Mapping Computational Thinking and Programming Skills Using Technacy Theory

Jayanti Nayak(✉), Therese Keane, and Kurt Seemann

Swinburne University of Technology, Melbourne 3122, Australia
{jnayak,tkeane,kseemann}@swin.edu.au

Abstract. *Digital Technologies* as a compulsory subject was introduced in the Australian Curriculum to enable students to build up their confidence in becoming creative developers of digital solutions and to develop thinking skills in problem solving. This theoretical paper examines a conceptual framework with the potential to form a working model for teachers teaching Computer Science/Digital Technologies in K-12 classrooms. Using Technacy Theory as a framework promises ideas for differentiating technology education by means of setting appropriate developmental expectations. This paper explores how the teaching of computational thinking and programming, key concepts found in the teaching of Computer Science subjects, can be mapped to the *Technacy and Innovation Chart* setting developmentally appropriate expectations in the teaching and learning of these subjects.

Keywords: Computational thinking · Technacy theory · Programming · Child development

1 Introduction

The last decade has seen exponential growth in digital technologies leading to an influx of innovative solutions. Students as young as seven years of age are now exposed to several digital devices and are generally quick to engage with these technologies. There is growing awareness that students should not just be consumers of technology but increasingly creators of innovative solutions. To achieve this, students from a very young age should be encouraged to create products and solutions whilst developing their problem-solving skills, critical thinking skills, digital literacy, creative thinking skills, collaboration and communication [1]. These are key elements of computational thinking, a term coined by Wing [2], defined as "thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer (or human) can effectively carry out". In more recent times, education policy makers have realised the importance of developing computational thinking skills and have endeavored to incorporate them across various subjects like mathematics, humanities and computer science. Mannila et al. [3] demonstrated that computational thinking skills can be developed within all disciplines, while Selby and Woollard [4], through a systematic literature review, stated that computational thinking is inherently embedded in computer science education. Other studies

[5, 6] have also shown that computational thinking skills can be acquired by students through the teaching of computer science concepts, programming and robotics. There has been much debate and discussion on how schools can change their curriculum to incorporate computational thinking from a young age. To illustrate this point, in 2015, the Australian Curriculum Assessment and Reporting Authority (ACARA) endorsed Digital Technologies as one of the two core subjects of the Technologies Curriculum [7] for students in Foundation to Year 10. The Digital Technologies curriculum aims to develop knowledge, understanding and skills to create digital products with programming as the vehicle for learning.

Teaching a new curriculum can pose challenges for existing teachers who have taught a very different curriculum and may not have sufficient technical knowledge and/or skills to adapt their pedagogical content knowledge to the new curriculum content [8]. Studies have indicated that teachers need adequate professional development to support teaching and implementation of new curriculum. Hill, Keane and Seeman [9] advocate for a clear set of guidelines, framework and practices that support teachers to improve student learning outcomes. While other disciplines such as English and Mathematics have developed pedagogies for literacy and numeracy respectively, little research exists to examine equivalent ideas for technological areas of learning such as Digital Technologies/Computer Science education. This paper will explore a promising new area known as Technacy Genre Theory as a framework to engage students in programming tasks and improve their computational thinking skills. Technacy provides a holistic framework with social, environmental and human factors, with significant discourse around developmental indicators in technical education. The Technacy Theory is discussed in detail in Sect. 3 of this paper.

2 Theoretical Background

2.1 Computational Thinking

Wing highlights the importance of computational thinking when she states that it is the “thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent” [2]. Progressive thinkers such as Margaret Mead have long argued that: “children must be taught how to think and not what to think”. It can be argued that computational thinking skills are important for students. Papert [10], one of the designers of the LOGO programming language aimed at school-aged students, has for many decades advocated that learning to program at an early age allowed children to develop problem solving and logical thinking skills.

At the core of computational thinking is the ability to solve complex problems by breaking it down into small procedures [3, 11]. These procedures include the reliance of the following skills: logical thinking, algorithmic thinking, problem-solving skills with understanding of abstraction, generalisation, and decomposition.

