Received: 14 August 2023 | Revised: 16 February 2024 | Accepted: 3 March 2024

DOI: 10.1111/jcal.12972

ARTICLE

Journal of Computer Assisted Learning WILEY

Primary school students' perceptions and developed artefacts and language from learning coding and computational thinking using the 3C model

David A. Martin [1] | Peter Curtis [2] | Petrea Redmond [2]

[1]School of Education, Edith Cowan University, Perth, Western Australia, Australia

[2]School of Education, University of Southern Queensland, Toowoomba, Queensland, Australia

Correspondence
David A. Martin, School of Education, Edith Cowan University, Perth, WA, Australia.
Email: da.martin@ecu.edu.au

Abstract

Background: A resurgence in teaching coding in primary school classrooms has led to a pedagogical swing towards using physical computing and coding to develop students' use of algorithms, computational thinking, and problem-solving skills. Two obstacles impede the optimal development of these objectives: the availability of a suitable pedagogy and an instructional sequencing model for primary school teachers to effectively present coding and computational thinking concepts and skills to students in alignment with their developmental stage.

Objective: This study aims to address both obstacles by introducing the 3C Model, a newly developed instructional sequence grounded in established pedagogies and designed to effectively teach coding and computational thinking skills to primary school students based on their developmental stage.

Methods: The qualitative study employed two data sources to triangulate findings, using: (1) semi-structured interviews and thematic analysis to investigate 11 primary school students' perceptions of their learning experiences with the 3C Model, and (2) researcher observations along with reflections of the students' developed and demonstrated learning through the method of knowing-in-action, reflection-in-action, and reflection-on-action.

Results and Conclusions: The findings of this study fill a gap in the existing literature by demonstrating that the pedagogical and sequential approach embedded in the 3C Model not only enhanced students' engagement levels but also resulted in improved curriculum learning outcomes. The 3C Model provides teachers with a coherent and age-appropriate instructional structure. It uses physical computing devices and digital coding platforms to introduce coding concepts, furthering the development of computational thinking skills in primary school students beyond mere procedural and rote learning.

Implications: The study holds important implications for practical applications, as it addresses an absence in the literature of an established pedagogy and instructional

sequencing model for effectively teaching coding and computational thinking concepts and skills to primary school students. Drawing on established pedagogical and developmental learning theories, the 3C Model provides primary school teachers with an engaging, age-appropriate instructional method that avoids decontextualised teaching and surface-based learning. Instead, it encourages collaborative student work and contextualised learning, steering away from isolated and generic approaches.

KEYWORDS
3C model, coding, computational thinking, digital technologies, mathematics instruction, primary school

1 | INTRODUCTION

Coding instruction in the primary school classroom was initially motivated by the work of Seymour Papert (1980). The term coding refers to the concept and skill of programming (Mills et al., 2021; Woo & Falloon, 2023). Programming is the process of creating a sequence of instructions designed to have a digital system execute a specific task or solve a particular problem (Zeng et al., 2023). In Australia, where this study was conducted, the national curriculum includes Digital Technologies as a subject area, which requires primary school classroom instruction in computer coding and computational thinking (Australian Curriculum, Assessment and Reporting Authority [ACARA], 2023a). One aim is for students to use computational thinking and digital systems to define, design and implement digital solutions to contextually authentic problems (ACARA, 2023b). Resources such as work samples, curriculum connections, activities and tools are available for primary school teachers to use for planning and teaching coding and computational thinking (ACARA, 2023b; Rich et al., 2022; Williams, 2021). As an outcome of students' programming activities, Papert (1980) believed a shift in pedagogy would emerge. However, generalist primary school teachers have been tasked with teaching coding and computational thinking skills without being provided a technology-specific pedagogy; consequently, they resort to employing general pedagogical strategies (Bjursten et al., 2023; Woo & Falloon, 2023). What remains a gap in the literature is an age-appropriate, specific activity sequencing model for guiding instruction which helps students break down problems into parts, defining abstract coding concepts and designing and implementing algorithms (Australian Catholic University, 2023; Bjursten et al., 2023; Woo & Falloon, 2023). Moreover, there is a lack of consensus on the best approach to teaching these abstract concepts and skills (Dag et al., 2023; Mason & Rich, 2019; Rich et al., 2022). Consequently, primary school teachers may encounter challenges in planning coding activities for students aged 7–11, who are developmentally at Piaget's (1964) concrete operational stage in terms of cognitive performance. As a result, a critical missing component exists in the relationship between task design and pedagogy when teaching coding and its associated computational thinking skills to this age group. This paper addresses the missing component by investigating the efficacy of a new instructional sequencing model, termed the 3C Model (Martin, Curtis, Redmond, & Byrne, in press).

2 | LITERATURE REVIEW

A key aim of the Australian Digital Technologies curriculum is that students use computational thinking skills when collaboratively defining, designing, managing, evaluating and implementing digital solutions to authentic problems (ACARA, 2023a). The Digital Technologies curriculum identifies elements of computational thinking as abstraction, data collection, representation and interpretation; specification; algorithms; and implementation (2023a). Computational thinking as a skill and process helps students organise data logically by decomposing problems into parts. It helps them in defining abstract concepts, which is needed when creating and using algorithms (ACARA, 2023b). In the Digital Technologies subject, coding is viewed as a vehicle for developing students' computational thinking skills (ACARA, 2023a). The authors adopted the above aim, definition and rationale from the Australian Curriculum: Technologies curriculum for this research project.

2.1 | Piagetian cognitive development theory as a basis for the 3C model

Piaget's (1964) theory of cognitive development defines four stages through which children progress in their ability to understand and process information: The Sensorimotor Stage (Birth to 2 years), the Preoperational Stage (2–7 years), the Concrete Operational Stage (7–11 years) and the Formal Operational Stage (11 years and older). When applying these age ranges to the students in this study, aged 10 and 11, they were likely primarily operating within Piaget's concrete operational stage. During this stage, children exhibit some capacity for abstract thinking. During this stage of students' learning, Piaget emphasised the importance of employing concrete experiences to help learners grasp the fundamental principles, concepts, and skills before transitioning to abstract representations.

Drawing from Piaget's (1964) theory, the 3C Model (Context, Capability, Computation) (Martin, Curtis, Redmond, & Byrne, in press) was designed to teach coding and computational thinking concepts and skills to primary school children (ages 7–11) who are performing primarily at the concrete operational stage. Broadly, the 3C Model is designed to leverage the students' emerging logical thinking by integrating physical computing, physical movement and age-appropriate language representations to enhance their learning.

2.2 | Physical computing and computational thinking

Physical computing is defined by Przybylla and Romeike (2017) as tangible interactive objects or systems, such as sensors and motors, which are programmable and communicate with their environment. Kastner-Hauler et al. (2022) described physical computing as connecting a computing device that is equipped with sensing capabilities to the environment outside the confines of raw or block coding. In classroom settings, physical computing involves students using robots or other programmable hardware for sensing and communicating with their environment. The procedure is designed for students to create tangible artefacts using a design process and their imaginations resulting in the creation of their own learning experiences (2022).

Key competencies that should be a result of students engaging with coding physical computing devices are shared by Przybylla and Romeike (2015) and include:

- Understanding computing systems from a hardware and software point of view.
- Formulating problems in terms of clearly describing what is supposed to happen from an outside perspective.
- Organising and analysing real-world data collected from the student's environment utilising measuring devices they designed and constructed themselves.
- Algorithmic thinking—where students describe a series of events, either in series or parallel, that control the response to incoming data.
- Troubleshooting and problem-solving. These required skills become immediately noticeable in physical computing, for example, when the directions or input selection are poorly coded, the undesired output (result) is instantly apparent, requiring students to re-evaluate their algorithmic design.

Kastner-Hauler et al. (2022) suggest six computational concepts as outcomes that should result from primary school students engaging with coding physical computing devices: (1) sequences, (2) simple loops, (3) nested loops, (4) IF-THEN conditionals, (5) IF-THEN-ELSE conditionals, and (6) WHILE conditionals.

This paper presents how primary school students can learn these concepts and achieve these key competencies through using physical computing devices, such as those found in LEGO® MINDSTORMS®, which offers a physical robotic device equipped with sensors having various capabilities. These include the light sensor, capable of detecting colour and ambient light, the proximity sensor which estimates distance using infrared technology, the ultrasonic sensor which determines distance by emitting an infrared signal and measuring its reflection time, and the touch sensor which responds to button pressure. The sensor system also includes features for displays, attenuating motors and speakers. The application of each sensor can prompt students to engage in complex thinking. Both Przybylla and Romeike (2017) and Kastner-Hauler et al. (2022) concur, noting that the distinctive capabilities of the sensor systems facilitate learning and the acquisition of related concepts and competencies.

