on 4th-Grade Students' Computational Thinking Skills	BCTt	X		X			Cronbach's alpha
2023	Exploring the underlying cognitive process of computational thinking in primary education	CTtLP					X	Cronbach's alpha
2023	Monitoring cognitive development through the assessment of computational thinking practices: A longitudinal intervention on primary school students	CT practices test	X	X				Cronbach's alpha

2023	Possibilities of diagnosing the level of development of students' computational thinking and the influence of alternative methods of teaching mathematics on their results	Didactic CT test		X	X		No evidence
2023	The effect of an unplugged coding course on primary school students' improvement in their computational thinking skills	CTST	X				KR-20
2023	Think together, design together, code together: the effect of augmented reality activity designed by children on the computational thinking skills	TechCheck				X	No evidence
2023	Unravelling the underlying mechanism of computational thinking: The mediating role of attitudinal beliefs between personality and learning performance	CTtLP		X		X	Cronbach's alpha
2024	A Bebras Computational Thinking (ABC-Thinking) program for primary school: Evaluation using the competent computational thinking test	cCTt		X			Cronbach's alpha
2024	The effect on computational thinking and identified learning aspects: Comparing unplugged smartGames with SRA-Programming with tangible or On-screen output	CTt				X	Cronbach's alpha

Figure 4. Overview of computational thinking assessment tools developed since 2023. Adapted from Ocampo et al., (2024).

In summary, when assessing an evidence base, there are several important factors to consider. Is there evidence for how best to inform teachers of CT for their professional development? Second, does teaching CT improve students' academic performance and abilities and does this generalize to other areas of learning? Finally, how can we best assess CT empirically to improve the evidence base of the first two questions?

3. Does Framework Variability Relate to Different Countries, Different Scopes or Different Age Groups?

Variability in Computational Thinking Frameworks Across Countries and Cultures

Computational Thinking (CT) frameworks are influenced by many factors, such as the educational goals of each country, cultural values, and available resources. As CT is integrated into school curriculums in different countries, it is important to understand how age-related development affects how these frameworks are taught. Key factors include cognitive development (how children think and learn) and socio-emotional development (how they manage emotions and interact with others). These factors change as students grow older.

For example, in countries like the United States, large initiatives like Computer Science for All focus on teaching coding and algorithms (Grover & Pea, 2013). However, in countries with less access to technology, CT might focus more on problem-solving and teamwork (Ocampo et al., 2024). This shows how frameworks must adjust depending on each country's culture, economy, and technology. Additionally, these frameworks should take into account the age-related development of students, including their emotional and social needs.

Differences in Scope of Frameworks Across Disciplines and Educational Contexts

The scope of CT frameworks differs across subjects, school levels, and even age groups. Children's ability to understand complex concepts changes as they get older. For example, younger children may need frameworks that focus more on basic skills, while older students can handle more advanced topics. As children develop cognitively, their ability to think more abstractly grows. For instance, when students are younger, they might focus on concrete tasks like solving puzzles, but by adolescence, they can work on more abstract concepts like designing computer programs, a pattern that aligns with Piaget's original theory (1952), although it is not universally accepted (Lourenco & Machado, 1996).

Additionally, emotional development, such as learning how to control one's emotions (called emotional regulation) and understanding how to work well with others (called prosocial behavior), also influences how they engage with learning. Emotional regulation refers to the ability to manage and control one's emotional responses, while prosocial behavior includes actions that benefit others, such as helping or sharing. As students grow, they become more able to manage emotions and engage in collaborative work, which can improve how they learn computational thinking (Eisenberg, Spinrad, Knafo-Noam, 2015).

Age Group Variability in the Application of CT Frameworks

Middle Childhood (Ages 6-11)

During middle childhood, children develop the ability to think logically about tangible objects

and relationships, but they may struggle with abstract reasoning (Piaget, 1952). At this age, CT frameworks often use hands-on tools like Scratch, a visual programming language. These tools allow children to manipulate images and objects on the screen to learn concepts like loops and sequences without needing to understand complex programming syntax (Resnick et al., 2009). This hands-on approach helps children break down problems into simpler parts, which is a key skill in computational thinking.

In terms of socio-emotional development, children at this age start to develop better social skills, such as working together with peers. Erikson (1963) calls this stage the development of competence through interactions with others. CT frameworks can take advantage of this by encouraging group-based activities. Group learning helps children develop skills such as communication, empathy, and teamwork. Emotional regulation also plays a role here, as children at this age start to learn how to manage their emotions in social situations, which can improve collaboration (Eisenberg et al., 2015).

Preadolescence (Ages 12-14)
In preadolescence, children begin to think more abstractly and deal with hypothetical situations (Piaget, 1952). As their cognitive skills advance, CT frameworks for this age group often involve more complex tasks, such as designing algorithms (step-by-step instructions for solving problems) or using programming languages like Python or JavaScript.

At the same time, preadolescents are exploring their identities and becoming more independent, which Erikson (1963) sees as crucial for developing self-confidence. During this stage, it is important for CT frameworks to offer opportunities for personal projects that allow preadolescents to explore topics they care about. Peer feedback is also very important in this stage, as adolescents tend to rely more on their friends and social groups for validation. Collaborative, peer-based learning can engage preadolescents more effectively (Doleck et al., 2017). This is where prosocial behavior—working well with others—is a key part of learning and collaboration.

Adolescence (Ages 15-18)
Adolescents are capable of advanced abstract thinking, which allows them to engage with more sophisticated computational topics, such as data structures, algorithms, and software engineering (Piaget, 1952). At this stage, CT frameworks focus on preparing students for careers by offering real-world applications like app development or data science projects. These projects challenge students to think critically and solve complex problems.

The emotional and social changes adolescents experience are important to consider in CT frameworks. As Erikson (1963) notes, adolescence is a time for exploring one's identity and deepening social relationships. Adolescents care about how they relate to their peers, and their ability to manage emotions is crucial for success in group projects. This is where emotional regulation becomes important in CT frameworks. Working on larger projects with others helps

students develop both technical skills and social-emotional skills like collaboration and conflict resolution (Grover & Pea, 2013).

Cultural and Educational Influences on Framework Design

The culture of a country plays a significant role in how CT frameworks are designed. In countries with collectivist cultures (like many East Asian countries), there is a focus on group work and cooperation. This fits well with the social-emotional development of children in these cultures, who may be more accustomed to working together to achieve shared goals (Doleck et al., 2017). On the other hand, in individualistic cultures (like many Western countries), there may be more focus on independent problem-solving and self-regulation.

The availability of technology also affects how CT is taught. In countries with limited access to technology, the focus may be more on cognitive skills, such as creative problem-solving, rather than on coding and programming. This ensures that all students, regardless of access to technology, can still benefit from learning the core ideas of computational thinking.

Studies from Nordic Countries

In countries like Finland, Sweden, and Denmark, CT education is integrated into various subjects, not just computer science. These countries emphasize creativity, problem-solving, and collaboration, which aligns with the developmental stages of students. For example, Finland's curriculum incorporates computational thinking into subjects like math, science, and the arts, showing how CT can apply to real-world challenges (Kikas et al., 2020). Additionally, Bocconi, Chioccariello, and Earp (2018) highlight the Nordic approach to integrating computational thinking and programming into compulsory education, where the focus is on fostering creativity and problem-solving skills, as well as ensuring that all students can access this type of learning regardless of their socio-economic background.

Studies from North America

In North American countries like the United States, CT is seen as essential for preparing students for a tech-driven future. Initiatives like Computer Science for All focus on providing coding and computational skills to a wide range of students (Grover & Pea, 2013). Similarly, Canada and other North American countries are heavily investing in STEM (Science, Technology, Engineering, and Mathematics) education to equip students with the cognitive and technical skills needed for future careers (Grover & Pea, 2013).

Studies from Latino Countries

In Latino countries, there is an increasing recognition of the importance of CT for socio-economic development. In Brazil, the government has launched initiatives to teach coding and problem-solving to students from elementary to high school, helping them compete in a global economy (Bocconi et al., 2018). Similar initiatives are underway in Mexico and

Argentina, where schools are focusing on improving digital literacy to enhance both cognitive and socio-emotional skills (Bocconi et al., 2018).

Summary for this section

The variability in Computational Thinking (CT) frameworks across different countries, educational levels, and age groups highlights the need for context-sensitive models that can be tailored to the local educational, cultural, and technological environment. While the core principles of CT, such as decomposition, abstraction, and algorithmic thinking, remain universal, their application and assessment must be adapted to meet the needs of students in diverse settings.