2.2 Literacy in Digital Age

Bers [12] defined literacy as “the ability to use a symbol system and a tool to comprehend, generate, communicate and express ideas or thoughts by making a sharable product that others can interpret". Education and policy makers acknowledge the importance of *technological literacy* to help prepare children for the 21st century. Keane, Keane and Blicblau [13] assert that *digital literacy* is as important as being literate and numerate. A review of scholarly literature, documents and reports over the last 30 years suggest that *digital literacy* is also referred to as *computer literacy*, *information literacy* and *technological literacy*. Often, the contextual meaning of the term is in relation to the technology of the given period. For example, computer literacy in the 1980s referred to working knowledge of desktop computers, whilst in the early 2000s information literacy incorporated web development skills. Currently, there are several initiatives to define and mandate assessment of *technological literacy* [14, 15].

2.3 Defining Technacy

While being literate and numerate are fundamental skills, with the dynamic and evolving nature of the digital environment, it is important to also be technate (to comprehend technologies) [16]. Technacy and its adjective technate [17] is defined as "the holistic understanding of technology in relation to the creation, design and implementation of technology projects". Drawing parallels between literacy, numeracy and technacy, Seemann and Talbot [18] argue that:

> *Just as there are levels of competence in literacy from writing one's name to writing profound poetry, and in numeracy from adding a few numbers together to compiling a fundamental formula in physics, so too there is a range in technacy from being skilled in joining materials together or repairing equipment to being innovative in the design and development of appropriate technologies and systems.*

Table 1 offers a comparative structure and parallel proposition of literacy and technacy with attention to the higher order cognitive demands of forming abstractions and inferences [19].

3 Technacy Theory

3.1 Background to Technacy Theory

Technacy Theory was derived from existing practices in cross-cultural technology education among Indigenous Australian communities. The Australian Science, Technology and Engineering Council [ASTEC] acknowledged that an improvement in science and technology education was necessary, stating that: "technacy – will be as vital to students of 21st century as literacy and numeracy were to Australians who grew up in the 20th century" [20]. ASTEC provided funding to look into innovative ways of teaching and assessing technological knowledge. One of the recommendations was to incorporate technacy education in primary and secondary school curricula across Australia [21]. Additionally, a generic framework, the Technacy and Innovation Chart (Fig. 1) was developed to identify, assess and measure developmental indicators of technological thinking and doing, in the student learner.

Table 1. Comparative structures and forms for literacy and technacy [19]

Structure	Key concepts in language comprehension (**literacy**)	Equivalent concepts in technological comprehension (**technacy**)
Lexicon	*Library* of words	*Library* of digital material and resources
Syntax	*Grammar*. Thought structure techniques to **choose and logically sequence** words (a story, a sentence)	*Sequence and Composition.* Thought structure and techniques for **choosing and sequencing** the logical assembly of digital or material resources
Semantics	*Meaning making.* Creating and communicating **sensible** paragraphs, reports, narratives, and expositions	*Meaning making.* Creating and communicating **working** components or programs that form all or part of a solution or idea: the outcome works, achieves the brief
Inference making and abstraction	*Reading Comprehension.* Understanding words and sentences in their wider real-world context of application	*Technological Comprehension.* Understanding technologies and systems in their wider real-world context of application

3.2 Technacy and Innovation Chart

The *Technacy and Innovation Chart* (Fig. 1) is a 3x3 grid that provides a model to understand how people respond to increasing degrees of complexity when solving technological challenges. Along the horizontal axis are the three phases *Emergent, Competent* and *Sophisticated* which form *Technacy Expectations and Complexity* whilst the vertical axis are the three domains of innovation: *Play, Consolidation* and *Pioneer*, which describe the level of innovative responses to the technological challenges addressed by the student learner. The chart is organised by child *Developmental Domains* and *Phases of Learning Complexity* in technologies.

The grid has been developed to reflect the learning journey of the student. The learning journey of the technology student typically begins with *Emergent Play*. As the learner gains confidence, they progress from left to right of the chart, attempting more complex tasks. The column progression from left to right (across the rows) shows the degree of task complexity. The degree of complexity is often set by the teacher. The row progression from bottom to top is attributed to the degree of personal initiative shown by the learner. The vertical progression is demonstrated by the student with the assistance of educational scaffolding and is subject to factors such as developmental ability [23]. The highest category in the framework aims to develop skills that prepare students to become independent creators, innovators and pioneers in their field. The centre of the Technacy Chart (*Competent Consolidation*) and the perimeter of the edges around it is