In determining the best pedagogical approach to developing the concepts and competencies related to coding and computational thinking, Wing (2009) cautioned against focusing on coding procedures rather than conceptual approaches in the teaching process. Simply focusing on the code as a skill to learn without emphasising computational thinking, systems thinking and design thinking in the context of creating solutions to authentic problems will result in shallow learning. The following section outlines existing research into instructional methods designed to develop students' computational thinking skills through the application of coding physical computing devices.

2.3 | Existing pedagogical approaches

Various pedagogical approaches were identified by Sentance et al. (2017), ranging from open-ended and unguided constructivist approaches to more teacher-directed, skill-based programs. One open-ended variety, termed 'bricolage' (Levi-Strauss, 1966), encourages using physical materials and purposeful tinkering, focusing on trial-and-error exploration when seeking solutions to ill-structured problems with undefined problem-solving processes. Also mentioned was a blended pedagogical approach which could include traditional instruction that relies on directly teaching some skills and concepts to supplement students' trial-and-error exploration and discovery learning. This blended approach aims to build initial skills by showing sample projects, delivering short code concept lessons, presenting starter code examples and having students read code. At the more directed end of the spectrum are hierarchical skill lessons, often presented in a decontextualised manner. Sentance et al. (2017) note that each teaching approach has implications regarding its effect on student engagement patterns. A consequence of this approach for students was cognitive and reading overload due to the density of the accompanying materials, leading to a potential loss of independence, engagement and creativity. Critically, this outcome was compounded by a lack of concept internalisation; instead, the learning outcomes focused on students' surface level and procedural knowledge.

When investigating how pedagogical models translate into engagement and instructional practice, Przybylla and Romeike (2017) proposed an instruction sequence when engaging with physical computing. The process of teaching physical computing begins with a motivational, engaging phase, usually in the form of a problem-solving context. It continues with a technical introduction, concluding with project work

and presentation of that work. During the technical introduction, based on the complexity of the content, the teacher makes pedagogical adjustments ranging from adopting a closely guided, step-by-step approach to more open-ended approaches such as tinkering (2017). The challenge with this approach is that the rationale for its use is based on a conglomeration of compensatory methods. This fragmentation of instruction highlights the importance of a pedagogical sequence that employs the appropriate types of engagement for learning.

In exploring the predictive effect of Year 3 through Year 6 students' STEM learning attitudes on their computational thinking skills, Sun et al. (2021) found that their attitude towards STEM learning significantly predicted the development of their computational thinking skills. The link between computational thinking and STEM learning has been further examined by Sirakaya et al. (2020) with Year 5 to Year 8 students. Using structural equation modelling, they found that STEM attitudes and the way students prefer to process information had a significant effect on computational thinking skills. Each of these findings should be considered when planning lessons. Lacking in the research designs are examinations of the pedagogical approaches used during the investigations.

During the 33rd annual conference of the Society for Information Technology and Teacher Education, Statti and Torres (2022) advocated for coding instruction in primary schools as a vital skill. Their paper's title implies the presentation of best practices for teaching coding to primary school students; however, the authors primarily provided resources and tools for teaching coding activities, such as the software program Scratch along with an explanatory activity. Consequently, a pedagogical sequence for teaching coding was not a focus of the paper.

In a study conducted by Kastner-Hauler et al. (2022), the impact of a set of teaching units on the computational thinking skills of 45 primary school students aged 8–10 was examined. The study utilised coding using the online block-based programming tool MakeCode and physical computing using the micro: bit within an integrated learning environment. The integrated learning environment involved two teachers delivering three units of work. Regarding the pedagogical strategies employed, Kastner-Hauler et al. (2022) proposed the teacher conceal all unnecessary command blocks at the start of unit 1 while presenting the activity that introduces the coding program. This approach appears to incorporate elements of inquiry learning. However, the explanation provided for employing this strategy at this stage of the instruction does not aim to elicit inquiry. The explanation provided is that "The hiding makes the initial orientation and the focus on the essential parts for the entry immensely easier" (p. 5). Overall, the instructional process of unit 1 began with a sequence of passive learning activities as tutorials, video presentations, and explanations covering the MakeCode programming environment and micro: bit simulator. Following those activities, the students connected the micro: bit and uploaded a program, at which time the students were given opportunities to test the code directly on the device, display output and make refinements. Unit 2 involved an introduction to the concept of triggering an event, followed by self-exploration of the use and capacities of the micro: bit's buttons. Unit 3 comprised sensor functionality and more physical computing, for example, using the micro: bit software maker's online guided tutorial which students reproduced to simulate the rock-paper-scissors game. The impact of the intervention was measured using the Beginners Computational Thinking Test (Zapata-Cáceres et al., 2021) in a pre- and post-test design. The significant results validated the effectiveness of combining block-based coding and physical computing to enhance computational thinking skills in primary school students. While the Kastner-Hauler et al. (2022) intervention did serve as a functional method for introducing coding and computational thinking, it revealed certain shortcomings that we also encountered in our study, ultimately limiting the students' achievement of key learning outcomes.

In the initial implementation of our study, students were similarly engaged in replicating code to create, for instance, the previously mentioned rock-paper-scissors game. Our experiences reinforced our belief that replicating code through follow-along projects ignored key pedagogical elements. Consequently, we recognised the need to reassess our teaching approach which led to the redevelopment of the 3C Model (Martin, Curtis, Redmond, & Byrne, in press). In the next iteration of the intervention, the instructional sequence of the 3C Model was modified to include: (1) the application of computational, design, processes and production, and project management skills in an authentic context, (2) an element that considers language development, and (3) the teaching of coding and the physical device's capabilities using physical materials and movement. These changes to the instructional sequence were based on the Language Model (Irons, 2014; Irons & Irons, 1989), an instruction approach used for teaching the abstract concepts and skills of mathematics, as explained in the next section. Similarly, the 3C Model was designed to ensure the students' learning was initially conceptual not procedural, collaborative not isolating, and contextualised not generic. Further, the lessons were meaningfully connected to self-initiated student experiences which encouraged creative responses to problems rather than the replication of existing code or projects.

The current study aims to fill the research gap by establishing the 3C Model as an appropriate pedagogical approach for introducing coding and computational thinking concepts to primary school students at the concrete operational stage (Piaget, 1964), in alignment with their developmental stage and the aims and expected learning outcomes of the Digital Technologies subject (ACARA, 2023a). The process involves using physical interactive digital devices, movement, staged language development, and subsequent coding procedures within the context of an engaging, authentic problem.

2.4 | Conceptualising the 3C activity sequencing model

One aim of the Australian Technologies curriculum is for primary school students to use computational thinking and digital systems to define, design, create, manage, implement and evaluate digital solutions to contextually authentic problems (ACARA, 2023b). The 3C Model (Martin, Curtis, Redmond, & Byrne, in press) was

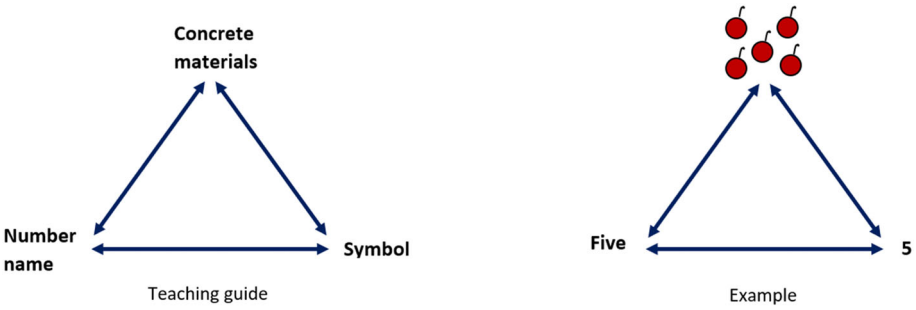

FIGURE 1 Rathmell's triangle for teaching mathematics concepts (modified from Payne & Rathmell, 1975).

conceptualised to attain these learning outcomes primarily through contextualised learning experiences. This alternative pedagogical approach is based on the Language Model (Irons, 2014; Irons & Irons, 1989), which also employs contextual problems, and interactions between physical movement, concrete materials and language representations, for developing mathematics concepts and skills.

2.4.1 | The language model for teaching mathematics

The Language Model (Irons, 2014; Irons & Irons, 1989) draws from the work of Payne and Rathmell (1975) who proposed a teaching guide illustrated as a triangular structure (known as Rathmell's Triangle) for developing mathematics concepts (Figure 1). The arrows that form the triangle establish the basis of the teaching approach, which is to prompt students to develop relationships between concrete and graphic representations of a number's value, the number itself, and the number expressed in words.