Frameworks in K-12 education are often designed to be flexible and accessible, focusing less on formal programming languages and more on the broader cognitive aspects of CT, such as problem decomposition and pattern recognition. These frameworks also tend to prioritize collaborative problem-solving and hands-on learning, as these approaches have been shown to enhance student engagement and facilitate deeper understanding (Grover & Pea, 2013). For instance, platforms like Scratch (Resnick et al., 2009) are commonly used in elementary and middle schools to introduce coding concepts through interactive and creative projects.

In contrast, higher education frameworks tend to be more specialized and focused on the technical development of specific computational skills. These frameworks are often built around formal programming languages, algorithm design, and data structures to prepare students for professional careers in computer science and related fields (Korkmaz et al., 2017). In university settings, CT frameworks are more structured, offering deep dives into technical topics, whereas K-12 frameworks focus more on providing students with a broad foundation in computational thinking that can be applied across disciplines.

Future research should explore how these frameworks can be adjusted to ensure that all learners, regardless of their background or access to technology, have the opportunity to develop the critical thinking and problem-solving skills needed in the 21st century.

Deliverable 2: Evaluation of the Arukay assessment tool and its psychometric properties

Process

Goal

The primary goal of this evaluation was to assess the content validity of the Arukay Assessment System by gathering external expert feedback. This system was reviewed by a team of independent evaluators (i.e., the fellows) to ensure that its items were clear, culturally relevant, and effective in assessing key skills. The evaluators provided feedback on specific aspects such as clarity, cultural appropriateness, and the ability to assess problem-solving skills.

Profile of the Independent Evaluators

The evaluation was carried out by three independent experts, each bringing unique expertise to the project, ensuring a diverse range of perspectives:

- **Evaluator 1**: Patricia Lockwood is a Professor of Decision Neuroscience and Wellcome Sir Henry Dale Research Fellow at the University of Birmingham, UK. Her research focuses on learning, decision-making and social cognition across the lifespan from childhood to old age. She regularly uses computational models of behaviour in her research work. Her skills helped her to ensure the psychometric properties of the Arukay Baseline Assessments, and in ensuring the language and structure was tailored to the abilities of different age groups.

- **Evaluator 2**: Natalia I. Kucirkova is a research professor affiliated with the University of Stavanger, Norway and The Open University and University College London, UK. Natalia's research takes place collaboratively across academia, commercial and third sectors. She co-founded and currently directs the International Centre for EdTech Impact that connects EdTech academia and industry. Natalia is widely published on topics of EdTech evidence in leading journals including Nature and NPJ of Learning. Natalia's expertise supported the

Arukay's Baseline Assessment in terms of its equity and inclusive design properties and alignment with the Science of Learning.

- **Evaluator 3**: Laura Di Giunta is a Professor of Personality Development Psychology at Sapienza University of Rome, Italy. Her specialization in children and adolescent socio-emotional development, alongside her expertise in quantitative psychology and factorial validity, allowed her to evaluate the content validity of the Arukay Assessment System. Her skills were helpful in ensuring the system aligned with developmental and socio-emotional concepts for children and adolescents.

It is important to note that no evaluators were native Spanish speakers, and the Arukay system was originally created in Spanish. Therefore, the evaluators reviewed the system's content based on its translated version. While their expertise ensured a focus on clarity and cultural relevance, they were not positioned to assess the system's overall effectiveness in its original language.

Evaluation Focus Areas

The evaluators provided feedback on the following key areas:

- **Clarity of Instructions**: How easy is it for children in different age groups to understand and follow the instructions for each task?
- **Cultural and Contextual Relevance**: How well do the items reflect the diverse cultural backgrounds of children, ensuring fairness and avoiding bias?
- **Decomposition**: How well do the items help children break down complex problems into smaller, manageable parts?
- **Abstraction**: How well do the items help children focus on relevant details and ignore irrelevant ones?
- **Patterns**: How well do the items encourage children to recognize patterns or trends?
- **Algorithm**: How well do the items help children understand and create step-by-step procedures to solve problems?

- **General Improvements**: Any suggestions for improving the wording or clarity of the items.

Types of Investigations

To achieve a comprehensive evaluation, two distinct types of investigations were conducted:

1. **Quantitative Data Analysis from Students**
   The fellows analyzed previously collected data from students of different ages who had completed the baseline assessment using the Arukay system. This data was analyzed quantitatively to provide insights into the system's performance in assessing key skills. While the fellows were not directly evaluating the validity of the system, their analysis contributed to the ongoing process of establishing the system's validity by offering valuable feedback on how well the system functioned in real-world conditions.
2. **Fellows' Rating of the Existing Measure**
   The fellows also rated the items based on various attributes, such as clarity, cultural relevance, and alignment with the problem-solving skills being assessed. This allowed the fellows to provide direct feedback on the system's content from their professional perspectives. The results from the fellows' ratings were then used to further inform the evaluation process, ensuring a comprehensive review of the system's strengths and potential areas for improvement.

Evaluator Scoring and Analysis

Each fellow scored the items on a scale of 1 (not much) to 3 (very much) based on how well the items met the evaluation criteria. Items were assessed for three distinct age groups: 3rd–5th graders, 6th–8th graders, and 9th–11th graders. The scores were then aggregated to provide insights into the content validity of the translated materials.

The results from both the evaluators' feedback and the quantitative data analysis offered a comprehensive understanding of how well the Arukay Assessment System aligns with its intended goals. Based on the findings, the fellows proposed recommendations for improving the system, helping to guide the next steps in validating its effectiveness.

Results

Quantitative feedback based on existing baseline assessment data

The fellows recorded the items provided by Arukay for the existing baseline assessments. Accuracy in responses for each of the items was calculated (0 or 1 depending on if the answer was correct or incorrect). After reviewing the composition of the questionnaire, it was determined that the clearest measure that could be psychometrically assessed was 'difficulty index (also known as the p-value)' which is a measure of how easy or difficult a question is, based on the percentage of students who could answer it correctly. This is a quantitative measure that can be calculated from existing baseline data provided by Arukay, and is different from the 3 independent ratings provided by raters on 'clarity' of the wording of the different items.

We provide the results of this difficulty index analysis below for the 3 baseline assessments we had access to, which contained responses for 3 questions each per domain of computational thinking (Decomposition, Patterns, Abstraction, Algorithmic thinking). We also provide a guideline for the range of values for each item to be classified as easy, average or difficult. Items should ideally be revised if they fall into the easy or difficult range (Bermundo, C., Bermundo, A. & Ballester, R., 2004; Sahoo, D. P., & Singh, R., 2017).

Difficulty Index		
Value	**Interpretation**	**Action**
<30%	Difficult	Revise or discard
30-75	Good or excellent	Keep
>75	Easy	Revise or discard

Difficulty index for baseline assessments

Please find below the results of the difficulty index analysis for each of the 3 age group baseline assessments.

3 to 5	Decomposition			Patterns			Abstraction			Algorithm		
Item	3	4	5	8	9	10	13	14	15	18	19	20
Difficulty index	93.9	57.1	75.5	71.4	57.1	77.6	98	85.7	73.5	59.2	79.6	91.8

6 to 8	Decomposition			Patterns			Abstraction			Algorithm		
Item	3	4	5	8	9	10	13	14	15	18	19	20
Difficulty index	83.5	88.2	92.9	88.2	77.6	85.9	75.3	78.8	70.6	85.9	72.9	80

9 to 11	Decomposition			Patterns			Abstraction			Algorithm		
Item	9	13	17	10	14	18	11	15	19	12	16	20
Difficulty index	91.1	96.2	82.3	86.1	88.6	57	92.4	98.7	68.4	86.1	88.6	97.5

