Australian Journal of Teacher Education

Volume 44 | Issue 4 Article 4

2019

The Next Chapter in the STEM Education Narrative: Using Robotics to Support Programming and Coding.

SUSAN BLACKLEY
Curtin University, susan.blackley@curtin.edu.au

Jennifer Howell
Jennifer.Howell@curtin.edu.au

The Next Chapter in the STEM Education Narrative: Using Robotics to Support Programming and Coding

Susan Blackley
Jennifer Howell
Curtin University

Abstract: In this paper, we use our qualitative research notes and observations to portray a model for integrated STEM education and summarise primary school students' typical and recurring ways in which they engaged with each new robot. The purpose of this paper is two-fold: first, to unpack key elements of the Australian Curriculum: Technologies in order to support teachers and pre- service teachers to implement these components, and second, to describe ways in which teachers can teach authentic integrated STEM education that also provides opportunities for students to develop and demonstrate 21st century competencies. Based on data collected from projects undertaken in a number of school sites over 18 months, we have developed and share a model for the gradual structured release of teacher control over student activity in STEM activities, and describe how this concept can be a basis for in-situ teacher professional learning. The affordances of robotics and visual programming as a context for integrated STEM education are discussed, and identified as promoting "head-heart-hands" learning.

Key words: STEM education, visual programming, robotics

Problem

Institutions that provide Initial Teacher Education (ITE) programs are faced with Australian Institute for Teaching and School Leadership (AITSL) accreditation imperatives, and of particular interest to this position paper is Program Standard 4:

> *In addition to study in each of the learning areas of the primary school curriculum sufficient to equip teachers to teach across the years of primary schooling, programs provide all primary graduates with a subject specialisation through:*
>
> *a) clearly defined pathways into and/or within a program that lead to specialisations, that are in demand, with a focus on subject/curriculum areas*
> *b) assessment within the program requiring graduates to demonstrate expert content knowledge and pedagogical content knowledge and highly effective classroom teaching in their area of specialisation*
> *c) publishing the specialisations available, and numbers of graduates per specialisation through their annual reports. (AITSL, 2015, p. 14)*

Many institutions that offer ITE programs in primary education (such as Curtin University, Queensland University of Technology, RMIT, and La Trobe University) have

targeted Science, Technology, Engineering, Mathematics (STEM) or STEAM (incorporating the Arts) education as a "specialisation" option in response to Commonwealth and State government financial and resource commitments to STEM education. Further evidence of this focus area, the University of Canberra has begun offering a Bachelor of Education, Primary STEM, delivered by the Faculty of Education, Science, Technology, and Mathematics.

The development and accreditation of these specialised units is time-consuming and costly, and requires commitment and consensus from faculty leadership and teaching staff. This in itself may be problematic for the following reasons: (1) lack of consensus about what constitutes STEM education, (2) the potential impact of the Technologies Curriculum upon ITE programs, and (3) the competence and confidence of in-service teachers to mentor pre-service teachers whilst on professional placement in a STEM specialisation, given their possible lack of training in this field (Blackley & Howell, 2015).

This paper unpacks the *Australian Curriculum: Technologies*, (ACARA, 2015) developed and released in Phase 3 of the Australian Curriculum rollout, was developed to provide some guidance as to how pre-service and in-service teachers can authentically incorporate the content descriptions into their practice, within a context of integrated STEM education. As such, this unpacking may serve as a useful starting point for STEM educators. Following this, we focus in this paper, not so much a report on our research, rather, how our work with programmable robotics in primary schools has informed our concept of integrated STEM education, and in particular how the T for technology can be robustly enacted. From semester 1, 2016 to the end of semester 2, 2017, we worked with three Western Australian schools – two metropolitan and one regional – with seven teachers and four Year 4 classes, two Year 5/6 classes and one composite class of Years 6 to 8 (students, $N = 198$). The primary qualitative data sources were student worksheets, which were completed each week and pertained to a particular aspect of the week's robot, teacher interviews, and the researchers' observations and field notes. In this paper, we use our observations and notes to portray a model for integrated STEM education and summarise the students' typical and recurring ways in which they engaged with each new robot.

As suggested by Ntemngwa and Oliver (2018) there is a need for "documentation with emphasis on the nature of the integration process, how teacher scaffold the instruction and the outcomes of the integrated STEM instruction on student and teachers are particularly necessary" (p. 12), and this paper addresses this need.

Background

The release and implementation of the *Australian Curriculum: Technologies* has already impacted teachers, particularly in primary schools, as they access digital tools and professional learning events with the goal of successfully teaching and assessing this additional curriculum area in an already over-crowded space. Further to this, Initial Teacher Education (ITE) programs have been impacted by new accreditation imperatives resulting in pre-services teachers having to nominate a specialisation in their degree, two of which are science and mathematics, although many Australian ITE programs have chosen to provide a specialisation in STEM education.