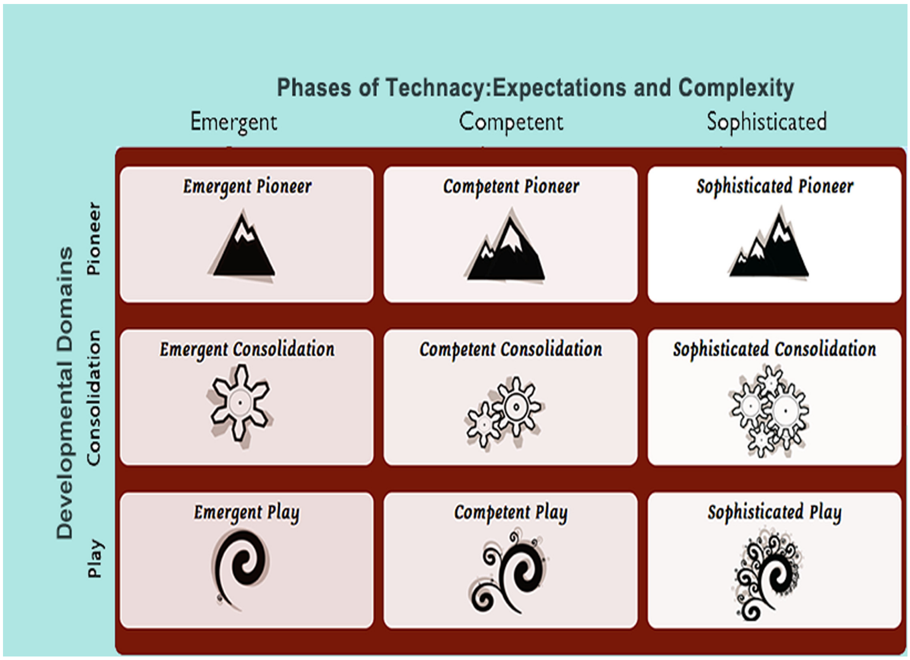

Fig. 1. Technacy and innovation chart [22]

where most learners tend to settle before engaging with higher demand technological challenges and pioneering expectations [23].

3.3 Mapping Key Dimensions of the Technacy and Innovation Chart

To understand the relationship between programming concepts, computational thinking, child development and technacy as discussed above, these have been mapped by the authors of this paper against the Technacy and Innovative Chart as shown in Table 2.

Play Domain. Students create knowledge in different ways. *Play* as a strategy has been identified as a powerful way to engage students in their learning, especially to form cognitive schema. Piaget [25], Vygotsky [26], Resnick [27], and Fleer [28] assert that play is essential for intellectual and cognitive development in children. Practical activities provide students with the opportunity to experiment and build ideas from their playful experiences.

In computer programming, play involves students interacting with on-screen objects such as *Scratchy* the cat in the programming language called Scratch, or the *Turtle* in the programming language called Python. The focus in the *Play Domain* is to foster early stages of schema development. The teaching of computer programming using on-screen objects assists in developing problem-solving skills, logical thinking skills and algorithmic thinking skills. The manipulation of on-screen objects requires the ability to comprehend technology knowledge being learned. Hrbacek, Kucera and Strach [29]

Table 2. Relationship: technacy domains, computational thinking and programming concepts

	Play domain	Consolidation domain	Pioneer domain
Goal	Engage participants' interest	Develop basic programming concepts	Learn to plan, design and develop solutions
Piaget's Developmental Stages [25]	Pre-operational stage	Concrete Operational stage	Formal Operational stage
Computational Thinking	Problem-solving skills Algorithmic thinking Logical thinking	Abstract thinking Problem-solving skills	Decomposition Generalisation
Programming Concepts	Low-level concepts [24]	Mid-level concepts [24]	High-level concepts [24]
Programming Tool	Drag-n-drop	Text-based	

determined that the age of 8 years is the optimal time to introduce early constructs in programming to students. Scratch and Blockly, block-based visual programming languages, are popular in the classroom with younger students due to their drag-and-drop features.

As the student develops confidence, teachers can facilitate their progress from *Emergent Play* to *Competent Play* towards *Sophisticated Play* by setting more higher order tasks to match the complexity of the project. Students will need assistance, guidance and directions from teachers [23] to progress through the three phases of the *Play Domain.*

Consolidation Domain. Block-based programming creates an interactive and engaging environment making it possible for students to create simple games without writing a single line of code. However, block-based languages have several limitations on what students can do. Text-based programming languages, on the other hand, offer greater flexibility at the expense of having pre-affirmed knowledge. At some stage, students need to be encouraged to move to a text-based programming language to extend themselves. There are several high-level programming languages, each with its own syntax and semantics; however, ultimately, from a pedagogical point of view, they all have similar underlying programming constructs. Butler and Morgan [24] grouped generic programming concepts into levels of difficulty - low, mid and high - based on their complexity as seen in Table 2.