Quantitative feedback based on Fellow ratings

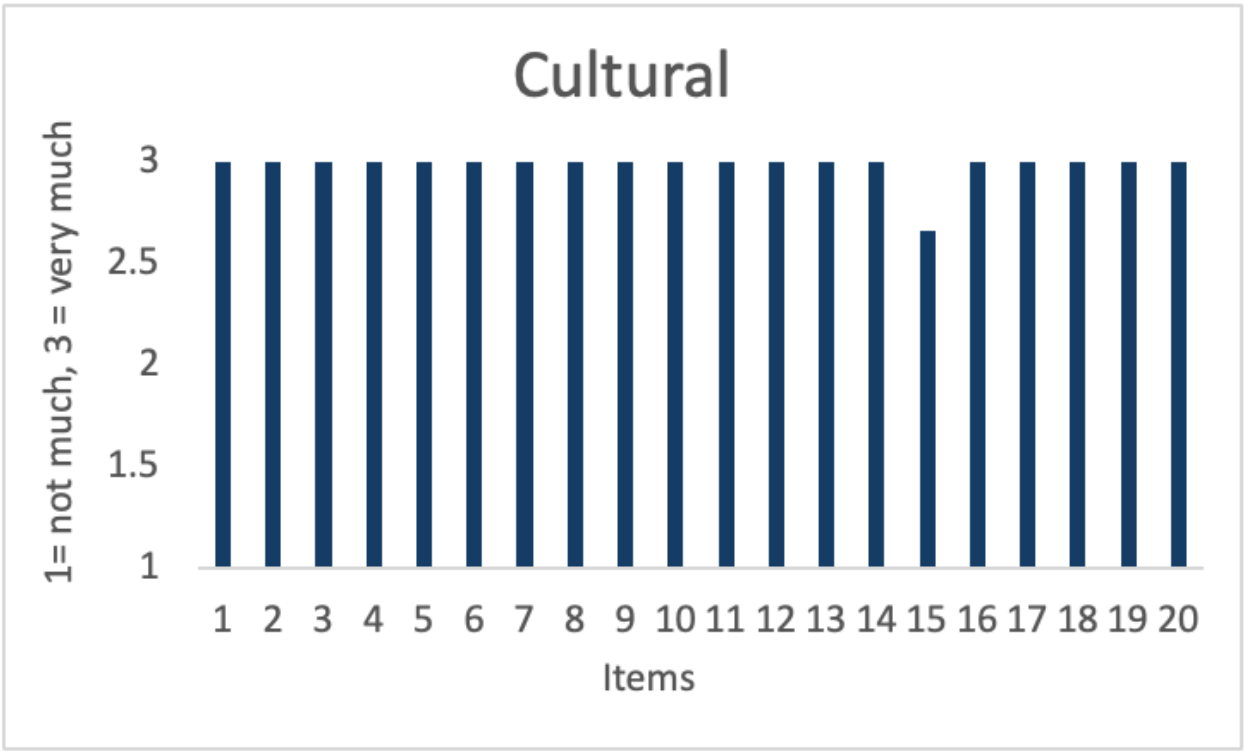
Please find below the statistical results from the fellow ratings assessment of the baseline measures. Summary statistics are provided as well as graphs depicting the average fellows scores for each of the 20 items in each baseline assessment.

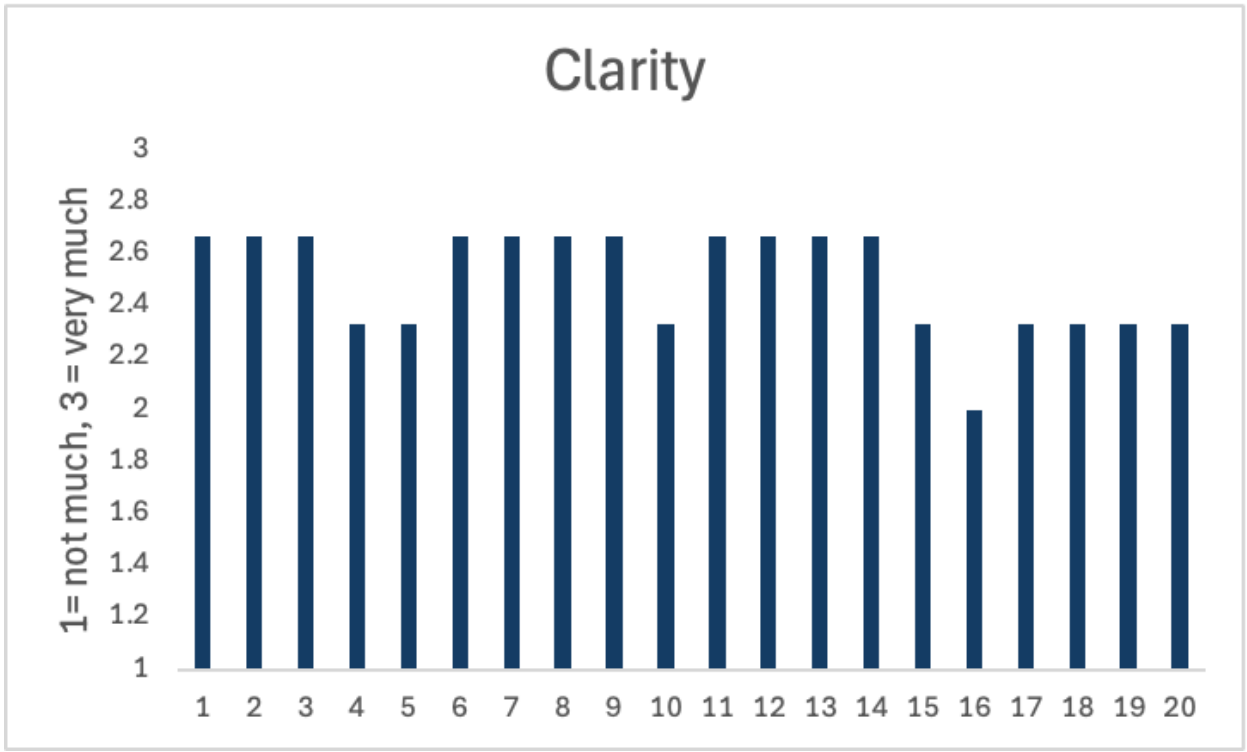
Combined ratings for different constructs assessed by the fellows.

Average score (scale 1 = not much, 3 = very much)	Clarity	Cultural	Decomposition	Abstraction	Patterns	Algorithms
Baseline 3 to 5	2.50	2.98	2.33	2.00	3.00	2.67
Baseline 6 to 8	2.35	2.60	2.67	2.33	2.00	2.00
Baseline 9 to 11	2.17	2.48	2.00	2.00	1.67	2.00