The implementation of the *Australian Curriculum (AC): Technologies* for students in Foundation to Year 10 (5 to 15 years of age) is one of a raft of curricula that have been developed globally to reignite engagement with what was called "Computer Science" in the 1960s. The Computer Science Teachers Association (CSTA) of the United States defined "Computer Science" as the study of computers and algorithmic processes, including

hardware, software, and programming (Heitlin, 2014), and Wilson, Sudol, Stephenson, and Stehlik (2010, p. 24) referred to Computer Science as "an academic discipline that encompasses the study of computers and algorithmic process, including their principles, their hardware and software designs, their applications and their impact on society". However, Computer Science *Education* includes: the creation of digital artefacts, computational thinking, algorithm development and implementation, programming (character and graphical user interfaces), networks, graphics, databases and information retrieval, information security and privacy, artificial intelligence, applications in information technology and systems, and the social impacts of computing (Wilson et al., 2010). Whilst a resurgence in Computer Science has been occurring this decade in countries such as the United States, New Zealand, England, Wales, Scotland, Greece, Israel, Germany, India, and South Korea (Jones, 2011), the depth and breadth of study, as outlined in Computer Science Education, does not seem to be addressed other than in specialist senior secondary school subjects.

As indicated in the *AC: Technologies*, and in many of the individual state curricula throughout the United States, computer *programming* is also making a comeback across K-12 levels of schooling. In the mid-1990s, schools tended to relegate programming to the "too hard" or "for what purpose" buckets: access to computer labs and qualified teachers was problematic, and why deal with exacting coding languages with syntactical challenges when access to pre-assembled multimedia packages via CD-ROMs was easy (Kafai & Burke, 2013). Over the last decade, the affordances and accessibility of mobile smart devices (e.g., tablets, iPads, and iPhones) and improved Internet bandwidth have supported a move to engage with programming that strongly reflects the organic and dynamic way in which 21st century learners utilise their technology of choice in their everyday life. The primary uses of Internet-connected personal mobile devices have been to connect on social media sites and to instantaneously access information. Increasingly, the scope of application is widening to incorporate the *entrepreneurial aspirations* of users to create websites and apps, and to upload clips to win fame and make money. In this paper, we suggest that the next chapter in the STEM narrative (Blackley & Howell, 2015) should be a deeper investigation into the T in STEM education, in particular the inclusion of programming and coding, and ways in which the T can be integrated with science, engineering and mathematics in authentic ways in the classroom.

The continued preoccupation with nationwide (e.g., National Assessment Program – Literacy and Numeracy (NAPLAN)), international, and high-stakes testing (e.g. Programme for International Student Assessment (PISA), Trends in International Mathematics and Science Study (TIMSS), and Progress in International Reading Literacy Study (PIRLS)) across a limited selection of curriculum (mathematics, science and literacy) has not only narrowed the taught curriculum in schools, but has also consolidated and elevated the status of didactic and implicit pedagogies in schools, and in so doing, has effectively eliminated engagement with the computer programming of the 70s and 80s (Pinkston, 2015). So why is there now a push to re-engage with computer programming and coding from the early years of schooling onwards? Clearly, the aim is not to produce tens of thousands of prospective professional coders and programmers; rather, the benefits of the cognitive demands of programming and coding have been researched (e.g., Eisenberg & Johnson, 1996), and it is evident that students who have had experience with programming and coding have superior problem-solving and higher-order thinking skills. Despite having computers in schools for the last 30 years, there has been a paucity of attention given to the explicit engagement in *computational thinking* across all phases of schooling; teaching word processing, spreadsheet creation, and pervasive PowerPoint presentations does little to strengthen student engagement in the deeper analysis needed to creatively, systematically and critically

engage with, and potentially solve, a wide range of problems (Collins & Halverson, 2009; Kafai & Burke, 2013).

Wing, in 2006, was one of the first researchers to define computational thinking as "solving problems, designing systems and understanding human behaviour, using the fundamental concepts of computer science" (Wing, 2006, p. 33). However, in her role as United States President's Professor of Computer Science and head of the Computer Science Department at Carnegie Mellon University, she explained that it is so much more than that. Wing (2006, p. 33) is expansive in her contestation that "computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability". Computational thinking is about "using abstraction and decomposition when attacking a large complex task or designing a large complex system" (Wing, 2006, p.33), and celebrating the ways in which humans think and create whilst utilising the functionality of computers to deal with huge data sets, representations and models, and complex calculations. In other words, humans do the thinking and the technology carries out the algorithms.

Along with the development of computational thinking as a process, is a specialised lexicon with which educators and students need to become familiar in order to share and interpret the thinking. This specialised register (referred to by some as "jargon") including terms such as *backtracking, prefetching, caching, preconditions,* and *algorithms,* can be daunting to the uninitiated. We contend that in school settings, the correct terminology should be modelled by the teacher and encouraged of the students when the process is being initially explained and demonstrated – in the same way that mathematics terminology should be introduced (e.g., use "equation" from the outset, rather than "number sentence"). In addition, the use of student- relevant examples would contribute to facility with the use of the terminology and understanding of the processes. For example, have the students bring their school bags into the classroom and turn out the contents. What they have packed for the day is a real example of both *prefetching* (deciding what to pack) and *caching* (packing the items in some order). Naturally there would be differences in the contents, and the teacher could segue to a discussion about variability and notions of *preconditions* (predicted activities and needed items are considered), and also highlight similarities in and differences between each student's thinking.