The *Consolidation Domain* has three phases – *Emergent Consolidation*, *Competent Consolidation* and *Sophisticated Consolidation.* In the *Consolidation Domain*, the focus is on progressing from early discovery to higher cognitive demand in difficulty thus advancing deep knowledge of the subject in a meaningful way [30]. In the *Emergent Consolidation Phase*, teachers can assign tasks that focus on early discovery concepts [24]. As students develop confidence, they should be encouraged to be stretched into the *Competent Consolidation Phase* using programming concepts such as decisions, loops and arrays: the *Competent Phase* introduces codified established knowledge and techniques. The Technacy Chart is consistent with Papert's idea of low floor, high ceiling [10], a metaphor where a medium with a low floor assures a low barrier to entry, but the high ceiling simultaneously does not constrict creativity.

Pioneer Domain. The *Pioneer Domain* is where students demonstrate a high capacity to manage novelty and express new ideas. The *Pioneer Domain* has three levels of capability demonstrated by the learner. The *Emergent Pioneer* accommodates student learners who demonstrate a high and consistent capacity for imagination but have not yet affirmed competencies to execute their ideas. The *Competent Pioneer* also has a capacity to imagine new ideas, but unlike the *Emergent Pioneer,* has knowledge and skills of known techniques to execute their ideas. The *Sophisticated Pioneer* not only presents novel and creative ideas and has the skills to execute them, but also demonstrates an ability to create new methods and frameworks to solve novel problems.

4 Conclusion

There is a growing recognition that "coding is the new literacy" [31]. Most technology researchers agree that introducing coding to students at an early age is considered a long-term investment in bridging the skills gap between technology demands of the labour market and the availability of people to fill them [32]. The Technacy Developmental Framework as presented in this paper can be applied to the Australian Digital Technologies subject or any computing course which demands students to have skills in programming and computational thinking. This Framework provides a method in which to map and describe how a student responds to increasing demand associated with learning computer science content such as developing programming skills.

This paper asserts that there is promise in combining cognitive development research and computational thinking theory with Technacy Theory. The *Technacy Innovation Chart* offers considerable opportunities for further research when combined with computational thinking concepts, including offering a common dialogue to describe the learning taking place.

References

1. Voogt, J., Roblin, N.P.: A comparative analysis of international frameworks for 21st century competences: implications for national curriculum policies. J. Curriculum Stud. **44**(3), 299–321 (2012)
2. Wing, J.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)
3. Mannila, L., Dagienė, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., et al.: Computational thinking in K-9 education. In: Proceedings of the Working Group Reports on Innovation & Technology in Computer Science Education Conference, pp. 1–29 (2014)
4. Selby, C., Woollard, J.: Computational thinking: the developing definition. In: Annual Conference on Innovation and Technology in Computer Science Education, University of Kent, Canterbury: ACM Special Interest Group on Computer Science Education (SIGCSE) (2013)
5. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada, vol. 1, p. 25 (2012)