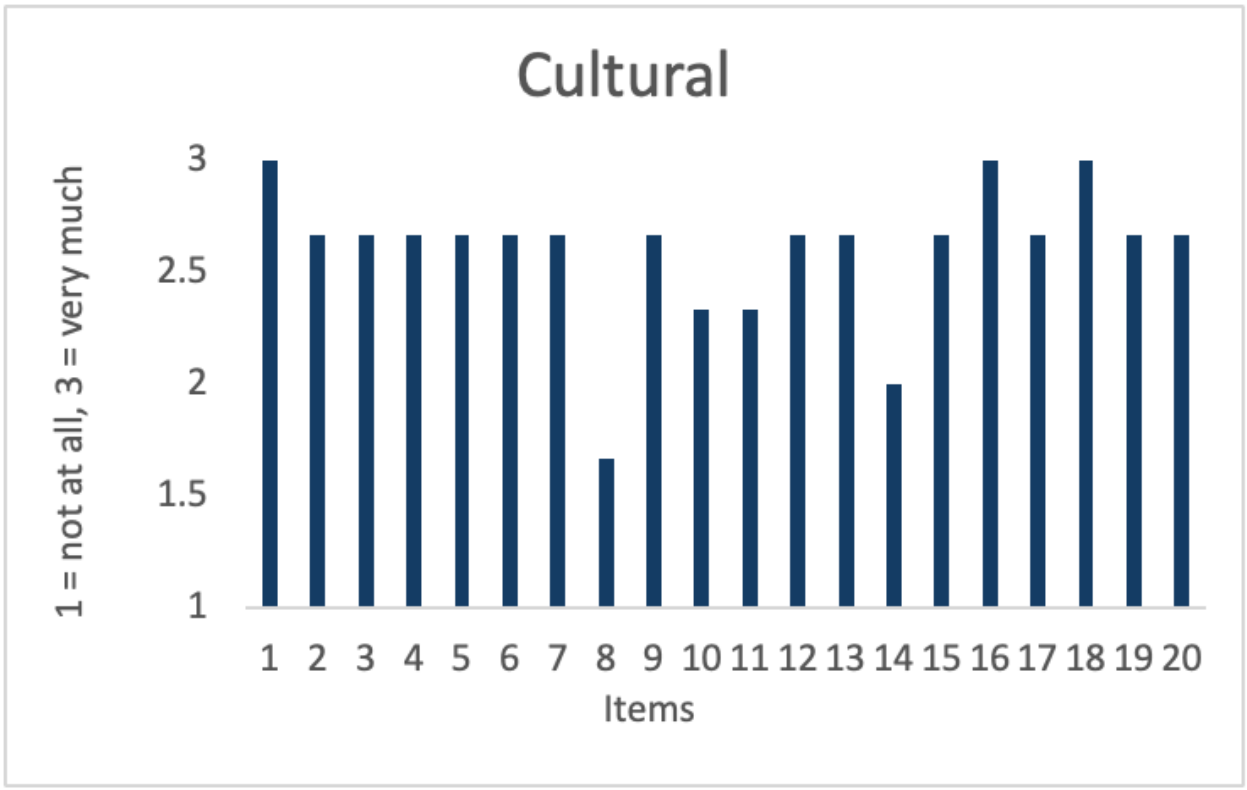
Combined ratings for individual items

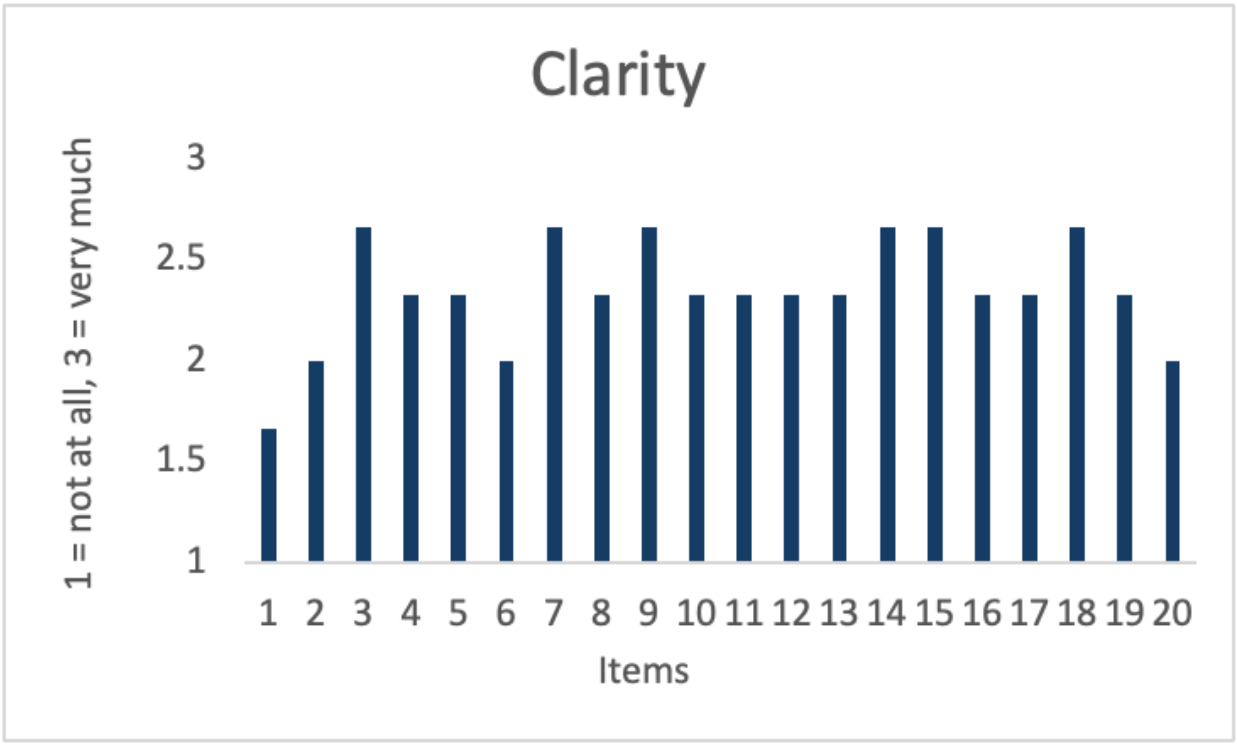
Baseline 3 to 5


Cultural
1= not much, 3 = very much
3
2.5
2
1.5
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Items

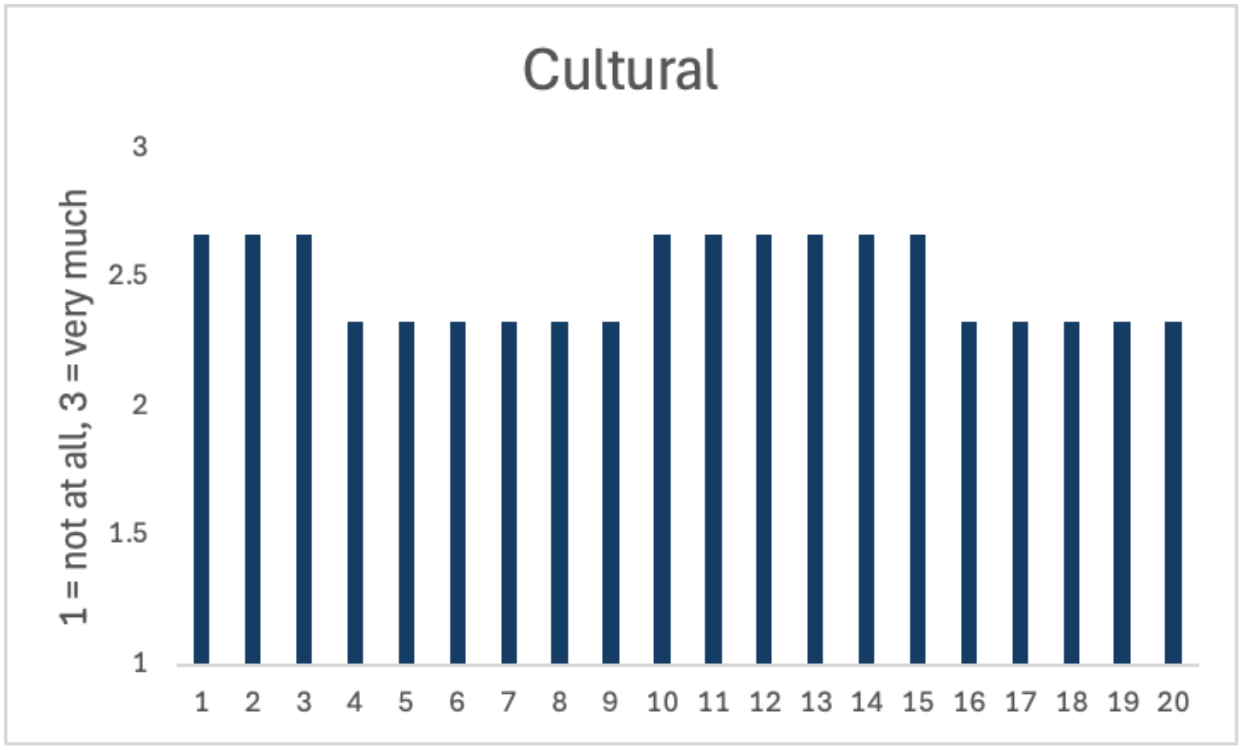

Clarity
1= not much, 3 = very much
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