Computer programming is an aspect of computational thinking and globally has become a focus of many school curricula over the last 10 years, and it strongly reflects Computer Science Education in that it is a way in which students can develop and demonstrate computational competences (Grover & Pea, 2013), higher-order thinking skills, and algorithmic problem-solving skills (Fessakis, Gouli, & Mavroudi, 2013; Kafai & Burke, 2014). There are two basic types of computer programming that are readily accessible for school students: *Character User Interfaces (CUIs)* and *Graphical User Interfaces (GUIs)* (Pinkston, 2015). With command line (CUI) editing programs, students enter computer language or *codes* (e.g., HTML or Java Script) onto a command line, thus allowing the student to become a creator rather than merely a user of technology. However, this kind of coding can take time to master, and the accompanying frustrations of syntax error may have a detrimental effect on students who are not willing or able to persevere. Perhaps this kind of computer programming has a place in specialist senior secondary school subjects for very keen students to engage with and master as part of their career projection.

The second and more accessible group, GUIs, take advantage of the affordances of touch screens with *tiles* or objects to "drag and drop", such as SCRATCH and SCRATCH Jr (Junior) developed by the Massachusetts Institute of Technology (MIT) Media Lab's *Lifelong Kindergarten* group, and allows users to intuitively program and to join an online

community of users. We contend that the *visual programming* associated with GUIs is most appropriate for students in the first year of formal education through to the middle years (i.e., ages 5 – 14), and that visual programming also has the potential to appeal to and develop The Arts with students through choices of images and sounds that embellish their digital work. This also allows personalisation of projects that can support identity development and positive self-efficacy. With this in mind, we explore the curriculum in Australia, focusing on the content and progression of programming and coding.

The Australian Context

The new *Australian Curriculum: Technologies* (Foundation -Year 10) has two subjects within it: *Design and Technologies* and *Digital Technologies* (ACARA, 2015), and the stated rationale for its inclusion in what is already a crowded curriculum includes the view that digital systems "support new ways of collaborating and communicating, and as such require new skills such as computational and systems thinking" (ACARA, 2015). Despite the richness of what is essentially a very ambitious, albeit commendable, curriculum, the focus seems to rest on "programming and coding (PAC)". This is reflected by the Federal Government's commitment of $3.5m on the 'Coding across the Curriculum' initiative in a bid to incorporate coding into existing subjects. Further, in 2016 the first round of the Federal Government Digital Literacy School Grants was awarded to 54 applications with a total funding amount of $1,989,312.

Perhaps the focus on programming and coding is because other aspects of *AC: Technologies*, such as graphical representations and data analysis, are reassuringly familiar in the context of the mathematics curriculum, and as such do not warrant extraordinary attention.

The Progression of Programming and Coding

In the *AC: Technologies*, the Foundation to Year 2 content description related to PAC states that students will "follow, describe and represent a sequence of steps and decisions (algorithms) needed to solve simple problems" (ACTDIP004). When the elaborations are accessed on the website, it becomes evident that four of the five reflect activities that are generally already undertaken in F-2 classes: nominally "activity sequences" that are incorporated into both the mathematics and English curriculum. For example, recounting a typical school day in chronological order. The fifth elaboration (situated in first position) is the "new" focus - engaging with programmable devices in order to generate a specified movement or series of movements. Whilst the elaboration states that the provision of instructions could be made to "physical or virtual objects or robotic devices" the key to the content description is the *purpose* of this programming, namely to solve simple problems. Without this purpose, we believe that the exercise of programming becomes somewhat trivial, in the same way the learning multiplication facts is not an end in itself. Further, questions of equity arise when consideration is given to classroom resourcing and access to the virtual and robotic devices: cost, location, bandwidth, and student home experience all come into play.

In the next band, Year 3 and 4, there are two content descriptions related to PAC: *Define simple problems, and describe and follow a sequence of steps and decisions (algorithms) needed to solve them* (ACTDIP010) and *Implement simple digital solutions as visual programs with algorithms involving branching (decisions) and user input*

(ACTDIP011). The elaborations for the second content description are considerably more removed from other curriculum content, in particular the use of a visual programming language, creating flowcharts, and implementing programs that make decisions on the basis of user interaction (input or choice) involving branching. The implications of enacting these elaborations for teachers could potentially be very stressful as it is unlikely that their initial teacher education programs or their ensuing professional learning covered these areas, and even if their personal experience is such that they can do these tasks themselves, their *digital pedagogy* knowledge or self-efficacy to teach this to their students may be lacking.

