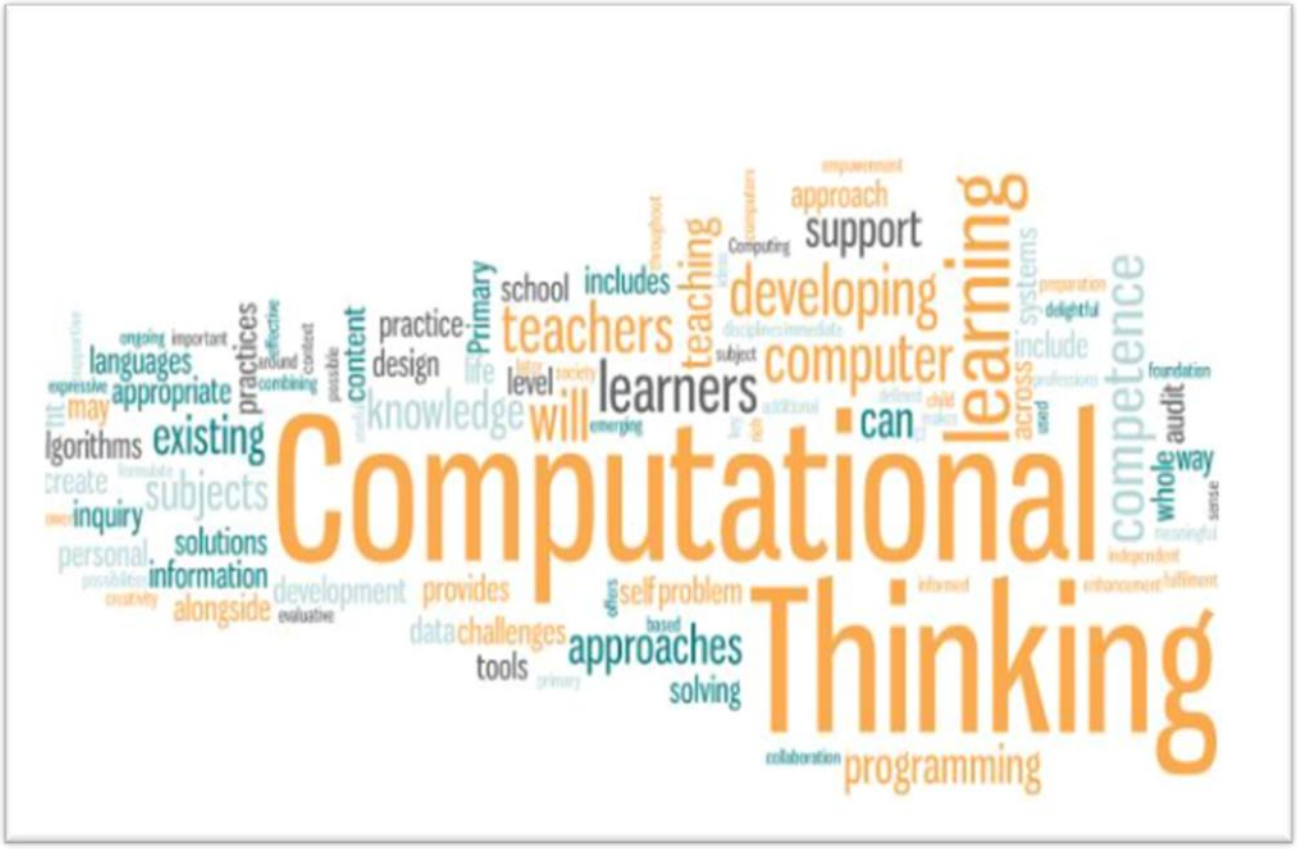
Review of Literature on Computational Thinking



Authored by Dr Richard Millwood, Nina Bresnihan, Dermot Walsh and Joy Hooper

With advice from Glenn Strong and Dr Stephen Powell

Table of contents

Executive summary

Computational Thinking (CT) is emerging as a key competence across all disciplines, professions and throughout society. It can be defined as combining problem solving and design to create useful solutions, informed by the possibilities that Computing offers.