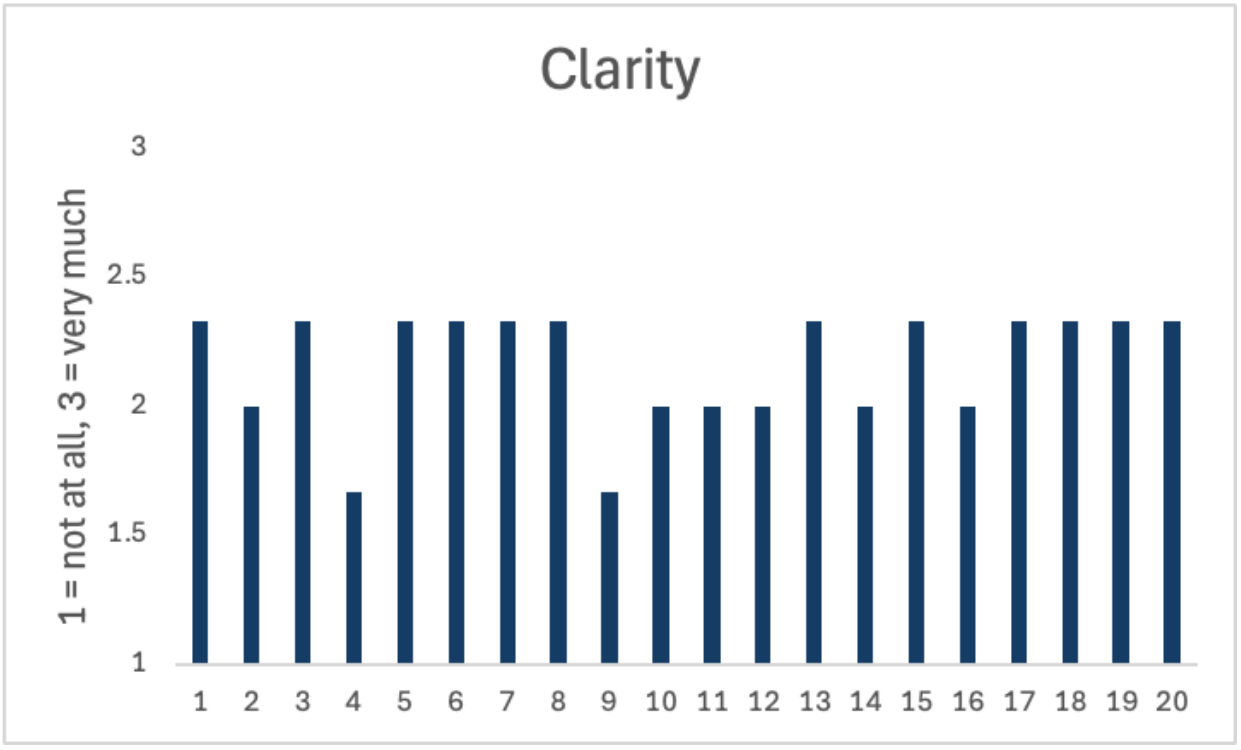
Baseline 6 to 8





Baseline 9 to 11





Recommendations to Arukay

General feedback from the Research Fellows

- To improve the quiz, consider breaking up multi-part questions into separate sentences to enhance clarity.
- Consistency in phrasing is also important—ensure that questions of the same type are worded exactly the same way to avoid confusion, such as standardizing "What figure comes next in the sequence?" rather than using variations like "Which figure continues in the sequence?"
- Keep question labels simple by just numbering them (e.g., "Question 1") instead of adding unnecessary descriptors like "Question 1: Problem/requirement," as straightforward instructions are most effective.
- To improve clarity and accessibility, avoid sub-questions and lengthy, multi-part questions. Instead, break them into simpler, more direct statements using full stops. For example, the question:

"*They ask you to make a work of art that contains only warm colors. Apart from that, the theme must be about any existing mammal. That is, you must draw a mammal with warm colors. You can use whatever art technique you prefer. Taking this requirement into account, how would you propose the breakdown of ordered steps to create the work of art?*"

Could be rewritten as:

"*Please draw a mammal using only warm colors. Select from the options below the steps you would need to complete the task.*"

This ensures the instructions are clear and easy to follow.

- Consider accessibility—does the use of colours accommodate colour-blind or visually impaired students? Universal Design principles suggest that colour should not be the sole means of conveying information. To make the quiz more inclusive, consider using high-contrast colour schemes that are distinguishable for individuals with colour blindness, such as blue and orange instead of red and green.
- For students with low vision, ensure that text is displayed in a clear, legible font with sufficient contrast against the background. Providing the option to enlarge text or switch to a high-contrast mode could further enhance accessibility. Moreover, screen-reader compatibility should be taken into account by structuring the quiz with alternative text for images and ensuring that interactive elements, such as drag-and-drop tasks, are navigable via keyboard controls.

- If possible, consider offering an auditory component where questions and answer choices can be read aloud. This would benefit not only visually impaired students but also those with reading difficulties or learning disabilities such as dyslexia. Ensuring accessibility from the outset will make the quiz more inclusive, allowing all students, regardless of their abilities, to engage with and benefit from the learning experience.
- True/false questions should always be formatted as clear questions with question marks. These question types may also encourage guessing rather than demonstrating knowledge, so consider using multiple-choice questions with three to four response options instead. Maintaining a consistent question format throughout the quiz will help students focus on the content rather than navigating different question structures.
- Overall, the images predominantly feature male figures, leading to an imbalance in the representation of female figures. To improve this measure, it would be beneficial to achieve a more balanced depiction of both genders in the visuals. Additionally, please refer to the notes in column C regarding certain items that may be relevant in some cultures (e.g., where a Christmas tree is recognized and holds significance), or during specific times (e.g., the Covid-19 pandemic), but not in others.

Specific feedback on Base 3-5 questions:

For Question 10, the answer is unclear—the instructions state that the number of lines must equal the number of spheres, yet this is not accurately reflected in the given diagram. It would be helpful to clarify this to avoid misunderstandings.

In Item #15, removing the 'Ñ' character might be beneficial, as it could introduce cultural specificity that is not relevant to the task's abstract nature. Lastly, the instructions for Item #16 are somewhat difficult to follow. To improve clarity, consider including an image demonstrating the completed task, similar to the original Tower of Hanoi, to provide students with a clear visual reference. These refinements will help ensure the quiz is more accessible, fair, and easy to understand for all students.

Specific feedback on Base 6-8 questions:

For items 1-7, the answers are all just pick the option that has the most steps, so they may not evaluate actual understanding too well.

Some of the items for patterns could be developed more from other assessment measures of patterns in this age group, as the items can be quite similar to the other categories and directly use the word 'patterns' rather than getting at pattern knowledge implicitly.

The way the questions map on to the four domains of CT is a bit clearer in baseline 3_5, could slightly more complex less text-based question be incorporated for this age group too?

Specific feedback for Base 9-11 questions:

Some of the items would be influenced quite strongly by cultural background, with examples talking about specific grades of 4.5 and 5, which are not universally applied across cultures.

After checking the Spanish and English versions, there might be a typo in item #15 (i.e., both items ask "what are the key points of these problems?"). Could this be an error from copying and pasting, or is it intentional?

Deliverable 3: Recommendations for Arukay to validate the revised the baseline assessment tool

Next steps for Arukay to validate the baseline assessment

Fellows recommend including a self-report measure of computational thinking for all age groups in the revised baseline assessment measure. This self-report measure will help evaluate students' perceptions of their own computational thinking abilities, and how these correlate with performance on the baseline assessment. The self-report scale can also be used before and after taking part in the Arukay programme to assess baseline computational thinking skills and how they change.

We recommend including The Computational Thinking Scale, shown below, which measures self-reported computational thinking skills in 5 domains (Tsai et al., 2020).

Abstraction1	I usually think of a problem from a whole point of view, rather than looking at the details.
Abstraction2	I usually think about the relations between different problems.
Abstraction3	I usually try to find the key points of a problem.
Abstraction4	I usually try to analyze the common patterns of different problems.
Decomposition1	I usually think if it is possible to decompose a problem.
Decomposition2	I usually think of the structure of a problem.
Decomposition3	I usually think about how to split a big problem into several small ones.
Algorithmic Thinking1	I am used to figuring out the procedures step-by-step for a solution.
Algorithmic Thinking2	I usually try to find effective solutions for a problem.
Algorithmic Thinking3	I usually try to lay out the steps of a solution.
Algorithmic Thinking4	I usually try to figure out how to execute a solution for a problem.
Evaluation1	I tend to find a correct solution for a problem.
Evaluation2	I usually think of the best solution for a program.
Evaluation3	I usually try to find the most effective solution for a problem.
Evaluation4	I usually think of the fast solution for a problem.
Generalization1	I tend to solve a new problem according to my experience.
Generalization2	I usually try to use a common way to solve different problems.
Generalization3	I usually think about how to apply a solution to other problems.
Generalization4	I usually try to apply a familiar solution for solving more problems.

Once the measure is revised according to the recommendations, fellows then recommend testing the revised measure in a new sample of students in the different age groups. We summarise the three stages of baseline assessment development to guide Arukay with how to implement these revisions and evaluate their effect on measurement validity.

Recommendations for Validating the Arukay Assessment Tool

Following the revision of the Arukay assessment tool, the next phase should be to rigorously validate the measure to ensure it reliably assesses the intended skills across age groups. This process involves gathering different types of validity evidence and conducting pilot testing. The steps outlined below are aligned with best practices in educational and psychological measurement (AERA, APA, & NCME, 2014) and should be implemented by a team with expertise in quantitative research methods and psychometrics.

Evidence of measurement validity

Construct Validity: To help assess whether the Arukay assessment measures the intended underlying competencies (e.g., algorithmic thinking), it is helpful to examine construct validity. This form of validity refers to the degree to which test scores reflect the theoretical construct(s) they are intended to measure (Brown, 2015).