Activities such as matching the physical quantity and number name, matching the number symbol and number name, or matching number symbols to the physical quantity aid in developing abstract thought as the students accumulate various representations and conceptual understanding from synthesising the connections between the three representations of the number. Extending the scope of learning beyond the connections between quantity, number name, and symbol, the Language Model (Irons, 2014; Irons & Irons, 1989) considers physical movement with concrete materials, such as acting out a subtraction operation in the context of an authentic scenario. Additionally, it incorporates the explicit language teachers should demonstrate and students should use as they transition from concrete operational thinking to abstract thought and reasoning (Piaget, 1964). Since mathematics, like programming, has a language of its own, this makes sense. As a learning strategy, students should be given opportunities to practice the language as they share strategies and solutions to problems. Students are more likely to learn concepts and skills when they have well-defined approaches to describing and discussing their experiences (Australian Education Council, 1991; Miller, 2019).

As the basis for the 3C Model (Martin, Curtis, Redmond, & Byrne, in press), the Language Model (Irons, 2014; Irons & Irons, 1989) begins with representation in context. For example, to simulate the concept and skill of division (as an operation) students are instructed to create a story where they apportion a collection of familiar objects by physically sharing them out equally among their group members, a strategy that embeds their learning in a context that is familiar and meaningful. During this activity, the expansion of Rathmell's triangle to include language development becomes apparent when the teacher prompts the students to verbally articulate the 'sharing out' process using language from their existing schema. Next, the students are asked to use diagrams or pictures to represent their division stories while using more advanced language the teacher used during modelling. In the final stage of the activity, the teacher translates students' stories into division algorithms on the whiteboard as they physically and rhetorically act them out. This strategy helps the students link the physical and graphic representations of sharing to the corresponding numbers, symbols and symbolic language. Once the students develop this link through the Language Model's pedagogical approach, the abstractions inherent in division algorithms become obvious. Deeper conceptual understanding and learning representations arise from both motoric and language-based pedagogical approaches, which steer away from relying solely on procedural and rote learning. The students' deep understanding becomes apparent in their ability to generalise, modify and apply mathematical concepts to other authentic problem-solving scenarios.

Informed by the theoretical and pedagogical approaches described, the instructional sequence of the 3C Model (Martin, Curtis, Redmond, & Byrne, in press) provides a method for developing a firm foundation of conceptual understanding and abstract thinking in primary school students that corresponds with their developmental stage of learning. The following section describes the key elements and intent of the 3C Model and provides examples of teaching segments and student responses.

3 | THE CONTEXT/CAPABILITY/COMPUTATIONAL STAGES OF THE 3C MODEL

3.1 | What is the context of the problem?

Learning coding within a contextual problem is a critical pedagogical element, emphasising collaborative problem-solving as a means to learn new content (ACARA, 2023b). During this first stage of the 3C Model (Martin, Curtis, Redmond, & Byrne, in press), the teacher

selects a contextual problem, usually linked to another curriculum area of study, but problems can also be sourced from children's own experiences. This first stage is critical in establishing a purpose for learning code and, consequently, is responsible for connecting all parts of the model. Examining the problem in context connects the device's capability, the developed computational thinking concept, and familiar language and action. For example, a Year 4 student may investigate safety as part of a health unit. The teacher begins by developing with the students, through an inquiry approach, contexts where they have identified safety as an issue. The teacher may scaffold this process by providing prompts or simple guided research tasks such as surveying community members. The students then come up with situations as familiar, safety-related contexts and problems such as: (a) pool latches that may be left open, (b) train station platforms being unsafe for toddlers, (c) cars reversing out of driveways and (d) visually disabled people finding their way through unfamiliar locations.

3.2 | What is the capability of the device?

In the second stage of instruction, students are exposed to capabilities of software or physical computing devices in an exploratory or play-based activity. The teacher's responsibility is to allow the students to observe pre-coded devices as they execute a program without revealing the underlying code. The strategy here is to enable the students to inductively determine the device's capabilities and elicit from them explanations that depict the code. This step is critical to establishing and reinforcing the connection between the system's input, process and output functions. A suitable activity to demonstrate the light sensor's capability is to connect it to the LEGO® light brick and allow students to observe what happens to the light readings as they point the light sensor at various light sources or reflections. The teacher's role is to frame the students' discoveries in familiar terms to support the development of conceptual coding ideas and elicit language that is reflective of branching statements such as, "IF the sensor is pointing towards a bright light, THEN the reading will go all the way to 255" or "IF the light sensor is pointing to black cardboard, THEN the reading will go down".

3.3 | What is the focus concept in terms of computational thinking?

In this third stage, there is an intentional and gradual shift in language from child-familiar terms to the more abstract representations of coding. This stage is designed to allow students to make the connection between algorithmic concepts using the verbal and modelling aspects of the Language Model (Irons, 2014; Irons & Irons, 1989). To illustrate this, consider the scenario where the teacher is considering teaching the branching concept of 'IF … THEN' statements and using variables. Using the safety context developed in the first stage, the teacher presents students with a physical representation of a problem, such as a visually disabled person moving through an unfamiliar location. This may be achieved by establishing a maze in the classroom, constructed on the classroom floor with masking tape. Within the maze are obstacles, such as rocks. As students encounter the rocks in the maze, they place them in the bucket and keep a running record of how many rocks have been encountered. The computational thinking focus is illustrated in Figure 2, which shows two students engaged in a body syntonic and language activity that explores the notion of variables.

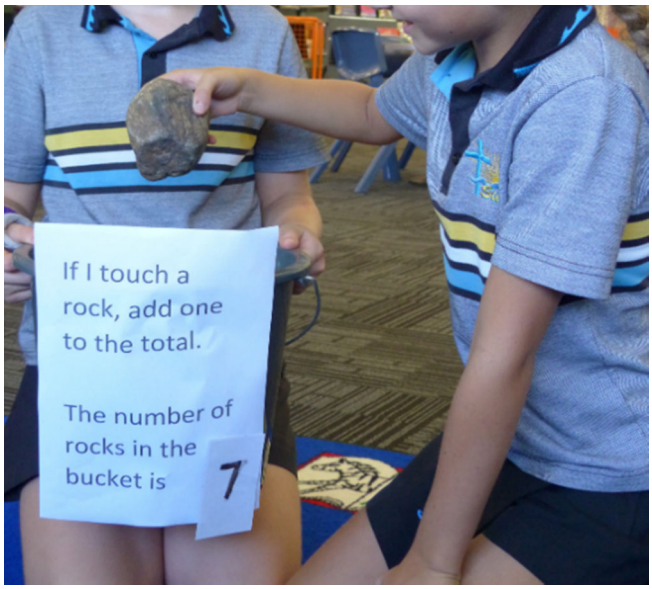

FIGURE 2 Students physically act out the code (body syntonic).

The code is written in everyday language, and the physical motion of acting out the code assists students in developing a conceptual understanding of decision branching and variables. Similar to the Language Models' pedagogical approach, these motoric, physical experiences, and language use segue into introducing the symbolic block code. At this point, the teacher shows the students the related code and draws connections between the written statements and that code. On the left side of Figure 3 is the everyday language developed by the students and used to describe the algorithm of finding each rock in the maze and adding it to the score. The corresponding visual block coding on the right describes the same action.

With a solid foundation of the task design and pedagogy for teaching coding and computational thinking skills, this study progressed to the delivery of the 3C activity sequencing model. The research problem, that there was a missing critical component in the relationship between task design and pedagogy when teaching coding skills, was addressed by the research question: *What artefacts, language representations, and student perceptions developed from learning coding and computational thinking using the 3C activity sequencing model?*

4 | METHOD

4.1 | Design

The research study adopted a qualitative design that utilised methodological triangulation involving two data sources: semi-structured

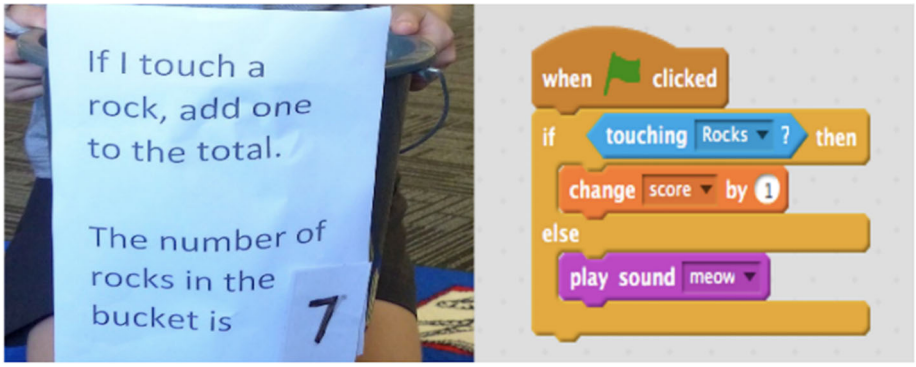

FIGURE 3 Linking everyday language to related abstract block code.

interviews and researcher observations along with reflections. Semi-structured interviews were subjected to thematic analysis to explore and examine the students' lived experiences during their engagement with the learning of coding and computational thinking through the 3C Model (Martin, Curtis, Redmond, & Byrne, in press). The researchers sought to capture the students' perspective on the value of the novel approach by eliciting their feedback on how it served as a means for showcasing their knowledge and language development through the creation and demonstration of artefacts using physical computing devices. The thematic analysis design reflects Clarke and Braun's (2014) 'top-down' approach. The process involves identifying common patterns within data sets, which are coded and then arranged into themes representing central organising concepts.