Where computers are used to formulate solutions, additional independent learning is possible (in subject content and Computational Thinking) as the computer itself provides a rich context for enhancement of expressive creativity and evaluative power of learners' ideas. Collaboration around a computer makes learning meaningful and supportive. In this sense, Computational Thinking is both delightful and effective for learners.

Appropriate competence in Computational Thinking at primary level provides an important foundation for the development of the whole child - their immediate fulfilment, ongoing empowerment and preparation for later life. In other subjects, Computational Thinking will support inquiry-based learning at every level and include wider civic and leisure interests. Furthermore, awareness of the potential of information systems and computer algorithms will help learners tackle with insight the personal, societal and value challenges as they grow.

Computational Thinking can be introduced and developed in all subjects, and indeed is already present in Primary teaching practice, and so embedding Computational Thinking is also about recognition of existing practices, not simply innovation. Evidence suggest that teaching programming in isolation does not lead to transfer. Therefore using Computational Thinking to support learning challenges across the curriculum should be the starting point, in order to embed it as a transferable competence.

Computational Thinking depends on acquiring crafts of user-centred design, data analysis and programming through developing skills in practice (hands). Essential character development includes dispositions of empathy, inquiry, imagination, perseverance and concern for quality (heart). Knowledge required includes the way in which information systems and computer technology work, are programmed and may be debugged through facts, mental models and strategies (head). Designing data structures and algorithms to create programs through coding is both vital and helpful: attainable by Primary learners aided by sophisticated but easy to begin programming languages and tools which support their developing capacity for self-directed learning. These languages and tools can be revisited with increasing levels of sophistication as the learner grows.

Developing teachers' teaching of Computational Thinking will include developing their own content knowledge and their pedagogical knowledge. The sensible way to approach this is to build on teachers' existing know-how, which so far they may not have recognised as Computational Thinking, and to

adopt creative and problem solving practices from other subjects when engaging in Computational Thinking learning activities.

Approaches to Continuing Professional Development should be diverse and adapted to fit different teachers' experience, preference and life circumstances. Online collaborative approaches designed to fit within the Irish Cosán and Digital Strategy frameworks would be particularly appropriate for rurally isolated teachers. Exploitation of national conferences, regional Teachmeets and existing Communities of Practice will ensure sustainability alongside more traditional approaches. But to be successful, it is crucial to consider the continuity and progression of each teacher's competence alongside the whole-school approach being taken. To this end it is recommended that school self-audit is combined with individual teacher audit to identify strengths and weaknesses and thus guide both strategic and personal planning.

1. How do we define Computational Thinking?

Computational Thinking is a term that has emerged widely in the last decade. Although first used in 1980 by Seymour Papert[1] it is only in the last decade that the term Computational Thinking has gained traction. But its proponents are plagued by uncertainty about the concept, which "has been criticized for vagueness, ambiguous definitions and visions of Computational Thinking, and arrogance, as well as for bold, unsubstantiated claims about the universal benefit of Computational Thinking" (Tedre and Denning, 2016, p.120).

It is not to be confused with Computing or Computer Science, a discipline specific to the computer itself with its capacities, methods and theory. Computational Thinking can be observed in all disciplines, notably in Art where, for example, the development of 'Alphabet Plastique' techniques by Herbin and Vasarely in the middle of the 20th Century. These artists recognised the importance of the computer and automation and developed creative methods for their assistants to generate works of art based on rules and patterns. Computational Thinking has become vital in all disciplines and the vast majority of vocations, and can be argued to be important to leisure, sport and civic society. Hence it is important to build Computational Thinking approaches naturally into all areas of the curriculum rather than to narrow its focus on 'coding' or programming.