A common method for assessing construct validity is factor analysis. Arukay's tool, which comprises 20 binary (correct/incorrect) items across four assumed factors per age group, could be assessed with two types of factor analysis:

- *Exploratory Factor Analysis (EFA)* is suitable when the underlying structure is uncertain.
- *Confirmatory Factor Analysis (CFA)* should be used when testing a predefined model, such as Arukay's 4-factor framework.

Since the items have a dichotomous (i.e., binary) outcome (students can either be correct or incorrect), standard factor analysis (which assumes continuous data) is inappropriate. Instead, you could use *tetrachoric correlation matrices* as input, which estimate the relationships between latent continuous traits underlying the binary responses (Brown, 2015; Holgado-Tello et al., 2010; Kilic, Uysal, & Atar, 2020). To have enough variables to estimate the factor structure accurately, collecting at least five items in each factor (e.g. decomposition) rather than the three currently used items is recommended.

If Arukay also integrates the suggested 20-item self-report measure (The computational thinking scale, Tsai et al., 2020), then factor analysis of this self-report scale could also be performed, and the individual items in Arukay's baseline assessment would be correlated with it. These analyses can be performed in many different statistical software packages, including R (free), JASP (free), or Mplus (paid).

Criterion Validity: To establish criterion-related validity, it is suggested to correlate the Arukay test scores with those from another validated assessment of similar constructs (e.g., digital

literacy, logic, or computational thinking tests). A moderate to high correlation indicates the tool measures what it claims to measure (Crocker & Algina, 2006).

Content Validity: This can be used to check that test items reflect the target competencies. This should involve expert reviews by subject matter specialists (e.g., educators, curriculum designers), who evaluate whether each item aligns with the learning outcomes and skills Arukay aims to assess (Haynes, Richard, & Kubany, 1995).

Predictive Validity: Assess how well test scores forecast future performance in academic or applied contexts—such as performance in coding competitions or standardized STEM exams. A method to do this relies on regression analysis (Anastasi & Urbina, 1997; Wong, 2020).

Item analysis: The difficulty index measures how easy or difficult a question is. A higher difficulty index means the question was easier, while a lower index indicates a more difficult question. This measure can be easily calculated by dividing the number of students who answered a question correctly by the total number of students who took the test. The result is expressed as a percentage. Guideline cut-offs can be viewed in this report's 'Results' section.

Reliability: Cronbach's alpha or the Kuder-Richardson Formula 20 (KR-20) could be used to assess the reliability of the individual items that evaluate each construct (e.g. decomposition). An alpha of 0.7 or higher is generally considered acceptable, reflecting good internal consistency.

Pilot Testing - once the tool is revised, according to our recommendations

Representative Administration: The revised tool should be pilot-tested with a sample reflecting the diversity of the target population across the three age groups. Ensuring demographic and skill-level variability will help assess whether the tool functions equally well across developmental stages.

Minimum Sample Size: Arukay's assessment assumes a model of 4 factors measured by 20 binary items, analyzed separately by age group. Based on guidelines for factor analysis with categorical data (MacCallum et al., 1999; Wolf et al., 2013), a minimum of 150–200 participants per age group is recommended. Power analysis for CFA with binary items can be conducted using software like Mplus or R (Kim, Winkler, & Talley, 2021).

Test-Retest Reliability: To evaluate the stability of the assessment over time, it is useful to check the revised tools test-retest reliability study. For this, it is recommended that the test be re-administered to the same group of students after 1–2 months and the correlation between the two sets of scores be computed. A high correlation (e.g., $r \geq .70$) suggests the tool yields consistent results across time points (Nunnally & Bernstein, 1994).

Anticipating Challenges and Iterative Adjustments

During the validation process, it could be important to anticipate and respond to challenges that may arise. For example, initial pilot data may reveal that certain items do not load clearly onto the expected factors or display limited variability (e.g., items that are too easy or too difficult), prompting item revision or replacement. Similarly, reliability indices for some constructs may fall below acceptable thresholds, requiring refinement of item wording or the addition of more items per factor. Differences in how the assessment performs across age groups could also indicate the need for age-specific adaptations or alternative forms. Moreover, recruitment for pilot testing may yield uneven sample sizes across groups, which may necessitate extending data collection or applying statistical techniques for handling unequal group sizes. Throughout the process, Arukay should adopt an iterative approach—analyzing results, revising the tool, and retesting where necessary—to ensure the final assessment is both psychometrically sound and developmentally appropriate across contexts.

Final Note

These steps will support Arukay in developing a scientifically rigorous, valid, and reliable assessment tool. We recommend partnering with experienced psychometricians or academic institutions to guide the validation process and ensure appropriate statistical modeling, especially given the binary nature of the items and the assumption of a four-factor structure.

The following table summarizes the proposed steps for validating the Arukay assessment tool, outlining key actions, purposes, and the types of expertise needed for implementation. The timeline for each step varies and will depend on Arukay's overall research stage and available expertise.

Step	Action	Details	Suggested Expertise
Construct Validity	Perform EFA or CFA using tetrachoric correlations	Confirms the assessment reflects the intended 4-factor structure; appropriate for binary items	Psychometrician or Quantitative Researcher
Criterion Validity	Correlate scores with external measure	Demonstrates alignment with another validated assessment of similar constructs	Psychometrician or Social Science Researcher
Content Validity	Expert review of items against curriculum goals	Ensures items comprehensively reflect target competencies; may involve panels or structured rubrics	Curriculum Specialist or Educator
Predictive Validity	Link current scores with future performance	Use regression or classification models to assess how scores forecast academic outcomes	Data Analyst or Educational Researcher
Item analysis	Check that the overall accuracy of each item is in a good range to pick up differences	Calculate the percentage correct for each item in the measure	Data analyst

Reliability	Check that the items within each factor are reliable measures of the factor	Use Cronbach's alpha or binary measure alternatives	Data analyst
Pilot Testing	Administer revised assessment to 150–200 students per age group	Ensures robust data for analysis; sample should reflect diversity across age groups	Data Analyst or Field Coordinator
Test-Retest Reliability	Re-administer test after ~2 months	Evaluates score stability over time by correlating initial and follow-up results	Psychometrician or Data Analyst

Deliverable 4: Linking Arukay's Mission with Global Partners

Arukay's work links up with the bigger picture in education through understanding school needs and preparation for various competitions of significance. Our goal is to help Arukay think through how it should pitch its value to students, parents and teachers and how it can link up with larger global partners for funding and support.

We started with Raspberry Pi Foundation given its prominence in the spaces of computational thinking and education. Raspberry Pi Foundation has developed a formal framework for computational thinking skills. The official document can be reviewed here as a reference:

The thinking behind the framework seems to be that it should be detailed enough to allow educators to build the key ideas into learning activities and resources and even use them to assess students.

These perspectives on computational thinking include experiences in the classroom and hence provide a different sort of perspective.

They define CT as a set of ideas and thinking skills that people can apply to design solutions or systems that a computer or computational agent can enact.

Underpinning all aspects of computational thinking is the logical analysis of problems and solutions. In that sense this is consistent with our earlier definitions which span both the uses of computers in problem-solving and general problem-solving techniques applicable to a wider range of scenarios.

They think of CT in terms of 6 distinct components:

- Decomposition in terms of identifying when a problem needs to be broken down, when instructions need to be broken down and when a problem can be broken down into simpler versions of itself. This includes an appreciation for how information flows between components, sensors and output devices. In that sense their interpretation of decomposition is less abstract than what we shared earlier in the report.
- Algorithm design in terms of sequencing instructions, grouping instructions, picking between instructions, the use of arithmetic and logical operators for instructions, naming collections of instructions and thinking of data in terms of variables and assignments. This includes identifying a range of test data and controlling the flow of

data through an algorithm. Again this definition stresses the importance of input and output and data flow and provides a different template upon which you could build assessment questions.

- Patterns and generalisation in terms of recognizing where multiple solutions are possible, predicting outcomes drawing on prior knowledge and transferring ideas from one problem to another. This idea of knowledge transfer which is widely discussed in literature about pedagogy and education is thus important to their definition of computational thinking.
- Abstraction in terms of reducing complexity, representing artefacts, identifying relationships and filtering information and modeling the behaviour of systems. The ideas of modeling, systems and artefacts are thus central to how they think about abstraction.
- Evaluation in terms of defining problems, design plans for testing and stepping through algorithms step by step and rigorous arguments to justify an algorithm. This includes social and ethical norms. These are also new ideas in computational thinking in that it makes students think in terms of testing plans and implementation which maps on to real world work. The social and ethical norms provide an important perspective in thinking about where data comes from and how the results are interpreted.
- Data in terms of solution fit, effectiveness, efficiency and modeling. The idea of efficiency in using data and thinking about computational efficiency and physical considerations are thus important to their interpretation of computational thinking.