Researcher observations and reflections formed the second data collection method providing triangulation of the findings and further insights into the effectiveness of the 3C Model's instructional sequence. Reflection is central to effective teaching. The notion of reflective practice derives from Dewey (1910, 1933) and Schön (1983, 1987) but continues to inform more recent research (DeLuca et al., 2023; Shah, 2022). Dewey (1933) considered "reflection" to be a process that is both "active and intentional" (DeLuca et al., 2023, p. 6). This process entails "persistent, and careful consideration of any belief or supposed form of knowledge" (Dewey, 1933, p. 9). Through examining their "experiences [and] forming thoughts and ideas about them in relation to" their knowledge of teaching theory and their students, along with feedback from students and peers, teachers act on those thoughts to alter their teaching (Carrington & Selva, 2010, p. 45; Schön, 1987). The method for analysing the researcher's observations and reflections was drawn from Schön's (1983) *The Reflective Practitioner*, where reflective practice is described as the means by which professionals become aware of implicit knowledge: 'knowing-in-action' (evidence of the achieved knowledge is in the implicit application of the knowledge), 'reflection-in-action' (reflecting on the understanding of the application of the knowledge), and 'reflection-on-action' (which represents the practitioner's reflections on the effects of his actions).

4.2 | Participants and setting

A purposeful sample of Years 5 and 6 students (ages 10 and 11) was sourced from a regional Catholic primary school. The sample ($n = 11$) consisted of five girls and six boys. The 11 students comprised the school's coding club and some had prior procedural knowledge in coding. The study was conducted during the days and hours the students attended coding club to minimise disruption to their core curriculum learning. The study was developed for the Australian Digital Technologies curriculum, which requires primary school classroom instruction in computer coding and computational thinking (ACARA, 2023a). Ethical considerations were informed consent, data gathering, and data management. For research participation of students under 18 years of age, both student and parent/carer consent are required and were obtained. Qualitative data collected during the focus group sessions were thematically coded and analysed using NVivo software. Only non-identifiable data were stored. This research was approved (H17REA179) by the University's Human Research Ethics Committee and conducted following all required ethics protocols.

4.3 | Delivery of the 3C activity sequencing model

With over 40 years of primary school classroom teaching experience, the insider researcher assumed the role of the teacher and is identified as the teacher in this section. The teacher implemented the 3C Model (Martin, Curtis, Redmond, & Byrne, in press) as a series of lessons within the school day, 1 h a week, with the students grouped in teams of two or three. In all, two iterations of lesson sequences were delivered over 6 months using two coding platforms. Both programs are block-coding platforms used in project-based learning using Lego bricks. The physical device used for the first set of lessons was the micro: bit and the accompanying MakeCode program. The second set used the LEGO® light brick physical device and the accompanying LEGO® MINDSTORMS® program. Using two sets of lesson sequences was purposeful as it allowed the students more learning time using the 3C Model and the researchers the opportunity to refine the model's design and pedagogy for the second iteration. The following describes the set of lessons that utilised LEGO® MINDSTORMS® and the LEGO® light brick. Throughout the section we highlight links between the 3C Model's instructional sequence and the established pedagogies from which it was derived.

Lesson 1: *Context*. The lesson began with the teacher exploring the context of a problem students encounter in their everyday lives. Alternatively, these problems can be sourced from the curriculum (for

example, Science, Mathematics, or Humanities). Either way, the students are to learn curriculum content through engaging with physical computing devices while solving an authentic problem. Drawing a parallel to the initial stage of the Language Model (Irons, 2014; Irons & Irons, 1989), the teacher introduces a mathematics concept or skill using movement with physical materials within an authentic context.

Lesson 2 addressed *Capabilities*. During this lesson, students learned about the capabilities of the LEGO® light brick. In other words, what can the tools and features of the digital device accomplish? The activity began with showing the students pre-coded programs. The students were not shown the code at this stage, nor did they code in this lesson. Rather, the teacher showed the students a completed program running and asked the students to observe and articulate, in general terms, what functions the Lego light brick can carry out. During this phase of the activity, the teacher asked key questions about their observations. The discussion outcomes enabled the students to inductively determine the device's capabilities while developing pre-cursor statements related to coding language. The strategy employed is comparable to the second and third stages of the Language Model (Irons, 2014; Irons & Irons, 1989), wherein the teacher prompts the students to enact the previously modelled concept and skill within the same context, using their own language, this time utilising substituted materials (e.g., 2-D shapes versus apples). This approach scaffolds the students' learning, enabling them to inductively develop a conceptual understanding of the functions of the skill being taught.

Computational thinking was addressed in Lesson 3. At this stage, the researcher modelled, through body movement, a set of coded sequences. Next, the students acted out the code (IF … THEN statements) to develop their knowledge and understanding. The intent was to have the students represent the code using everyday language, which represents a form of 'pseudo-code'. The modelling and acting out of the code were designed to be an iterative process that immersed the students in learning that developed their conceptual understanding of the computational thinking elements. The motoric aspect of this activity is consistent with Papert's (1980) syntonics, while the recording of these experiences aligns specifically with the third (representational) stage of the Language Model (Irons, 2014; Irons & Irons, 1989). At this stage of learning the students are ready to apply their knowledge and understanding through mathematics representations that represent the concept or skill while using mathematical words (e.g., add or plus versus and).

In Lesson 4, the teacher employed a guided discovery approach steering students to realise the connection between the code they modelled and rhetorically acted out (using the vocabulary they developed in Lesson 3) to the symbolic representation of the software's code, as illustrated in Figure 3. This teaching strategy reflects moving students from the concrete and semi-concrete stages of learning to the abstract stage. This stage of learning is consistent with the final phase of the Language Model (Irons, 2014; Irons & Irons, 1989). At this stage, the teacher prompts students to re-enact their concrete or semi-concrete representations. While doing so, the teacher presents the symbolic representations, using symbolic language. This teaching strategy synthesises for the students the connection between the physical representations of the concept or skill and its abstract (symbolic) representation.

In Lesson 5, there was no more explicit teaching. The students now had a deep understanding of the conceptual coding concept and its representation in block form and were ready to apply it in a novel situation. The teacher's role was to set up the environment for students to decide on a problem they wanted to solve. Next, the teacher took on the role of facilitator and gave the students the responsibility to research and solve the new problem they chose. For example, students identified that a recent recycling drive in the community had confused children about which waste material was suitable for which type of recycling bin. As their solution, the students repurposed a barcode scanner to read QR codes. Using IF … THEN statements, they coded a micro: bit to identify a list of items depending on their recycling categories. The device was designed to read the QR codes and then display a line of text on the micro: bit's LED, for example, *Place item in the red recycling bin*.

4.4 | Data collection

One set of data was collected using semi-structured interviews conducted in focus group settings. The student participants were interviewed twice in 6 months, timed at three- and six-month intervals to coincide with the completion of each lesson sequence. The group interview sessions were conducted in a room on school grounds by the insider researcher. The group interview sessions were approximately 30–40 min. Students were encouraged to discuss freely and share their experiences in their own words. This was accomplished by allowing students to participate in small group discussions. All group sessions were digitally audio-recorded and later transcribed verbatim.

A set of six guiding questions allowed the researchers to focus on the topics of interest broadly while allowing the student participants to initiate discussion of issues pertinent to their own experiences. The semi-structured interview questions were:

1. What was your experience like when you worked on your project?
2. What did you like?
3. What would you change?
4. What did we do together that helped?
5. What did not help?
6. What did you do when you were stuck?

Insider researcher observations and reflections (Schön, 1983) formed the second source of data. These data were derived from conducting formative assessments throughout each lesson and during the summative assessments when the students presented and articulated explanations of their digital solutions. The process involved the insider researcher: (1) observing the students' problem-solving skills and engagement throughout the lessons, as well as during the explanations and presentations of their artefacts (knowing-in-action), and (2) attentively listening to the students as they articulated the explanations of their designed digital solutions (reflection-in-action), which

served as further evidence of their learning and language development. The insider researcher's reflections completed the data set. The process involved reflecting on the events of his actions to gain further insights into the potential of the 3C Model's instructional sequence (Martin, Curtis, Redmond, & Byrne, in press) and its contribution to the students' enhanced learning (reflection-on-action).