To establish a clear definition, this review starts by asking, "what is it that we expect of learners as an outcome of…." and argues that the answer is a holistic one of rounded competence that is learner-centred: both their ability to be effective and their fulfilment through the act of learning. We lean on Priestley's argument for:

> "focus on developing the capacity of young people to act within the world, and characterised by more dialogical and collaborative pedagogies, continuous approaches to assessment and higher degrees of teacher agency as they act as curriculum developers."
>
> (Priestley 2016, p.5)

But being effective and fulfilled does not simply relate to those aspects that are usually cited as learning outcomes, which are the teacher's goal with respect to the learner. The most popular framework for learning outcomes, Bloom's taxonomy as revised by Anderson, L. W. *et al.* (2000), is written to help in developing planning and questioning for the teacher.

[1] Papert worked with Solomon and Feurzig to invent the popular programming language Logo in 1967, which inspired the current popular programming language and online community of learners found in Scratch.

A weakness of this kind of framework is the way it misses out dispositional development, such as 'perseverance', 'openness' or 'concern for quality' all of which are vital in developing solutions through Computational Thinking.

Thus Bloom's disconnects rather than combines aspects of learning, by organising them in levels which are not clearly related to learning progression, but this is how it is often interpreted by teachers. Critics have observed that teachers do not use its depth effectively nor are they aware of its weaknesses. A more practice-friendly framework, remembered through the simple metaphor of 'head, hands and heart', representing knowledge, craft and character (Millwood 2018), may be better suited to define learning outcomes for lesson planning.