Participation in Robotics competitions

As discussed on calls, Arukay is interested in understanding how global robotics competitions link up with education around computational thinking as these competitions are of significance among students across Latin America.

Here are some of the key ideas extracted from this paper, based on mixed methods research among coaches of the World Robotics Olympiad 2019:

- The significance of robotics in STEM comes from how they convey complex mathematical and scientific thinking. It is also established that they bring to students innovative spirit and practical ability as well as computational thinking and problem-solving.
- The significance of robotics education is even among rural elementary students as demonstrated by the Children's Robot Theater. This is based on data from this paper that involves studying this phenomenon in rural China across two years and reports consistent findings.
- There is a range of global robotics competitions even beyond the World Robotics Olympiad that can be explored:
    - Botball

    - FIRST organization
    - RoboCup Junior
- The focus in such competitions tends to be on teaching robot design, assembly, coding, operations and modifications.
- Particularly in China, there is an emphasis on training agencies that specialize in teaching students how to build the robots quickly
- Coaches' feedback suggests a general consensus on how students who take part in such competitions improve programming skills and develop knowledge and skills around teamwork, career planning, interacting with foreigners and bringing honors to their country. The main areas of improvement are consistent concentration and team cohesion throughout the project.

Based on this, we reviewed another paper reviewing how computational thinking is integrated into secondary education.

- The focus in this paper is integrating computational thinking through project based learning into a secondary school in Barcelona, Spain
- The purpose of such programs is to lay the foundations for future programmers
- These programs also need to be seen as a means to develop a person for society
- Data collected has found that participating in such programs improve their performance and motivation
- The involvement in such programs improves computational thinking across concepts, practices and perspectives

We feel that these points should be communicated by Arukay to stakeholders who are thinking how computational thinking benefits them.

Given the significance of Raspberry Pi Foundation to global growth in computational thinking, we identified a series of possible collaborations that you could reach out to them for:

- They support Code Clubs around the world with resources and training and many of these activities are ancillary to what Arukay might want to take up
- They look for research partners particularly in areas like: developing frameworks for teaching AI/ML/data science to young people, role of language in the programming classroom and the support from AI tools, computer pedagogy in formal and informal settings, the impact of different approaches to curriculum and policy and the value of physical interfaces in teaching computing
- There is particular focus on the recent Experience AI initiative to provide free resources to teachers on AI literacy and distribute these through partners
- One of the concerns raised on a previous call was the expense around Raspberry Pi products. We'd like to emphasize that it is possible to collaborate with them on education and computational thinking without specifically deploying their hardware.

Even if you are interested in using their hardware, there seem possibilities around fundraising either directly through them or via third parties that can make this happen.

Bebras Computational Thinking Challenge

Bebras is an international initiative designed to promote Informatics (Computer Science) and computational thinking among students of all ages. The Bebras challenge, typically integrated into classroom activities by teachers, encourages students to develop problem-solving skills through engaging tasks that can be completed on computers or mobile devices. Computational thinking, a key focus of the challenge, involves breaking down complex problems, designing algorithms, recognizing patterns, and applying abstraction—skills essential for software development and logical reasoning. The main Bebras challenge takes place during the second week of November, recognized as World-Wide Bebras Week, though many countries extend it to two weeks or run related activities year-round. These include award ceremonies, second-round challenges, summer camps, teacher workshops, and research initiatives, further strengthening computational thinking education worldwide.

- Bebras includes five computational thinking skills:

Table adapted from Bebras, Computational Thinking Cheat Sheet (2022), https://www.bebras.org/

CT Skill	**How to spot use of the skill**
Abstraction *Focusing on the important information only, ignoring irrelevant detail*	Hiding unnecessary details; Spotting key elements in problem; Choosing a representation of a system
Algorithmic Thinking *Developing a step-by-step solution to the problem, or the rules to follow to solve the problem*	Thinking in terms of sequences and rules; Executing an algorithm; Creating an algorithm
Decomposition *Breaking down a complex problem or system into smaller, more manageable parts*	Breaking down tasks; Thinking about problems in terms of component parts; Making decisions about dividing into sub-tasks with integration in mind, e.g. deduction

Evaluation *Ensuring that your solution is a good one.*	Finding best solution; Making decisions about whether good use of resources; Fit for purpose
Pattern Recognition *Looking for similarities among and within problems*	Identifying patterns as well as similarities and connections, and identifying when patterns are not fully established; Extrapolating or interpolating data; Putting repeated instructions into a loop or function;

Summary

This report presents a summary of four key deliverables developed as part of the LEAP project—a collaborative initiative between four LEAP Fellows and the Arukay organization.

The first deliverable focused on validating Arukay's existing measurement framework. This included a review of current frameworks and baseline quizzes, a literature review guided by Arukay's core questions and a rapid review to align findings with the framework. It concluded with tailored recommendations to enhance Arukay's current tool.

The second deliverable built on this foundation by reviewing methodologies for data collection and analysis, assessing rater alignment, and conducting an analysis of the psychometric properties of the Arukay measurement tool.

The third deliverable expanded these recommendations by suggesting concrete ways to strengthen and test a revised tool—both with existing resources and through future internal or external research—with particular attention to measurement validity and statistical analyses.

The fourth deliverable provided a strategic perspective on Arukay's broader impact and future directions. It highlighted opportunities to position Arukay more effectively within the global conversation on computational thinking, and to attract interest and funding from major foundations and international organisations operating in the same space of computational thinking and young learners.

The key aspects of this project's contribution are visually summarised in Figure 4, followed by a list of key take-aways.

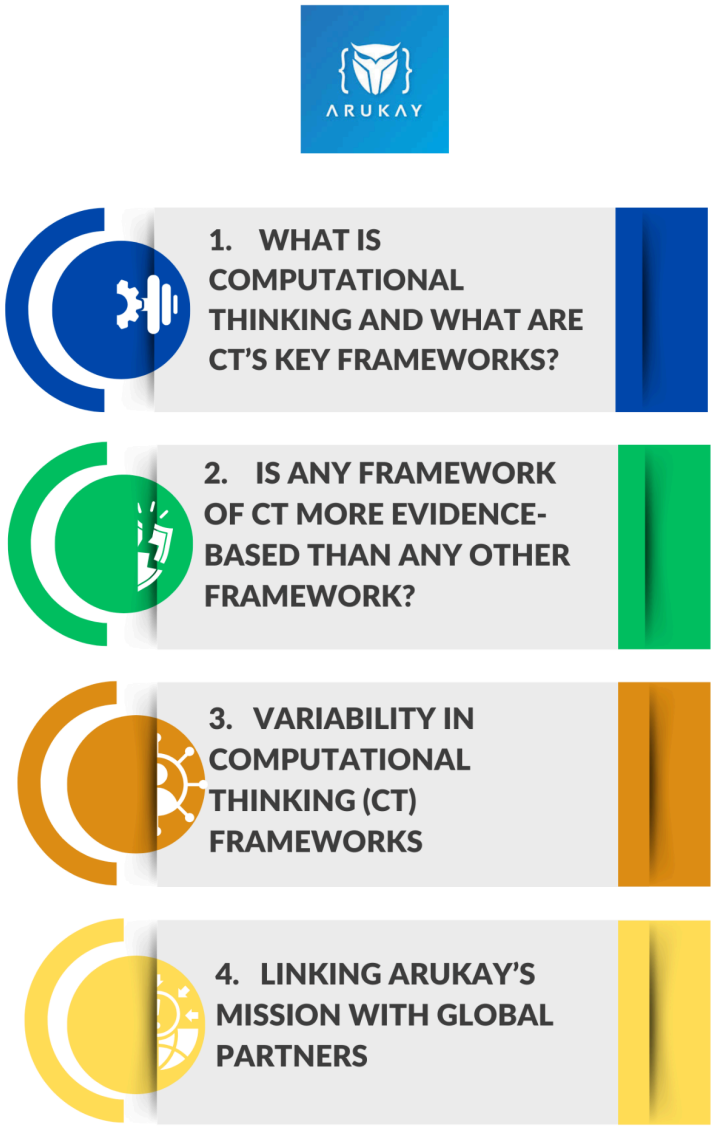


Figure 4: Visual summary of the key project's contributions

What is computational thinking and what are CT's key frameworks?