4.5 | Data analysis

Analysis of the data collected from the focus group sessions followed the method suggested by Clarke and Braun (2014), which focuses on a rigorous process of data reading, coding and development of themes. The process underwent the following phases:

1. Familiarisation with the data through repeated readings and transcription. The audio-taped group sessions were listened to multiple times by the researchers. The researchers then proofread each transcribed session for accuracy while listening to the audio-taped group sessions.
2. Transcriptions were imported and stored in NVivo. Using the tools of NVivo, nodes were developed to identify initial codes based on the conceptual understanding of the research question. Researchers then examined the entire data set, sentence-by-sentence, for clusters and patterns of meaning and a storyline related to the research question. Sub-nodes were also developed; for instance, students commented about their experiences when seeking help, which collapsed into a code called *Resourcing*.
3. Guided by phase 2, preliminary themes were identified by sorting and collating the codes into relevant concepts that appeared across the students' experiences. For instance, a number of codes related to students' experiences when their peers had developed diverse ways to solve problems coalesced into the broader theme of the *Opportunity to develop multiple answers to the same question*.
4. Analysis, defining and naming of themes. The researchers synthesised the preliminary themes and related meanings to determine whether they accurately captured and reflected the students' rich descriptions using the research question as the basis.
5. Guided by phase 4, the researchers confirmed that the narratives created a storyline that addressed the research question yet retained the students' voices. Through this process, the researchers derived five themes:
    1. Opportunity to creatively develop multiple pathways to solutions.
    2. Availability of varied resources to discover multiple pathways to solutions.
    3. Positive challenge and the reality of perseverance.
    4. The importance of visuals.
    5. The connection between language and code.

The first three themes reflect the students' perceptions of learning with an inquiry-based, student-centred, social constructivist approach. The fourth and fifth themes are presented as a progression of their experiences from acting out computational thinking to their experiences with the visual representations and accompanying language to the development of the code.

The analysis of the second data set was derived from the insider researcher's recorded observations and reflections made during and after the formative and summative assessments, which were evaluated against the intended learning outcomes and related curriculum achievement standards. Guided by Schön (1983), the insider researcher paid close attention: (1) throughout the lessons for achieved coding concepts and computational thinking skills in the form of both language representation and explicit motoric modelling as the application of the knowledge (knowing-in-action), and (2) during the summative demonstrations as the students presented their digital solutions (knowing-in-action). Further, the insider researcher attentively listened to the students articulate their explanations, which showcased their developed language while demonstrating their solutions (reflection-in-action). As the instructor, the insider research reflected on his actions taken during the lessons which affected the quality of the learning and practical assessments (reflection-on-action).

5 | RESULTS

5.1 | Student focus group sessions

5.1.1 | Theme 1: Opportunity to creatively develop multiple pathways to solutions

Theme 1 emerged from the data based on the student responses that learning under the 3C Model's (Martin, Curtis, Redmond, & Byrne, in press) pedagogical sequence allowed them to engage in what they perceived as an open and creative process. The structure of the 3C Model allowed students to develop digital solutions that were uniquely individual and characterised by a preference for learning and a demonstration of that learning. Students recognised that their chosen pathways to their solutions differed from those of their peers. An indicative response was: *It gives you a chance to get creative, so they have just given you one task, but you can do whatever you like so long as you meet the requirements of the criteria, and everyone could end up differently*. This is a revealing response in that each group project reflected a distinctive manner of problem-solving and product.

Another feature of this open-endedness identified by participants was the opportunity to tinker, improve and refine. One student summarised this process as: *You can elaborate on it. With coding, you can always keep going, like you can always improve*.

Students noted the difference between their physical computing experiences and those in other learning areas, such as mathematics and science. Both learning areas, particularly mathematics, were seen as requiring them to provide a particular process and 'correct' answer. An indicative comment was: *Well, in mathematics, everybody has to get the same sort of thing, but I think with coding, depending on what your skill level is and what you're interested in, you can be different*.

Science investigations were viewed as slightly more open, with students focusing again on the perspective of a closed nature of demonstrating a particular process and solution. Two students commented in this regard: *Still with science; there is still sort of 'the' answer*, and

> *Well, if you're investigating something in Science, there are unknown things that you don't know and that is also the way with coding. Learning more about each coding thing–with coding, there are so many possibilities.*

5.1.2 | Theme 2: Availability of varied resources to discover multiple pathways to solutions

Students conveyed the degree to which they were able to access other resources to find answers to their questions. Didactic interaction with the teacher was not the students' preferred method for obtaining information or solving problems. Rather, students sought assistance in other ways. One student stated: *Different people have different things to offer too–for instance, (student name) helped a lot with the platform building because he's good with building things. He was also good when it came to solving problems as well.* This was reiterated by another student who stated: *You get the chance to ask other people and you learn from them as well.*

Students seemed to be able to identify which team members would be the 'go-to' people for particular areas of expertise, resulting in enhanced confidence and perseverance. One student remarked that two peers were particularly good at generating ideas: (*student name*) *and (student name) are like talented people and it is ok to combine all of their ideas*, while another identified a peer who was particularly good at the design aspect of the process: *When you're working with someone else, you're not rushing to get it finished—also got someone to rely on.*

Also evident was the impact of the social constructivist aspect of the 3C Model's (Martin, Curtis, Redmond, & Byrne, in press) link to increased motivation levels. One student who regularly asked to continue his work at lunchtime with groups of friends pointed out that the social problem-solving aspect was helpful:

> *I have to say that one of the reasons that I come at lunchtime is that we can work as a team to get things done. And while we're doing that, we can interact and have a good laugh. Because we're working in a team, we can make the thing more detailed, and we can learn from each other too.*

Students also commented that as another resource, the physical device provided immediate feedback to their coding. Students find debugging programs easier with physical devices. This was encapsulated in the statement:

> *It's so different from when you're in a classroom and you are kind of writing it down on paper and you don't care very much, but when you're coding and building robots with motors and sensors … when you're testing the program, you can kind of see what's going wrong.*

5.1.3 | Theme 3: Positive challenge and the reality of perseverance

One viewpoint communicated by the students was that the problems presented to them in the coding sessions were challenging but that they appreciated the challenges that this offered to their skill level. One student summarised this viewpoint stating, *I like the struggle part, the fact that it's hard.* Two student responses conveyed agreement but also expressed the sentiment that there is a fine balance in terms of their patience. *Sometimes it depends on how much—like if I do seem to struggle forever and ever, then it does get frustrating*, and, *You lose that drive to complete it. But it's also good for it to be challenging because if it's too easy, then you don't learn.*

There was a caveat to the notion of positive challenge. An unexpected aspect that emerged from this theme was the difficulty that physical computing created in terms of making a mistake with building materials. Such errors, because they involved cutting, gluing and joining, were perceived as difficult to fix, creating anxiety about rebuilding. Two students noted this: *Hands-on is a bit scary because you can stuff it up sometimes*, and, *You've got to restart, but with coding, you can just undo. So, we kind of felt a little bit more comfortable with Scratch.*

5.1.4 | Theme 4: The importance of visuals

The solutions students created in the form of physical and viewable artefacts are indicative of how the 3C Model's (Martin, Curtis, Redmond, & Byrne, in press) structure and intent allowed students to see the connection between their natural language and the coding concepts. For example, a student made a diagram of their group's solution showing how the 'train stop device' is enacted in code. The purpose of the train stop device is to prevent children from running onto the train tracks, which are protected by a hinged, steel wall erected on the platform's edge. A series of microprocessors control the hinged wall. The purpose of the microprocessors is to establish and then announce the train's arrival via a pressure sensor initially. After 3 s, the ramp is lowered, allowing people to board. During the focus group session, the student illustrated the process (Figure 4) while describing how the solution worked:

> *The train crosses over a button on the track as it approaches a station. It then waits three seconds before it triggers an audio message that warns passengers that a ramp is being lowered. The infrared sensor then waits until all passengers have boarded the train (the sensor is in wait mode until the platform is cleared.) The sensor then sends a message to the processor, which then sends a message to a speaker to play an audio file that says,*

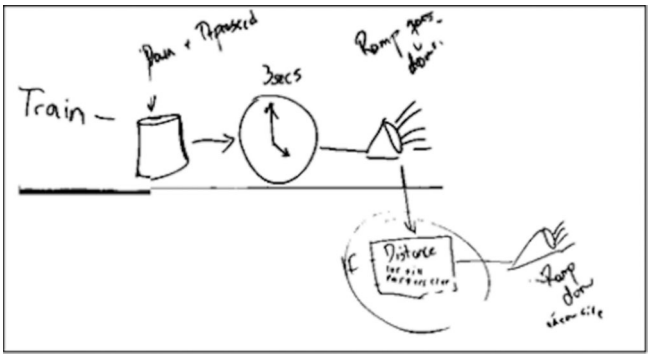

FIGURE 4 Student drawing showing how the train stop device works.