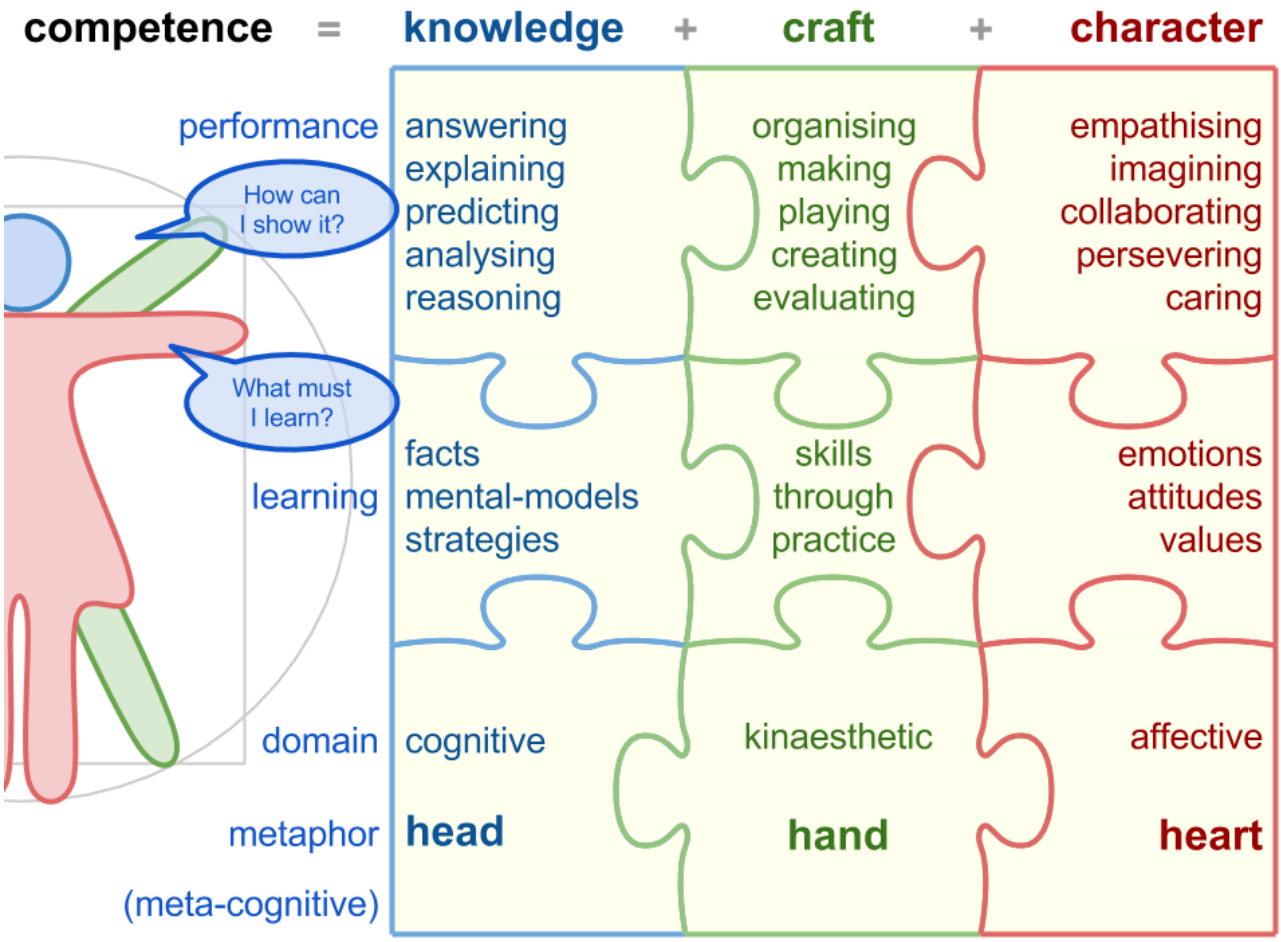


Fig 1.2 Competence = knowledge + craft + character (Millwood 2018)

In this analysis, the overall learning outcome is that the learner is competent - effective in using their capacities to achieve. Such competence is a holistic combination of knowledge, craft and character.

For example, to produce a story about the scientific phenomenon of a butterfly's life-cycle, one might be involved in **collaborating** with others, **explaining** it through **making** a poster. Each of these:

collaborating, explaining and making, are combined seamlessly in life and interact as the work proceeds, not necessarily in this order:

- To explain the story one must learn the **facts** of the butterfly's life-cycle and construct a **mental model** of how transformation from egg to butterfly takes place in dynamic sequence (head).
- To make the poster, one must learn **skills through practice** using tools and media (hands).
- To collaborate one must master **emotions** and manage **attitudes** towards others (heart).

1.1 A clear definition that is readily understood

Computational Thinking can be defined as competence in problem solving & design to create useful solutions, informed by the possibilities that Computing offers.

1.2 Justification and rationale

1.2.1 Introduction

The teaching of computing in schools can be broadly justified under 3 categories:

1. The development of Computational Thinking;
2. The development of more broadly applicable skills such as digital literacy, broadened social participation and 21st century skills and
3. Preparation for participation in the technology sector.

This introduction explains each of these, then focusses on 1. The development of Computational Thinking.

1. The development of Computational Thinking

The Digital Strategy for Schools 2015-2020 Action Plan 2017 proposes a number of curriculum reforms including an investigation of the role of coding as part of the primary school maths curriculum. This is a narrow focus and although coding (programming) is a vital craft that supports competence in Computational Thinking, it is by no means clear that it is best thought of as a mathematical topic. Mathematical concepts are clearly linked, but Computational Thinking defined as the problem solving

& design to create useful solutions demands a much broader, creative and playful approach engaging children's imagination.

2. The development of more broadly applicable skills

The Action Plan also recognises the need to embed ICT in other emerging curriculum developments. During the launch of the plan Minister Bruton spoke of, "the potential of digital technologies to enhance teaching, learning and assessment to help students become engaged thinkers, active learners, knowledge constructors and global citizens". This statement very much aligns to a concept of Computational Thinking that argues, not just that understanding how our digital world works is a critical 21st century skill, but also that Computational Thinking can actually be applied to many non-digital spheres and has the potential to support the development of knowledge, craft and character development required for creative thinking and problem-solving.

3. Participation in the technology sector

The global economic demand for Information Communication Technology (ICT) skills has been a key driver for the introduction of Computing in schools. Major reports in the UK (Livingstone & Hope, 2011) and US (Wilson et al., 2010) highlighted the danger to advanced economies of the lack of clear education policies centred around the subject of Computing, leading to a flurry of interest on both sides of the Atlantic around its teaching.