6. García-Valcárcel-Muñoz-Repiso, A., Caballero-González, Y.-A.: Robotics to develop computational thinking in early childhood education. Media Educ. Res. J. **27**(1), 63–72 (2019)
7. Australian curriculum assessment and reporting authority technologies. https://www.australiancurriculum.edu.au/f-10-curriculum/technologies. Accessed 20 May 2020
8. Webb, M., et al.: Computer science in K-12 school curricula of the 2lst century: why, what and when? Educ. Inf. Technol. **22**(2), 445–468 (2016). https://doi.org/10.1007/s10639-016-9493-x
9. Hill, E., Keane, T., Seemann, K.: Pedagogies and practices for developing innovation capability: beyond the AITSL standards. Paper presented at the 10th Biennial International Design and Technology Teacher's Association Research Conference (DATTArc), Swinburne University of Technology, Hawthorn, VIC (2018)
10. Papert, S.: Mindstorms: Children, Computers and Powerful Ideas, 2nd edn. Basic Books, New York (1993)
11. Vee, A.: Understanding computer programming as a literacy. Literacy Compos. Stud. **1**(2), 42–64 (2013)
12. Bers, M.U.: Coding, playgrounds and literacy in early childhood education: the development of KIBO robotics and scratch Jr. Paper presented at the IEEE Global Engineering Education Conference (EDUCON), Santa Cruz de Tenerife, Spain, pp. 2094–2102 (2018)
13. Keane, T., Keane, W.F., Blicblau, A.S.: Beyond traditional literacy: learning and transformative practices using ICT. Educ. Inf. Technol. **21**(4), 769–781 (2014). https://doi.org/10.1007/s10639-014-9353-5
14. Avsec, S., Jamšek, J.: Technological literacy for students aged 6–18: a new method for holistic measuring of knowledge, capabilities, critical thinking and decision-making. Int. J. Technol. Des. Educ. **26**(1), 43–60 (2015). https://doi.org/10.1007/s10798-015-9299-y
15. Thorsteinsson, G., Olafsson, B.: Piloting technological understanding and reasoning in Icelandic schools. Int. J. Technol. Des. Educ. **26**(4), 505–519 (2015). https://doi.org/10.1007/s10798-015-9301-8
16. Seemann, K.: Technacy education: understanding cross-cultural technological practice. In: Fien, J., Maclean, R., Park, M.G. (eds.) Work, Learning and Sustainable Development. Technical and Vocational Education and Training: Issues, Concerns and Prospects, vol. 8, pp. 117–131. Springer, Dordrecht (2009)
17. Macquarie dictionary: the Macquarie dictionary, 7th edn. https://www.macquariedictionary.com.au/features/word/search/?search_word_type=Dictionary&word=technacy7. Accessed 20 May 2020
18. Seemann, K., Talbot, R.: Technacy: towards a holistic understanding of technology teaching and learning among aboriginal Australians. Prospects **25**(4), 761–775 (1995)
19. Seemann, K.: Your super-powers: applied design led innovation and working technologically. Keynote address to the Design and Technology Teacher's Association of the Australian Capital Territory, Annual Conference, p. 25. Design and Technology Teacher's Association of the Austrailan Capital Territory, Canberra, ACT (2019)
20. Desert Knowledge CRC.: Media Release: Technacy – key to the education revolution - Ninti One, p. 2. DKCRC, no place of publication (2008)
21. Seemann, K.: Linking Desert Knowledge with Pedagogy Research for Middle School Curriculum, vol. 1, p. 23. Southern Cross University and Design Knowledge Coorperative Research Centre, Coffs Harbour, NSW (2008)
22. Seemann, K.: Technacy chart. http://technacy.org/tools/chart/106. Accessed 28 Aug 2019
23. Cerovac, M., Seemann, K., Keane, T.: Systems engineering: identification and fostering of inferential and social skills. Paper presented at the 10th Biennial International Design and Technology Teacher's Association Research Conference (DATTArc), Swinburne University of Technology, Hawthorn, VIC (2018)

24. Butler, M., Morgan, M.: Learning challenges faced by novice programming students studying high level and low feedback concepts. In: Proceedings 2007 ASCILITE Singapore, pp. 99–107 (2007)
25. Piaget, J.: Cognitive development in children: development and learning. J. Res. Sci. Teach. **2**(3), 176–186 (1964)
26. Vygotsky, L.S.: The role of play in development. In: Cole, M., Jolm-Steiner, V., Scribner, S., Souberman, E. (eds.) Mind in Society, pp. 92–104. Harvard University Press, Harvard (1978)
27. Resnick, M.: Lifelong Kindergarten: Cultivating Creativity Through Projects, Passion, Peers, and Play. MIT Press, Cambridge (2017)
28. Fleer, M.: Play in the Early Years. Cambridge University Press, New York (2013)
29. Hrbáček, J., Kučera, M., Strach, J.: Teaching robot programming can be a new opportunity for technical subjects of study. In: IEEE 11th International Conference on Emerging eLearning Technologies and Applications (ICETA), pp. 133–137. IEEE, New York (2013)
30. Winslow, L.E.: Programming pedagogy—a psychological overview. ACM SIGCSE Bull. **28**(3), 17–22 (1996)
31. Bers, M.U.: The TangibleK Robotics program: applied computational thinking for young children. Early Child. Res. Pract. **12**(2), 1–20 (2010)
32. DePryck, K.: From computational thinking to coding and back. In: Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, Spain, pp. 27–29. Association for Computing Machinery (ACM), New York (2016)