The last band situated in the primary school years, Year 5 and 6, incorporates designing and producing user interfaces and the repetition of a process or set of instructions in programming (iterations) as an extension to designing and following simple algorithms. For Years 7 and 8, in the last band that is mandated for all students to engage with, students are required to *Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language* (ACTID030), which in essence is "coding" as a distinct activity from programming. Situating coding in secondary schooling is perhaps not as problematic as programming is in primary school, as there would generally be specialist digital technology (or computer science) teachers who could either teach this curriculum or mentor other teachers to do so.

Digital Technologies and Integrated STEM Education

In 2001, Judith Ramaley, the director of the United States' National Science Foundation's education and human resources division, was working to develop curriculum that would enhance education in science, technology, engineering and mathematics, and coined the term "STEM" (Teaching Institute for Excellence in STEM, 2010). Following this, Sanders (2009) was the first to promote the concept of *integrated STEM education,* and he described a pedagogical approach of "purposeful design and inquiry" (Sanders, 2009, p. 21). In this paper, by "integrated" STEM education we refer to the intentional engagement with products or solving real world problems that requires utilising two or more of the STEM disciplines, with or without other discipline areas such as the Arts, in tandem with 21st century competencies – *adaptability, communication, social skills (collaboration), creativity, non-routine problem solving, self- management, self-development* and *systems thinking* (Bellanca & Brandt, 2010).

How can teachers incorporate the new *AC: Technologies* curriculum and its focus areas effectively into a crowded curriculum program? Our response to this question is integrated STEM education. We believe that the incorporation of the T in STEM being sourced from the *AC: Technologies* presents a great opportunity for improved integrated STEM education; particularly as there already is a synergy between some of the content and the mathematics curriculum in the areas of data collection, management and representations. We posit that the key to successful integrated STEM education lies in the tools that teachers are able to access and confidently engage with in their classrooms, and the nature of professional learning opportunities and support to which they are privy. Ideally these tools would: provide opportunities for students to learn by doing; exemplify some of the key mechanisms that revolutionised human labour and production; link design and construction to visual programming for the operation of the devices; and support students to develop and demonstrate the 21st century competencies. In this way, students would not merely be consumers of digital technologies; they would be producers of digital artefacts. Are such tools readily available to and affordable for schools or are they merely elusive and aspirational? This paper describes how one such tool, LEGO *WeDo,* can foster integrated

STEM education in primary schools, including visual programming and 21st century competencies, thus providing potentially rich and authentic learning experiences.

We began using *WeDo* in 2010 when it was relatively new on the scene in Australia, as a tool to develop integrated STEM education, and to foster literacy and numeracy development in a culturally diverse class of Prep/Year 1 students (ages 5 to 6). Six years later, with the resurgence of interest in STEM education, we conducted professional learning for in-service teachers at the Curtin University, School of Education Professional Learning Hub in 2016, to engage in-service teachers in using *WeDo*, and in so doing, make explicit related digital pedagogies and overt links to integrated STEM education and the Western Australian Curriculum. Robotics resources, such as *WeDo*, are a rich means of introducing students to the interplay between the component disciplines of STEM, and we believe they are superior to visual programming environments such as *Scratch* (Resnick et al., 2009) and *Alice* (Dann, Cooper, & Pausch, 2009) as the programming relates to student-constructed objects rather than virtual characters ("sprites") on screen (Armoni, Meerbaum-Slant, & Ben-Ari, 2015). As a result of this professional learning, a number of schools approached us to bring *WeDo* to their school to introduce it to students and interested staff. In our planning for these sessions, we decided to base the professional learning upon a *model of structured release of instructor control* that we had previously developed in 2010, as it had proven effective in supporting real pedagogical change (McDonald & Howell, 2011).

Integrated STEM in Action

It is of note that the robotics sessions were not inserted into mathematics, science or technology blocks – rather the teachers surrendered a whole teaching session (90 minutes) each week to provide adequate time to the project. The strategy used at each school site was essentially the same and incorporated staged transitioning from highly scaffolded to independent learning for both the teachers and the students. The *WeDo* robotics were deployed over four weeks, for one 90-minute session per week per class. The session for each 4-week cycle positioned the students in *modelling, exploring, challenging* and *evaluating* engagements with the robots, and concurrently the level of teacher support decreased from *highly-scaffolded* to *independent problem-solving group work.* The class teachers' roles developed from *observing and participating,* to *managing group*s, to *co-teaching* with the researchers, to operating as the *lead teacher*. The gradual release of input into student activity whilst also gradually increasing class teacher responsibility is a particular feature of this project that has proven successful in student and teacher engagement with integrated STEM education. Table 1 shows the *model of structured release of instructor control,* in which the research team members ("instructors") model the digital pedagogy to the class teacher and support them to gradually lead the session.