Ireland's current ICT Skills Action Plan identified a shortage of up to 864,000 ICT professionals across the European Union (EU) and the European Economic Area (EEA) by 2015 (Department of Jobs, Enterprise and Innovation, 2014, p.4] with 44,500 job openings forecast to arise in Ireland over the period to 2018. If second level computing is to be successful in preparing for such jobs, then the foundations must be laid in Primary to build both competence and inclination.

1.2.2. Why Computational Thinking?

Jeanette Wing's 2006 call for Computational Thinking to be added "to every child's analytical thinking" sparked a resurgence of interest in the notion that the kinds of problem solving used by computer scientists could have broader applications. Indeed, if we look to who many consider to be the father of the field, Seymour Papert (1980), we see an even broader argument for its importance: that "certain uses of very powerful computational technology and computational ideas can provide children with new possibilities for learning, thinking, and growing emotionally as well as cognitively." (Papert, 1980,

p17-18). This is a powerful idea; the accompanying Programmes of Study to the English Computing Curriculum place Computational Thinking at its core with the opening claim that "A high-quality Computing education equips pupils to use Computational Thinking and creativity to understand and change the world." (Department for Education, 2014).

This claim for Computing as a source for Computational Thinking is reinforced more recently by the Royal Society's report of developments in the UK countries:

> "The broad subject of computing – covering the three vital areas of computer science, digital literacy and information technology (IT) – has become mandatory in English schools from ages 5 to 16. In Scotland, we have seen the implementation of the Significant Aspects of Learning, a framework where computing is broken down into distinct areas of knowledge. In Wales, the Digital Competence Framework is bringing computing in schools to the forefront, while Northern Ireland has continued to deliver a comprehensive computing framework."
>
> (The Royal Society, 2107)

However, translating these aspirations into practical models for the content, delivery and assessment of Computational Thinking has proven to be challenging for educators. Much work has been done in an effort to resolve this in recent years, one outcome of which has been the pragmatic adoption of 'working definitions'. For example in the US, Barr and Stephenson (2011, p112) reported on "developing an operational definition of Computational Thinking for K-12", which would take into account the practical difficulties of bringing Computational Thinking into classrooms from kindergarten through to year 12 (Senior Cycle in Ireland). Indeed, the "definition", which was developed by the US Computer Science Teachers Association and International Society for Technology in Education, can give a clear idea of the kind of learning outcomes that proponents of Computational Thinking in the classroom envisage:

> "Computational thinking is a problem-solving process that includes (but is not limited to) the following characteristics:
>
> - Formulating problems in a way that enables us to use a computer and other tools to help solve them
> - Logically organizing and analyzing data
> - Representing data through abstractions such as models and simulations
> - Automating solutions through algorithmic thinking (a series of ordered steps)
> - Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources

- Generalizing and transferring this problem solving process to a wide variety of problems.

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of Computational Thinking. These dispositions or attitudes include:

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution"

(http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf)

If it is possible to develop these competencies through the teaching and learning of Computational Thinking, this has obvious immediate benefits for children as well as laying a strong foundation for further development of both Computing and the broader key skills already identified as part of the second-level cycles where working with digital technology is an explicit part (Key Skills of the Junior Cycle, 2012).

1.2.3 The rationale for Computational Thinking at primary level

Most research in the development of Computational Thinking has been conducted with learners at second and third levels of education rather than primary. Indeed there are some (Waite, 2018) that argue that Computational Thinking requires dealing with levels of abstraction that are simply not compatible with the cognitive development of primary age children. These arguments generally rely on Piaget's early work (Piaget, 1953) which argues that it is only when children reach the 'formal operational' level of development, around the age of 12, that they gain the ability to think in an abstract manner. This assertion has been challenged, by Piaget himself in his later work but also, in this domain by Syslo & Kwiatkowska (2014) who argue that paying attention to progression by introducing concrete objects and examples before moving on to more abstract Computational Thinking about these objects and concepts may be effective. Gibson (2012) uses a similar approach, teaching abstract ideas to younger children through the use of concrete games and puzzles. Gibson reports inviting children to draw simple diagrams with circles and lines according to spoken rules and then to critically reflect on the attributes of these diagrams, leading to heated argument and sophisticated reasoning.