> *'Stand back'. Three seconds later, the ramp moves up, keeping people on the platform safe again and the train moves off.*

This type of visual representation of their thinking was often used when students became puzzled regarding the abstract element of the solution. One student stated, *So if we get stuck with the code problem … sometimes we can come and see you [teacher], but sometimes you can just have a go and then test it and see what it does. And kind of do things like moving around or drawing. So, you look at the robot, you look at the code and look at the drawing, then you look back at the robot.*

Alternatively, the students would revert to earlier learning stages of the 3C Model (Martin, Curtis, Redmond, & Byrne, in press) such as verbalising, motoric, or diagrammatic representations to problem-solve and formulate solutions. This approach was evident in the statement from a participant: *Another really good way is just to read the code… read it and see what it says. Try and say it. Of all the ways mentioned, (the) one that I would like the best is to try really hard to say it. It's what works best.*

5.1.5 | Theme 5: The connection between language and code

One aspect of student language use that was apparent in discussions with the students regarding their projects was how they, seemingly unconsciously, expressed their ideas using the formal language of code. Throughout the interviews, particularly when students were describing their solutions, they consistently used language that was reflective of the relevant coding concepts. For example, one student who created a scanning device using touch and light sensors to help blind people recognise products in stores required a series of coded IF … THEN statements. The touch sensor functioned as the trigger to start the light sensor, which recognised coloured dots on the side of the product (Figure 5). The student explains:

> *When you touch the touch sensor, it starts the colour sensor, and **if** the sensor saw a particular colour, **then** it was [acted] like a trigger to play an audio file—like if it's all orange, I programmed it to play the audio file; this is a can of beans, but **if** it saw blue, **then** would say 'this is a loaf of bread'.*

FIGURE 5 A sensor device was created to help blind people.

In the student's verbal description of the sensor device, the language he used reflects the intent of the Australian Curriculum: Digital Technologies, where learners are to "Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition) (ACTDIP019)" (ACARA, 2023a). Evidence of the intent being made explicit is when he used the IF … THEN statements. Another student demonstrated this level of understanding and skill with this WHEN … THEN statement: *When the train passes over the sensor, **then** the LED displays the message.*

5.2 | Researcher observations and reflections

The results of two sets of insider researcher observations along with reflections are presented, derived from two teams of students. The curriculum content for these two scenarios was sourced primarily from the Australian Year 5 and Year 6 Mathematics and Technologies curricula (ACARA, 2023b; 2023c).

5.2.1 | Classroom lesson observations # 1

The insider researcher provided the context for Lesson 1 from a local news story where a young person had been injured when a car reversed out of a parking space at a local shopping centre.

Lesson 2 involved distributing to the students Lego blocks which had pre-coded infrared devices attached to them. The insider researcher programmed the devices so that their screens displayed the distance in centimetres they were from another object. He did not explain the purpose of the programming to the students as they engaged in free play with the devices, strategically aiming to foster

discovery learning regarding its capabilities. He noted that initially the students interpreted that the numerical value displayed was related to the amount of light around the device. After further exploration, the students discovered that the device was measuring the distance in centimetres from the object at which it was directed, verified by the insider researcher as an instance of the students' knowing-in-action. By attentively listening, the researcher substantiated naturally emerging language from the students, which were noted as pre-cursor statements to the IF … THEN branching statement. For example, *If I point it at the floor, then the reading is about 60 cm*.

During Lesson 3 the insider researcher collaborated with the students to create a physical re-enactment of a car reversing out of a parking space. The lesson aimed to develop the students' computational thinking and for them to contemplate a digital solution that would automatically stop the car if it approached a person too closely while reversing. In teams of three, students used drama and movement to act out, articulate and record what was occurring moment-by-moment. One student took the role of the car reversing. Another student's role was to be the person the car was reversing towards. This student was pointing the programmed infrared sensor at the 'car'. Student 3 was tasked with writing down what was happening in everyday language (which was used in the next lesson to connect to the related block code). The insider researcher noted that as the car (role played by student 1) was reversing towards the infrared sensor (student 2), student 2 was calling out decreasing values, *the car is 60 cm from me, the car is 40 cm from me…* When it was estimated that the distance between them was 10 cm, student 2 said *Cut the gas and hit the brakes!* During the enacting, the students were heavily involved in problem-solving conversations which were pre-cursors to the debugging (computational thinking) process. For instance, the students realised that if the car was reversing too quickly, then although the sensor detected the object at 10 cm, the momentum of the vehicle was too great, and the device was ineffective for its intended purpose. The insider researcher associated this realisation by the students as an instance of reflection-in-action since they were able to conceptualise the programming solution by initially engaging with the scenario through physical movement. The students' approach to debugging this issue was to set different responses according to the decreasing distance the car was from the sensor. In everyday language the researcher noted the students saying, *When the car is 3 m away from the child, cut the power 30%. When the car is 2 m away from the child, cut the power by 70%. When the car is 1 m away from the child, cut the power by 100%.*

Consistent with the 3C Model's instructional sequence, during Lesson 4 the insider researcher illustrated side-by-side the everyday language used by the students from Lesson 3 (above) alongside the actual code from the software's interface printed on cardboard strips. Figure 6 illustrates an example of this instructional strategy.

As a reflection-on-action, the insider researcher established that alternatively he could have provided the sequence of code to the students and asked them to replicate and recreate the coded device; but they would have missed the conceptual understanding behind its functioning and would not have been involved in the computational thinking process. He corroborated the effectiveness of the 3C Model's instructional sequence by observing how it enabled students to be generative with the code and apply it to unfamiliar problems, as evidenced in Lesson 5.

In Lesson 5, the students applied their newly acquired knowledge while designing, developing and presenting various other digital solutions. For example, one team developed an alarm system that would sound an audio alert if an object approached too closely. Another team designed a device to assist a visually impaired person in exiting a room, playing an increasingly louder audio tone as the person moved closer to the exit. The insider researcher assessed the students' learning and artefacts as highly accomplished by observing: (1) the students' transfer of concepts like the IF … THEN conditions to their chosen problem, (2) articulating their explanations regarding the algorithms they developed in both every day and coded language, and (3) aligning with the aims of the Australian Technologies curriculum: students use computational thinking and digital systems to define, design, create, manage, implement and evaluate digital solutions to contextually authentic problems (ACARA, 2023b).

5.2.2 | Classroom lesson observations # 2

The insider researcher developed the context for Lesson 1 from discussions around safe practices associated with the ongoing COVID situation and the need for immunocompromised individuals and their carers to maintain a 1.5-m distance from each other. The initial problem-solving stemmed from the reality that many students at the school had parents working in hospitality. The determined need was a way to solve the 1.5-m spacing issue.

Lesson 2 involved students being given light sensors attached to Lego blocks, pre-coded by the insider researcher to follow a pre-drawn black line imprinted on the table where meals are served. While the students were not introduced to the code at this stage, they were allowed free play with the pre-coded devices. The insider researcher observed that the students first noticed a jerky movement of the Lego block as it sensed and then lost the line, moved forward slightly and then looked for the line again. Students noticed that if they took the Lego block away from the line it simply circled until it found the line again. The insider researcher noticed emerging language around the IF … THEN branching statement. For example, one student stated, *IF the robot can't find the line, THEN it keeps circling* and *IF it is on the line, it moves forward a little bit with one wheel and THEN looks again. IF it can't see the line, THEN the other wheel turns on for a little bit*. The researcher determined that the student's use of language demonstrated a precursor understanding of the conceptual idea of looping represented in an everyday language such as *The robot just keeps doing the same thing, over and over again*.

Next, an additional sensor was added which was programmed to respond to a selected line colour which triggered a turn off the line and play an audio file. The insider researcher noted that the students were quick to detect, and articulate in everyday language, that the robot would continue until a particular condition was identified.

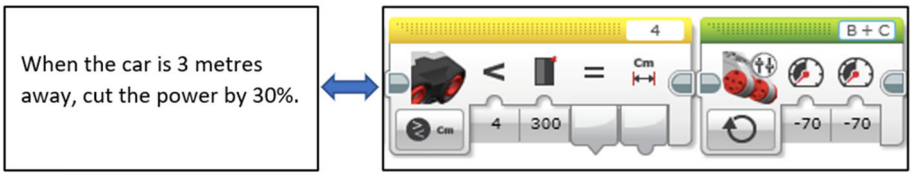

FIGURE 6 Example of everyday language used by students, alongside the related code.

Statements from the students included the keyword UNTIL which denotes exiting a loop. For example, *The robot will keep following the line UNTIL it sees the colour red*.

During Lesson 3, the insider researcher outlined simple mazes on the floor using masking tape. The interior walls of the maze were made of black lines and coloured squares indicated the corners. Next, the insider researcher pretended to be the robot, and articulated some of the words that the students were using in the line-following exercise from the previous lesson. The insider researcher then simulated the second sensor by shining a torch on the ground using an extended right arm. Then he articulated the exit from the loop using the UNTIL phrase after each physical step. *I will keep going UNTIL I see a colour*. When that torch lit up a colour square, the insider researcher again voiced the sentence *I can see a colour, so I will turn right*. Students were then broken into small groups and assigned their own mazes where they engaged in the same behaviours. The insider researcher noted that the students developed similar algorithms using everyday language. He categorised this use of language as an example of reflection-in-action.