Week focus	Student engagement	Teacher role	Instructor role
1 Modelling	Highly scaffolded	Observing & participating	Leading
2 Exploring	Moderately scaffolded	Overseeing group work	Co-teaching
3 Challenging	Independent group work	Co-teaching	Overseeing group work
4 Evaluating	Problem-solving group work	Leading	Observing & participating

Table 1: Model of Structured Release of Instructor Control

The basis for this model of release of instructor control was built on our belief that, in order to authentically teach STEM education, as it is enacted in vocational and professional settings (Reiss & Holman, 2007), an integrated approach appears to work most effectively in primary school settings, accompanied by opportunities for students to discuss, contest, modify, and evaluate their work and how they work together. This can be achieved by scaffolded professional learning in teachers' classrooms, using contexts and digital technologies that provide the potential for cross-curricular learning, such as those afforded by *WeDo.* Conducting teacher professional learning *in situ* is one of the most effective forms of generating change (Evans, 2019; Jung & Brady, 2016; Takker & Subramanian, 2018).

During the *modelling* focus of Week 1, the students were in groups (pre-assigned by the class teacher) and the finding and placing of the component parts was managed explicitly by one of the research team. This was the opportunity to demonstrate how to interpret each instructional building frame (which component to locate and where precisely to place it – this also included decoding the 2D isometric representation on the screen to the matching 3D component and location), to model the language (e.g., a *2 by 6 flat*), and to demonstrate how to develop the visual programming. Whilst this was highly scaffolded, and some groups went ahead of their own accord, it was necessary for all students to have this basic starting point. This initiation into the *WeDo* set-up is an example of "teachable moments": the 2D to 3D matching aligns with the mathematics curriculum (Measurement & Geometry), whilst the brick identification using array terminology (i.e., the *2 by 6 flat* example above) is the underpinning of the concept of area as well as multiplication.

The groups, ideally of four students, were managed by role assignation: component finder, robot builder, assemblage checker, and computer operator, and the students were rotated through these roles each week. Working in this way, the students developed and could demonstrate the key 21st century competencies of collaboration, communication, and critical reasoning. The skill of problem-solving and the attribute of perseverance were evident every week, as students struggled to deal with instances in which their robot did not work or their systems broke down. Whilst role assignation was used for expediency, it challenged many of the students' interpersonal and collaborative skills.

In Week 2, the initial context-situating video was played on a large screen to the whole class, and then the students were encouraged to undertake the finding-constructing processes in their groups, moving at a pace that was suitable for them. In Week 3, students were merely told which robot they were to construct, and they were asked to modify the programming once they had the basic functionality operating satisfactorily. During this stage, students were actively engaged in the *AC: Technologies* – Digital Technologies Strand – as they used the drag-and-drop functionality of the visual programming component of *WeDo* to program their robot to move in different ways, make different sounds, and incorporate text or backgrounds related to the context of the introductory scenario established by the video. Week 4 was similar to Week 3 however a problem was introduced to the scenario that required alterations or modifications to both the robot construction and the visual programming. Altering the construction of the robots lends itself to exploring basic engineering processes, such as ideate, create, operate, and evaluate. During this week, students were also invited to demonstrate their modifications to other groups and were asked to describe the changes they made and the resulting impact on the operation or construction of their robot.

After several iterations of the 4-week cycle (as outlined in Table 1), we compared our field notes and recognised that there were distinctive stages through which the students moved during each session. We describe these stages as *component related* (to do with the actual pieces and how the robot is assembled) and *programming related* (to do with the visual programming required to mobilise the robots). Each stage seemed to have three

phases that were similar: *recognition, placement,* and *system,* and it could be argued that these resemble the *recall, comprehension,* and *synthesis* levels of Bloom's taxonomy (Bloom, 1956). "Recognition" involves making the link between the 2D isometric image of a piece (as displayed on the laptop screen, and the actual 3D piece in the kit. As familiarity with this process develops, students "recall" the matches and this results in a more rapid assembly of the robots. "Placement" refers to comprehending where the new piece is situated – first, in the 2D image on screen, and second, matching this position on their 3D robot. Finally, "system" describes how the pieces interplay or are synthesised into the functionality of the robot – for example, how a number of cogs work together to drive a shaft that in turn rotates a wheel.

Relating these stages to Bloom's taxonomy may provide clarity and a sense of familiarity to teachers who are engaged in integrated STEM education, as they may able to link these concepts to their initial teacher education training and current teaching practice. Figure1 illustrates the stages that students were observed to have passed through during each 4-week cycle: the component related stages (1, 2, & 3), the programming related stages (4, 5, & 6), and a final evaluation of the entire system in stage 7. We believe that this framework could assist teachers in planning for integrated STEM activities, and it also reinforces the language of the *AC: Technologies*.