- Computational thinking consists of four key techniques (decomposition, pattern recognition, abstraction, and algorithms), which work together to structure and solve problems efficiently.
- Computational thinking extends beyond programming, serving as a cross-disciplinary problem-solving approach applicable in fields like mathematics, science, and the arts.

- Key frameworks include Papert's exploratory learning approach (1980), Wing's conceptualization of CT as a universal skill (2006), Brennan & Resnick's three-dimensional framework (2012), Tikva & Tambouris's five-area model (2021), and Zapata-Cáceres's computational thinking assessment (2020).
- Ongoing research aims to integrate socio-emotional aspects with computational skills to develop a more comprehensive framework that enhances both cognitive and collaborative learning.

Is any framework of CT more evidence-based than any other framework?

- Systematic reviews have probed how we can teach teachers to use CT in their classrooms. The current consensus is that future research should investigate what constitutes "good CT instruction" and how it can be effectively measured with robust assessment tools.
- There is some emerging evidence that teaching elements of CT improves problem-solving, critical thinking, social skills, self-regulation, and other academic skills such as reading and spelling. There is limited evidence for the evidence base of adopting one framework over others.
- There are several assessments of CT that have been developed and evidence that some measures show reasonable reliability. Assessments should include multiple choice correct/incorrect answers as well as self-reported evaluations of the level of CT that students adopt (such as the Computational Thinking Scale, Tsai et al, 2020).
- Ocampo et al (2024) provide the latest systematic review of instruments to assess CT.

Variability in Computational Thinking (CT) Frameworks

- CT frameworks are shaped by cultural values, educational goals, and technological access, influencing how CT is taught, with some cultures emphasizing group collaboration and cooperation, while others may prioritize individual problem-solving.
- Frameworks are tailored to students' cognitive and socio-emotional development, from hands-on tools for younger children to complex problem-solving for older students.
- The scope of CT frameworks varies across disciplines and educational contexts, with a focus on foundational skills in K-12 and technical specialization in higher education.
- CT frameworks emphasize collaboration, creativity, and problem-solving, with tools like Scratch used in elementary and middle schools for interactive learning.

Linking Arukay's Mission with Global Partners

- For growth and sustainability, Arukay should look to the significance of computational thinking to global foundations as well as applications to practical competitions such as robotics ones.

- Bebras and Raspberry Pi Foundation have their own practical frameworks with some relevance to how Arukay frames its content. Arukay can review these as a way of associating with such brands. There is also the potential to partner with Raspberry Pi Foundation through various models proposed
- There is some synergy with the skills needed to succeed in global robotics competitions and Arukay can stress this relationship in building its profile and partnerships

References

AERA, APA, & NCME. (2014). *Standards for Educational and Psychological Testing*. Washington, DC: American Educational Research Association. Available from https://www.testingstandards.net/uploads/7/6/6/4/76643089/standards_2014edition.pdf

Anastasi, A., & Urbina, S. (1997). *Psychological Testing* (7th ed.). Prentice Hall.

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introducing Computational Thinking and programming in compulsory education*. Report prepared for the Nordic@BETT2018 Steering Group. https://doi.org/10.17471/54007

Brennan, K., & Resnick, M. (2012, April). *New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 annual meeting of the American educational research association* (Vol. 1, p. 25).

Brown, T. A. (2015). *Confirmatory Factor Analysis for Applied Research* (2nd ed.). New York: The Guilford Press.

Crocker, L., & Algina, J. (2006). *Introduction to Classical and Modern Test Theory*. Mason, OH: Cengage Learning.

Digital Promise (online). What is Computational Thinking? Available from: https://digitalpromise.org/initiative/computational-thinking/computational-thinking-for-next-generation-science/what-is-computational-thinking/

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of computers in education*, *4*, 355-369.

Eisenberg, N., Spinrad, T. L., & Knafo-Noam, A. (2015). Prosocial development. In M. E. Lamb & R. M. Lerner (Eds.), *Handbook of child psychology and developmental science: Socioemotional processes* (7th ed., pp. 610–656). John Wiley & Sons, Inc. https://psycnet.apa.org/doi/10.1002/9781118963418.childpsy315

Erikson, E. H. (1963). *Childhood and society*. Norton & Company.

Gouws, L., Bradshaw, K., & Wentworth, P. (2013, October). First year student performance in a test for computational thinking. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference* (pp. 271-277).

Grover, S., & Pea, R. D. (2013). Computational thinking in K-12 education. *Educational Researcher, 42*(1), 38-43. https://doi.org/10.1145/2462782.2462801

Haynes, S. N., Richard, D. C. S., & Kubany, E. S. (1995). Content validity in psychological assessment: A functional approach. *Psychological Assessment, 7*(3), 238–247.

Holgado-Tello, F. P., Chacón-Moscoso, S., Barbero-García, I., & Vila-Abad, E. (2010). Polychoric versus Pearson correlations in exploratory and confirmatory factor analysis of ordinal variables. *Quality & Quantity, 44*(1), 153–166.

Kikas, E., Henn, A., & Kärt, O. (2020). A national approach to teaching computational thinking: The Estonian experience. *Computer Science Education, 30*(4), 1-18. https://doi.org/10.1007/s11423-020-09718-5

Kilic, D., Uysal, D., & Atar, B. (2020). Investigation of the construct validity of a binary scored scale by factor analysis: An application in STEM education. *Participatory Educational Research, 7*(3), 89–107.

Kim, M., Winkler, C., & Talley, S. (2021). Binary item CFA of behavior problem index (BPI) using Mplus: A step-by-step tutorial. *The Quantitative Methods for Psychology*, *17*(2), 141-153.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in human behavior*, *72*, 558-569.

Liu, Z., Gearty, Z., Richard, E., Orrill, C. H., Kayumova, S., & Balasubramanian, R. (2024). Bringing computational thinking into classrooms: a systematic review on supporting teachers in integrating computational thinking into K-12 classrooms. *International Journal of STEM Education*, *11*(1), 51.

Lourenço, O., & Machado, A. (1996). In defense of Piaget's theory: A reply to 10 common criticisms. Psychological review, 103(1), 143.https://doi.org/10.1037/0033-295X.103.1.143

MacCallum, R. C., Widaman, K. F., Zhang, S., & Hong, S. (1999). Sample size in factor analysis. *Psychological Methods, 4*(1), 84–99.

National Research Council, Division on Engineering, Physical Sciences, Computer Science, Telecommunications Board, & Committee for the Workshops on Computational Thinking. (2010). *Report of a workshop on the scope and nature of computational thinking*. National Academies Press.

Nunnally, J. C., & Bernstein, I. H. (1994). *Psychometric Theory* (3rd ed.). New York: McGraw-Hill.

Ocampo, L. M., Corrales-Álvarez, M., Cardona-Torres, S. A., & Zapata-Cáceres, M. (2024). Systematic review of instruments to assess computational thinking in early years of schooling. *Education Sciences*, *14*(10), 1124.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Piaget, J. (1952). *The origins of intelligence in children*. International Universities Press.

Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education*, *128*, 365-376.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., & Brennan, K. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Shen, L., Mirakhur, Z., & LaCour, S. (2024). Investigating the psychometric features of a locally designed computational thinking assessment for elementary students. *Computer Science Education*, 1–20. https://doi.org/10.1080/08993408.2024.2344400

Tikva, C., & Tambouris, E. (2021). *Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature review. Computers & Education, 162*, 104083.

Tsai, M. J., Liang, J. C., & Hsu, C. Y. (2021). The computational thinking scale for computer literacy education. *Journal of Educational Computing Research*, *59*(4), 579-602.

Vihavainen, A., Paksula, M., & Järvinen, H. (2014). Exploring the relationship between problem-based learning and computational thinking in programming education. *Computer Science Education, 24*(2), 137-156. https://doi.org/10.1007/s11357-014-8651-6

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

Wolf, E. J., Harrington, K. M., Clark, S. L., & Miller, M. W. (2013). Sample size requirements for structural equation models. *Educational and Psychological Measurement, 73*(6), 913–934.

Wong, S.-C. (2020). Competency Definitions, Development and Assessment: A Brief Review. *International Journal of Academic Research in Progressive Education and Development,* 9(3), 95–114.

Zapata-Cáceres, M., Martín-Barroso, E., & Román-González, M. (2020, April). *Computational thinking test for beginners: Design and content validation. IEEE Global Engineering Education Conference (EDUCON), 1905-1914.*