In Lesson 4, the insider researcher printed the students' language representations from Lesson 3 on strips of paper and presented them alongside the actual code printed on cardboard strips (Figure 7).

This instructional step was repeated, providing students with various connections between the actions, the words and the symbolic code. Subsequently, the insider researcher observed that the students were able to build and articulate the symbolic code independently and with a higher degree of complexity. As a reflection-on-action, the insider research proposes that his work during this critical pedagogical step enabled students to conceptually work out (reflection-on-action) the programming solution by initially working through the scenario motorically. This critical pedagogical step within the 3C Model's sequence of learning enabled the students to apply their acquired knowledge of the code to new and more complex contexts during their summative assessments.

In the final lesson, the students progressed to further develop and innovate on this code as part of their practical assessments. One team designed and created a factory floor system for delivering parts using specific code, while another team designed and created a ride for a children's fun fair. The insider researcher assessed these designed and created artefacts as a high standard evidenced by: (1) the student's transference of concepts such as looping and iterations to their chosen problem through the application of their acquired knowledge, (2) the ability to articulate the meaning of the symbolic block code in both written and coding language, and (3) aligning with the Digital Technologies achievement standards specified in the curriculum (ACARA, 2023a).

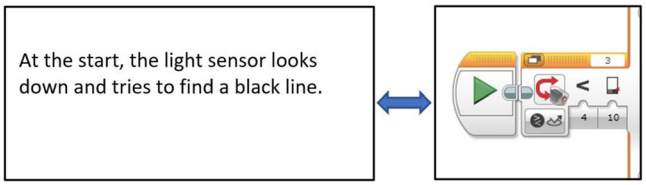

FIGURE 7 Example of everyday language used by the students, alongside the actual code.

6 | DISCUSSION

The study engaged 11 upper primary school students in two STEM activities using the 3C Model (Martin, Curtis, Redmond, & Byrne, in press) to answer the research question, *What artefacts, language representations, and student perceptions developed from learning coding and computational thinking using the 3C activity sequencing model?* Through capturing the students' experiences during the focus group sessions, the researchers gained insights into the efficacy of the 3C Model's instructional approach.

Student responses which represent theme 1 revealed they recognised the opportunity to develop multiple pathways to their solutions creatively. For example, *You have to find out something, but [in Mathematics] there is still 'the answer'. In coding, you are working on different things and you have to find a way to make your coding work*. This indicative response supports the aim of the Australian Digital Technologies curriculum of developing in students the ability to use computational thinking skills creatively when formulating digital solutions to authentic problems (ACARA, 2023a).

Student responses also indicated engagement with varied resources, such as their peers, to share ideas and discover ways to solve the problems inherent in a socially constructed learning environment. One student noted: *I like working together so much more because you kind of exchange ideas—rather than forming something on my own and then (student name) forming something on her own. If it is together, it is a lot better*. The collective meaning from the student responses in theme 2 is consistent with the assertions of Bers et al. (2019), indicating that knowledge constructed socially, such as sharing applications, is a critical part of the learning process.

Students' comments representing theme 3 conveyed having positive experiences with their authentic challenges and the reality of perseverance. One student stated, *The stuff, it's a lot harder but this is more friendly. But with coding, you can just keep working and it's really exciting when you can solve that problem*. Another student stated, *I think getting into trouble with your work helps you … I think it expands your knowledge*. The viewpoints shared by the students are supported

by Sirakaya et al. (2020), who reported a clear predictor of positive attitudes in STEM was linked to student preference for authentic learning and that allowing students to choose in this respect led to a significant positive effect on computational thinking skills.

The fourth and fifth themes relate to Piaget's (1964) levels of cognitive development and the Language Model (Irons, 2014; Irons & Irons, 1989) in that there is a progression from the concrete operational stage (acting out the computational thinking using movement and concrete materials and visual representations) to the abstract (in the form of code). When asked about the importance of visuals, a student conveyed,

> I think this way is good because you can ask questions and you can use different ways like drawing it to show you what you actually can do. If you are in a book, then you don't get to visually see what it does. In a book you have to think, you can't see it.

For the last theme, code is apparent in the language, a student provided the following: *So I made a tea dipper—if you dip your teabag ten times, you get a really strong cup of tea? But if you only dip your teabag three times, then you get a really weak one. So you can program it to do like 10 dips, and then it does 10 dips, and so on (laughs).*

As one of the researchers delivered the intervention, his observations along with reflections provide further insight into the efficacy of the 3C Model's (Martin, Curtis, Redmond, & Byrne, in press) instructional approach. As a reflective practitioner, he needed to become aware of and question his 'tacit' knowledge (Schön, 1983), exploring "events and assertions in relation to other experiences" (p. 49), which can lead "to new meanings and forms of practice" (DeLuca et al., 2023, p. 6). Teachers engage in this reflective practice continuously and in a deliberate way to uncover their underlying thoughts, beliefs, and biases.

Support of the 3C Model's (Martin, Curtis, Redmond, & Byrne, in press) instructional sequence in producing these types of student learning experiences lies in its theoretical and pedagogical alignment with Piaget's (1964) theory of cognitive development and the pedagogy underpinning the Language Model (Irons, 2014; Irons & Irons, 1989). Piaget (1964) highlighted the notion that progression from concrete to abstract thinking is a fundamental aspect of cognitive development. The Language Model provides similar learning experiences in the same way multi-arithmetic blocks allow students to grasp and represent concepts of place value and then switch to symbolic numerals. Physical devices, movement, language and simple representation also aid in developing the concepts and skills of branching and creating algorithms.

7 | IMPLICATIONS, LIMITATIONS AND FURTHER RESEARCH

To date there is no widely accepted pedagogical approach for teaching coding and computational thinking to primary school students according to their developmental level. This study fills the existing gap by investigating the efficacy of the 3C Model (Martin, Curtis, Redmond, & Byrne, in press). Key considerations in using the 3C Model are ensuring that conceptual development occurs using physical digital computing devices and language students are familiar with prior to introducing coding language and its symbolic representations.

The findings from this research have various implications. First, the lens of this research has focused on developing computational thinking and coding skills from the student's point of view. From a teacher's perspective, a practical implication of using the 3C Model (Martin, Curtis, Redmond, & Byrne, in press) is that it applies pedagogical thinking to the planning and teaching process. The 3C Model represents a transfer of the successful and familiar pedagogy for teaching mathematics to the teaching of coding and computational thinking. Using body movement provides students with another way to connect with the complex concepts involved in coding and computational thinking. It provides a process for teaching coding from a conceptual background, ensuring a deeper understanding and opportunities to be creative rather than the learning being procedural and confined to isolated skills. Using common language developed through the learning stages ensures that the terminology is specific to the learning context. The verbal and modelled support is gradually withdrawn as students learn to understand the relationship between the elements of movement, language, the intent of the code and the computational focus. In line with the Language Model (Irons, 2014; Irons & Irons, 1989), once conceptual ideas are grasped, coding becomes easily achievable. Students can then generalise and apply the coding concepts to a context of their choosing. Conversely, when instruction is largely explicit, teaching can become decontextualised and learning tends to be procedural and surface-based, similar to how learning an algorithm as an isolated skill in mathematics can stifle creativity and promote the fracturing of learning into unrelated segments (Sentance et al., 2017). Further research is needed to address the process of inculcating teachers into this instructional sequencing model and exploration of their associated connection with the teaching of mathematics. Hence, the next logical step in the research is to examine the three stages of the 3C Model when embedded into a teaching sequence from the teacher's perspective.

Second, the generalisability of the study's results is limited because the data were collected from a small sample of regional Catholic primary school students. Also, the participants were Year 5 and Year 6 students involved in a coding club. Future research could expand the study to students in other year levels and from general classroom teaching. Also, the 3C Model (Martin, Curtis, Redmond, & Byrne, in press) has yet to be statistically validated, which is an avenue for future research.

The 3C Model links contextual issues, device capabilities, physical movements and staged language development with the abstract concepts and skills of coding and computational thinking. This connection is based on Piaget's (1964) learning theory and the Language Model for teaching mathematics (Irons, 2014; Irons & Irons, 1989). A practical implication of the 3C Model is its provision of a coherent and recognisable pedagogical approach that enables teachers to effectively integrate coding concepts. This supports the development of

computational and algorithmic thinking, aligning with the objectives of the Digital Technologies curriculum (ACARA, 2023a).

AUTHOR CONTRIBUTIONS

David A. Martin: Methodology; formal analysis; conceptualization; data curation; writing – original draft; writing – review and editing. **Peter Curtis:** Investigation; formal analysis; conceptualization; data curation; writing – original draft; writing – review and editing. **Petrea Redmond:** Formal analysis; methodology; writing – original draft; writing – review and editing.