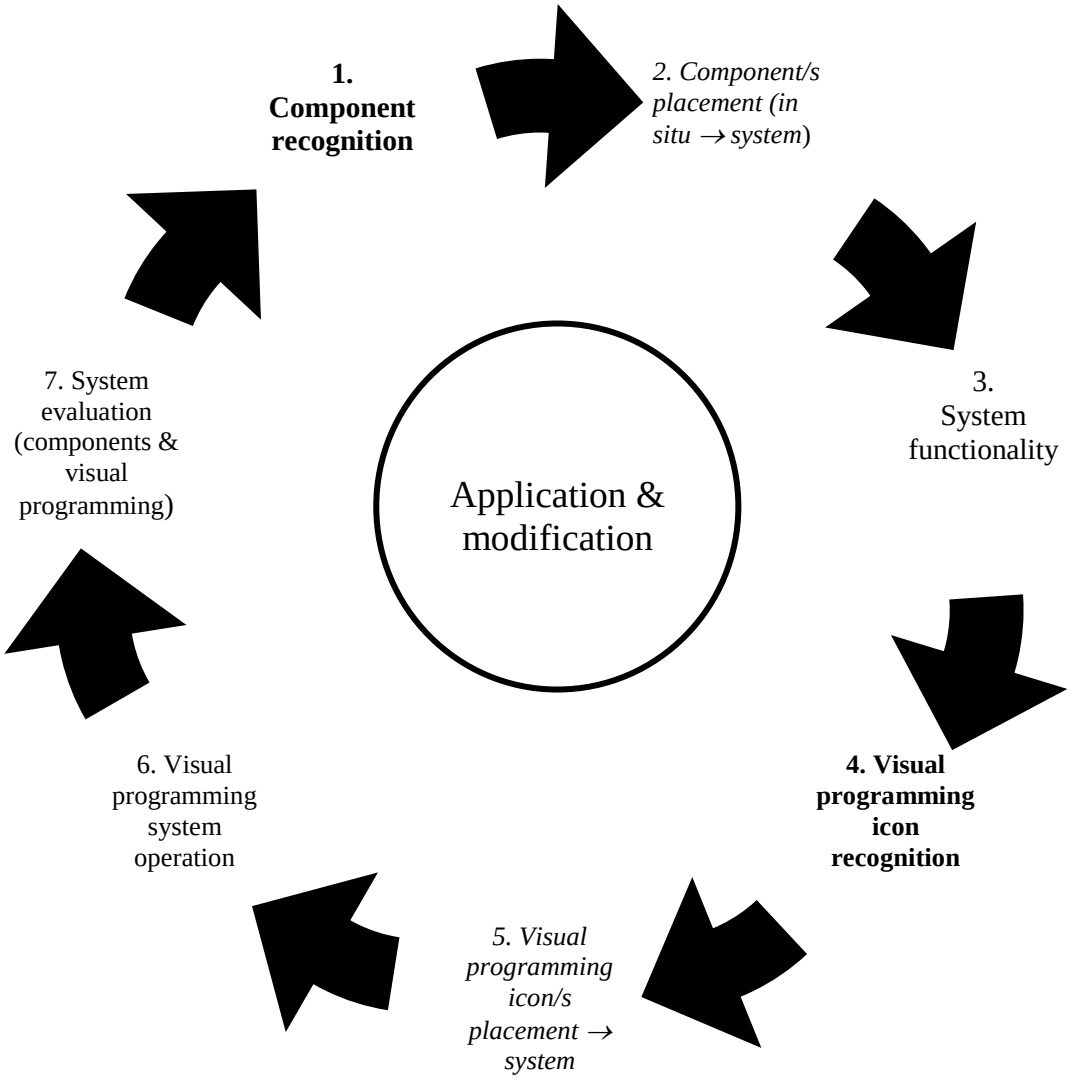


Figure 1: Framework for Integrated STEM Development

Figures 2 – 19 show examples of the seven stages using illustrations from the *WeDo* kits and visual programming software.

Figure 2: Stage 1 - Component recognition (Crown gear, as opposed to standard gear.)

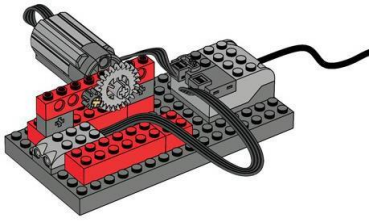


Figure 3: Stage 2 - Component placement (Crown gear is secured to the axle that has been inserted into the motor.)

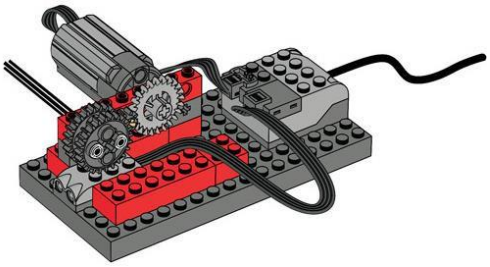
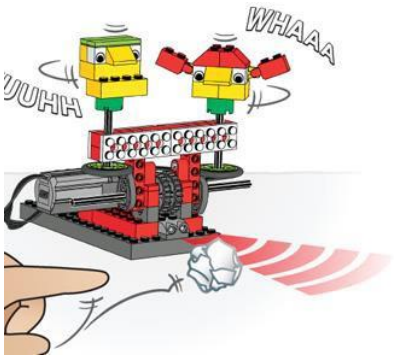


Figure 4: Stage 3 - System functionality (The system works by the crown gear meshing with the smaller cog that in turn meshes with the gear.)

Figure 5: Stage 4 - Visual programming icon recognition (This is the START icon.)

Figure 6: Stage 5 - Visual programming icon/s placement (By placing the icons in series, a system is created.)

Figure 7: Stage 6 - Visual programming system operation (By activating the program the robot can be made to do certain actions.)



Figures 8 & 19: Stage 7 - System evaluation (component & visual programming) – students make judgements about the quality and functionality of their robot, and experiment with making alterations to the construction or the programming.

Although the framework suggests transitioning from stage to stage in a one-way progression, what actually occurred in all groups was *iterative* movement between stages. In most cases this was the result of the realisation that the selected component was the wrong one or had been placed in the wrong location, and so could not join into the system under construction or could not allow the system functioning to occur as needed to activate the robot.

Discussion

The research used to underpin this paper was conducted in numerous school sites, across various year levels, and over time. As such, the limitations are minor, however we acknowledge that we have conducted this work without other collegial input other than feedback at conference.

Whilst the original LEGO *WeDo* robots were tethered (that is, the power block is attached to the laptop via a USB connection), *WeDo 2.0* in not tethered and uses Wi-Fi with iPads to power and program the robots, providing a greater scope and range of construction and movement. By using the iPads, students can also capture still shots and videos of their robots, and these captures can then be incorporated into other class work and curriculum areas. LEGO *Mindstorms* ® *EV3* robots are also not physically tethered, and by combining LEGO® elements with a programmable brick, motors and sensors, the creations can walk, talk, grab, think, shoot or anything else students can imagine. As with *WeDo*, *Mindstorms*® *EV3* provides rich opportunities for integrated STEM education, high-level programming, complex component assembly, and the development of 21st century competences. The financial outlay is significantly more than for *WeDo* however this is mediated by the greater potential for breadth and depth of learning, particularly the level of programming required for complex, staged movement incorporating higher-level mathematical concepts. Another tool for this kind of integrated STEM education, in which students construct robots and program their movements, are the Edison robots. Edison educational robots are an innovation of Microbric, an Australian company based in South Australia, and were launched in 2014. Edison robots work with any compatible LEGO brick building system, and so could value-add *WeDo* kits or basic LEGO kits, and are a powerful, engaging tool for teaching computational thinking and computer programming in a hands-on way.

These three tools authentically engage students in what we refer to as *head-heart-*

hands learning: head – cognitive demands and intellectual engagement; heart – enthusiastic engagement and development of interpersonal skills; hands – fine motor skills and spatial reasoning. We posit that these are key to authentic integrated STEM education, and whilst there are other high-profile digital tools that schools are acquiring, such as *Spheros* ®, *LittleBits* ™, *MakeyMakey* ® and *Arduino* ®, they have limited scope for covering all of the characteristics of integrated STEM education we have outlined in this paper, and the most expensive, Sphero, focuses solely on programming. The key point of difference between these digital tools and robotics is that when constructing the robots, students are actually engaging in building mechanical systems (e.g., using appropriate cogs to gear up) and thus they develop a genuine understanding of the synergies of the components and how their functionality can be altered.

Walker, Moore, Guzey, and Sorge (2018) developed nine categories for quality integrated STEM curricula and used these to develop their framework to support curriculum planning and reflection. The categories are: "(1) a motivating and engaging context, (2) an engineering design challenge, (3) integration of science content, (4) integration of mathematics content, (5) student-centred instructional strategies, (6) teamwork, (7) communication, (8) organisation, and (9) performance and formative assessment" (Walker et al., 2018, p. 332). We contend that the construction and programming of robots, as described in this paper, in conjunction with a focus on building teacher capacity and confidence by employing our *Model of Structured Release of Instructor Control,* can result in quality integrated STEM education as determined by Walker et al. (2018). Further, the use of robotics provides an engaging context for students to develop engineering skills and practices in a way that no other digital tool can achieve, and clearly provides value for investment.

Implications

The flurry of activity and fiscal commitment by governments and education authorities in regard to STEM education needs to be moderated by a ground swell of reason – we believe that educators have been left out of the frenetic STEM agenda, despite the responsibility for making it work being firmly placed on their shoulders. Are we now at the stage where all stakeholders need to pause, take breath and think long and hard about the progress of STEM education to date?

We have concerns about an approach to STEM education that: promotes one discipline over the other, that sidelines the Arts and creativity, that channels every student into what has been referred to as the "STEM pipeline" or even a STEM career, and zealously clambers onto the programming and coding bandwagon, whilst expecting that every student will have the ability and desire to engage with these skills. Rather, we champion an approach that recognises that integrated STEM education should not be conceived as the context for the *explicit* teaching of science, technology and mathematics; integrated STEM education is a space for students to apply their discipline knowledge to create products and/or solve problems that can be made or solved using engineering principles. We contend that students need to be involved in situations that demonstrate how STEM knowledge and skills and 21st century competencies can be applied in different contexts. In this paper we have presented one such approach that builds capacity in classroom teachers and engages students. Using *WeDo* as the tool to do this, allows us to also implement key components of the *AC: Techno*logies, and to spotlight 21st century competencies in classroom practice.