ACKNOWLEDGMENT

Open access publishing facilitated by Edith Cowan University, as part of the Wiley - Edith Cowan University agreement via the Council of Australian University Librarians.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ETHICS STATEMENT

This research was approved by the University of Southern Queensland's Human Research Ethics Committee (H17REA179) and conducted in accordance with all required ethics protocols.

ORCID

David A. Martin https://orcid.org/0000-0002-4094-591X
Peter Curtis https://orcid.org/0000-0002-8737-5205
Petrea Redmond https://orcid.org/0000-0001-9674-1206

REFERENCES

Australian Catholic University: Institute for Learning Sciences & Teacher Education. (2023). Coding animated narratives. https://codingstories.com.au/

Australian Curriculum, Assessment and Reporting Authority (ACARA). (2023a). Version 9 Australian curriculum: Digital technologies. https://v9.australiancurriculum.edu.au/teacher-resources/understand-this-learning-area/technologies#digital-technologies

Australian Curriculum, Assessment and Reporting Authority (ACARA). (2023b). Version 9 Australian curriculum: technologies. https://v9.australiancurriculum.edu.au/teacher-resources/understand-this-learning-area/technologies#technologies

Australian Curriculum, Assessment and Reporting Authority (ACARA). (2023c). Version 9 Australian curriculum: Mathematics. https://v9.australiancurriculum.edu.au/f-10-curriculum/learning-areas/mathematics/year-5

Australian Education Council. (1991). Enhancing mathematics learning. In *A national statement on mathematics for Australian schools* (pp. 16–24). Curriculum Corporation.

Bers, M. U., González-González, C., & Armas-Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education*, *138*, 130–145. https://doi.org/10.1016/j.compedu.2019.04.013

Bjursten, E. L., Nilsson, T., & Gumaelius, L. (2023). Computer programming in primary schools: Swedish technology teachers' pedagogical strategies. *International Journal of Technology and Design Education*, *33*(4), 1345–1368. https://doi.org/10.1007/s10798-022-09786-7

Carrington, S. B., & Selva, G. (2010). Critical social theory and transformative learning: Evidence in pre-service teachers' service-learning reflection logs. *Higher Education Research and Development*, *29*(1), 45–57. https://doi.org/10.1080/07294360903421384

Clarke, V., & Braun, V. (2014). Thematic analysis. In *Encyclopedia of critical psychology* (pp. 1947–1952). Springer.

Dag, F., Şumuer, E., & Durdu, L. (2023). The effect of an unplugged coding course on primary school students' improvement in their computational thinking skills. *Journal of Computer Assisted Learning*, *39*(6), 1902–1918. https://doi.org/10.1111/jcal.12850

DeLuca, C., Willis, J., Dorji, K., & Sherman, A. (2023). Cultivating reflective teachers: Challenging power and promoting pedagogy of self-assessment in Australian, Bhutanese, and Canadian teacher education programs. *Power and Education*, *15*(1), 5–22. https://doi.org/10.1177/17577438221108240

Dewey, J. (1910). *How we think*. DC Heath & Co.

Dewey, J. (1933). *How we think: A restatement of the relation of reflective thinking to the educative process*. DC Heath & Co.

Irons, C., & Irons, R. (1989). Language experiences: A base for problem solving. In P. Trafton & A. Shulte (Eds.), *Handbook: New directions for elementary school mathematics* (pp. 85–98). National Council of Teachers of Mathematics.

Irons, R. (2014). Language is the core for the concept of addition. *Educating Young Children: Learning and Teaching in the Early Childhood Years*, *20*(1), 38–41.

Kastner-Hauler, O., Tengler, K., Sabitzer, B., & Lavicza, Z. (2022). Combined effects of block-based programming and physical computing on primary students' computational thinking skills. *Frontiers in Psychology*, *13*, 875382. https://doi.org/10.3389/fpsyg.2022.875382

Levi-Strauss, C. (1966). *The savage mind*. University of Chicago Press.

Martin, D. A., Curtis, P., Redmond, P., & Byrne, M. (in press). The 3C model for teaching coding and computational thinking with an M in STEM focus. In E. Langran (Ed.), *Society for Information Technology & Teacher Education International Conference*. Association for the Advancement of Computing in Education (AACE).

Mason, S. L., & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education*, *19*(4), 790–824.

Miller, J. (2019). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns. *ZDM Mathematics Education*, *51*(6), 915–927. https://doi.org/10.1007/s11858-019-01096-y

Mills, K., Coenraad, M., Ruiz, P., & Burke, Q. (2021). *Computational thinking for an inclusive world: A resource for educators to learn and lead*. Digital Promise. https://doi.org/10.51388/20.500.12265/138

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

Payne, J. N., & Rathmell, E. C. (1975). Mathematics learning in early childhood: Number and numeration. *National Council of Teachers of Mathematics Yearbook*, *37*, 125–160.

Piaget, J. (1964). Part I: Cognitive development in children: Piaget development and learning. *Journal of Research in Science Teaching*, *2*(3), 176–186. https://doi.org/10.1002/tea.3660020306

Przybylla, M., & Romeike, R. (2015). KEYCIT 2014: Key competencies in informatics and ICT. In T. Brinda, N. Reynolds, R. Romeike, & A. Schwill (Eds.), *Proceedings of the IFIP TC3 Conference "Key Competences in Informatics and ICT (KEYCIT 2014)"* (Vol. 7, pp. 351–361). Universitätsverlag Potsdam.

Przybylla, M., & Romeike, R. (2017). The nature of physical computing in schools: Findings from three years of practical experience. In C. S. Montero & M. Joy (Eds.), *Proceedings of the 17th Koli calling*

international conference on computing education research (pp. 98–107). The Association for Computing Machinery. https://doi.org/10.1145/3141880.3141889

Rich, P. J., Bartholomew, S., Daniel, D., Dinsmoor, K., Nielsen, M., Reynolds, C., Swanson, M., Winward, E., & Yauney, J. (2022). Trends in tools used to teach computational thinking through elementary coding. *Journal of Research on Technology in Education*, 1–22. https://doi.org/10.1080/15391523.2022.2121345

Schön, D. (1983). *The reflective practitioner*. Basic Books.

Schön, D. (1987). *Preparing professionals for the demands of practice in his educating the reflective practitioner*. Jossey-Bass.

Sentance, S., Waite, J., Yeomans, L., & MacLeod, E. (2017). Teaching with physical computing devices: The BBC micro: bit initiative. In E. Barendsen & P. Hubwieser (Eds.), *Proceedings of the 12th workshop on primary and secondary computing education* (pp. 87–96). The Association for Computing Machinery. https://doi.org/10.1145/3137065.3137083

Shah, M. A. (2022). Teachers as reflective practitioners: From individualism to Vygotskian social constructivism. *Alberta Journal of Educational Research*, *68*(3), 297–307. https://doi.org/10.11575/ajer.v68i3.68598

Sirakaya, M., Alsancak Sırakaya, D., & Korkmaz, Ö. (2020). The impact of STEM attitude and thinking style on computational thinking determined via structural equation modeling. *Journal of Science Education and Technology*, *29*, 561–572. https://doi.org/10.1007/s10956-020-09836-6

Statti, A., & Torres, K. (2022). Best practices of teaching coding skills in elementary education. In E. Langran (Ed.), *Society for information technology & teacher education international conference* (pp. 2102–2107). Association for the Advancement of Computing in Education (AACE).

Sun, L., Hu, L., Yang, W., Zhou, D., & Wang, X. (2021). STEM learning attitude predicts computational thinking skills among primary school students. *Journal of Computer Assisted Learning*, *37*(2), 346–358. https://doi.org/10.1111/jcal.12493

Williams, H. (2021). *No fear coding: Computational thinking across the K-5 curriculum* (2nd ed.). International Society for Technology in Education.

Wing, J. (2009). Computational thinking. *Journal of Computing Sciences in Colleges*, *24*(6), 6–7. https://doi.org/10.1145/1118178.1118215

Woo, K., & Falloon, G. (2023). Coding across the curriculum: Challenges for non-specialist teachers. In *Teaching coding in K-12 schools: Research and application* (pp. 245–261). Springer International Publishing.

Zapata-Cáceres, M., Martín-Barroso, E., & Román-González, M. (2021). BCTt: Beginners computational thinking test. In *Proceedings of the Raspberry Pi Foundation Research seminar series* (pp. 46–56). Raspberry Pi Foundation.

Zeng, Y., Yang, W., & Bautista, A. (2023). Teaching programming and computational thinking in early childhood education: A case study of content knowledge and pedagogical knowledge. *Frontiers in Psychology*, *14*, 1342297. https://doi.org/10.3389/fpsyg.2023.1252718

How to cite this article: Martin, D. A., Curtis, P., & Redmond, P. (2024). Primary school students' perceptions and developed artefacts and language from learning coding and computational thinking using the 3C model. *Journal of Computer Assisted Learning*, *40*(4), 1616–1631. https://doi.org/10.1111/jcal.12972