The implications for teachers and school leaders are not subtle: plan and map intentional, integrated STEM activity across all year levels providing access to all students;

carefully consider the procurement of digital tools; focus on skills and processes rather than specific tools that could soon be superseded; engage in authentic professional learning; explore other avenues that may provide a platform for integrated STEM education, such as Makerspaces; and improve robust, explicit teaching of science, technology and mathematics.

References

ACARA. (2015). Australian Curriculum: Digital Technologies (F-10). Retrieved from http://www.australiancurriculum.edu.au/technologies/digital-technologies/structure

AITSL. (2015). Accreditation of ITE programs in Australia: Standards and Procedures. Retrieved from https://www.aitsl.edu.au/docs/default-source/general/accreditation-of-ite- programs-in-australia.pdf?sfvrsn=3013e33c_2.

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to "Real" Programming. *ACM Transactions on Computing Education, 14*(4). Retrieved from https://doi.org/10.1145/2677087

Bellanca, J.A., & Brandt, R.S. (2010). *21st century skills: Rethinking how students learn*. Vol. 20. Bloomington, IN: Solution Tree Press.

Blackley, S., & Howell, J. (2015). A STEM Narrative: 15 Years in the Making. *Australian Journal of Teacher Education, 40(*7). https://doi.org/10.14221/ajte.2015v40n7.8

Bloom, B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. New York: David McKay Co Inc.

Collins, A., & Halverson, R. (2009). *Rethinking education in the age of technology*. New York, NY: Teachers College Press.

Dann, W., Cooper, S., & Pausch, R. (2009). *Learning to Program with Alice* (2nd ed.). Pearson.

Eisenberg, M. B., & Johnson, D. (1996). *Computer skills for information problem-solving: learning and teaching technology in context.* Retrieved from http://www.ericdigest.org/1996-4/skills.html

Evans, L. (2019) Implicit and informal professional development: what it 'looks like', how it occurs, and why we need to research it. *Professional Development in Education, 45*(1), 3-16. https://doi.org/10.1080/19415257.2018.1441172

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6-year-old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63,* 87-97. Retrieved from http://dx.doi.org/10.1016/j.compedu.2012.11.016

Grover, S., & Pea, R. (2013). Computational thinking in K-12, a review of the state of the field.

Educational Researcher, 42(1), 38-43. https://doi.org/10.3102/0013189X12463051

Heitlin, L. (2014). *Computer Science: Not Just an Elective Anymore.* Retrieved from www.edweek.org/ew/articles/2014/02/26/22computer_ep.h33.html

Jones, S. P. (2011). *Computing at School: International comparisons.* Retrieved from http://www.computingatschool.org.uk/data/uploads/internationalcomparisons-v5.pdf

Jung, H., & Brady, C. E. (2016). Roles of a teacher and researcher during in situ professional development around the implementation of mathematical modeling tasks. *Journal of Mathematics Teacher Education, 19*(2-3), 277-295. https://doi.org/10.1007/s10857-015-9335-6

Kafai, Y. B., & Burke, Q. (2013). *Computer Programming Goes Back to School.* Retrieved from www.edweek.org/ew/articles/2013/09/01/kappan_kafai.html https://doi.org/10.1177/003172171309500111

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming.* MIT Press. Retrieved from https://searchworks.stanford.edu/view/10610063

McDonald, S., & Howell, J. (2011). Watching, creating and achieving: Creative technologies as a conduit for learning in the early years. *British Journal of Educational Technology (BJET), 43*(4), 641-651. https://doi.org/10.1111/j.1467-8535.2011.01231.x

Pinkston, G. (2015). *Forward 50, Teaching Coding to Ages 4-12: Programming in the Elementary School.* Retrieved from http://dl4.globalstf.org/?wpsc-product=forward-50- teaching-coding-to-ages-4-12-programming-in-the-elementary-school

Reiss, M., & Holman, J. (2007). S-T-E-M Working Together for schools and colleges. *The Royal Society handbook of research on environmental education*. New York: Routledge.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60-67. Retrieved from https://dl.acm.org/citation.cfm?id=1592779

Sanders, M. (2009). STEM, STEM Education, STEMmania. *The Technology Teacher, December/January*, 20-26. Retrieved from https://vtechworks.lib.vt.edu/bitstream/handle/10919/51616/STEMmania.pdf?sequence

Takker, S., & Subramaniam, K. (2018). Teacher Knowledge and Learning In-situ: A Case Study of the Long Division Algorithm. *Australian Journal of Teacher Education, 43*(3). http://dx.doi.org/10.14221/ajte.2018v43n3.1

Teaching Institute for Excellence in STEM (2010). What is STEM education? Retrieved from http://www.tiesteach.org/stem-education.aspx

Walker, W. S., Moore, T. J., Guzey, S. S., & Sorge, B. H. (2018). Frameworks to develop integrated STEM curricula. *K-12 STEM Education, 4*(2), 331-339. http://dx.doi.org/10.14456/k12stemed.2018.14

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). *Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age.* Retrieved from http://www.fit-in-it.ch/sites/default/files/downloads/Running%20on%20Empty%20(ACM).pdf

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. http://dx.doi.org/10.1145/1118178.1118215.