It can also be argued that computers make abstractions concrete as can be seen in the way that program code can be created in the Scratch programming language by simply dragging jigsaw pieces and locking them together in sequence. This makes it necessary to review the assertions regarding abstraction.

If we accept that it is possible to introduce Computational Thinking at primary school level, then the next question needs to be whether it is desirable to do so. There are a number of pragmatic reasons to start at this level which include:

1. **cross-curricular approaches** - Computational Thinking provides an opportunity for primary school teachers who teach a class across all subjects which is not available to second-level or third level educators. This is important, because research shows that transfer of Computational Thinking to learning in other subjects is not seen when taught as a separate topic, so beginning within other subjects can assure that it is effective in this sense.

2. **addressing issues of gender** - targeting girls at a younger age is found to be effective at increasing their participation in applying Computational Thinking in learning (Lapan, et al., 2000, Turner, et al., 2008, Graham and Latulipe, 2003)

3. **motivation, confidence, delight** - While Wing's early definition of Computational Thinking involved thinking "like a computer scientist", ideas of Computational Thinking have been broadened, and by introducing meaningful projects, character dispositions can be developed. Designing solutions with Computational Thinking, particularly if implemented with technology, can have value in providing fulfilment in learning itself, providing the delight of 'zest' through free choices and effective outcomes. Such zest is normally fostered through play, where children are meaningfully responsible for choices and outcomes, even when founded in imagination and fantasy. (Millwood, 2008)

4. **collaboration** - Computational Thinking projects lend themselves well to teamwork and indeed the competence particularly demands the capacity to consult, work and learn with others. This also has the capacity to lean on play and in particular the delight of 'conviviality' (Millwood, 2008).

5. **creativity** - Computational Thinking is best developed through creative project work, where the solutions designed use computing technology. Combining the arts, sciences and technology is currently considered to be highly desirable, permitting dynamic and interactive outcomes through making. New technologies, such as the BBC Microbit have made such creative projects well within the reach of a playful primary child.

1.2.4 Challenges with Computational Thinking

Significant challenges with transfer, progression, pedagogy and teacher readiness have been identified:

1. **Transfer -** The idea of Computational Thinking is to integrate computational techniques and approaches into all disciplines requiring problem-solving skills. Programming is an essential craft to practice to fully develop Computational Thinking, and has had many claims made about its effect on wider thinking and thus 'transfer' to other subjects. Feurzig, one of the authors of the programming language Logo, believed that among other things, learning to program:

    a) promoted rigorous thinking and expression through its need for statement precision;

    b) gave insights into key concepts such as variables and functions as it made them less abstract;

    c) provided concrete models for heuristic[2] thinking by, for examples, demonstrating the benefits of debugging as a learning process;

    d) demonstrated the possibilities of extending a problem to a larger domain or 'generalisation';

    e) exposed the importance of *process* in the area of problem-solving, and

    f) encouraged an experimental approach.

    (Feurzeig et al., 1969)

    However, as early as 1984 researchers were concerned about a lack of evidence for such claims. Studies found more social and motivational merit in learning to program rather than effects on wider cognitive development. Unfortunately, interest in this research area waned in the 1990s and had flattened out almost completely by the early 2000s. The result of this has been that the resurgence of interest in the benefits of teaching Computational Thinking has been stymied by the lack of evidence-based research to provide guidance.

[2] A practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals - heuristic thinking relies on craft - rules of thumb, trial and error, estimates.

1. **Continuity, progression and pedagogy -** Similar problems surround the identification of what should be learnt when and in what order regarding Computational Thinking. This is in spite of there being found considerable consistency in the curriculum defined in international jurisdictions. Such consistency might demonstrate agreement, but is not necessarily evidence based. There is a lack of clear evidence to guide the best way to introduce Computational Thinking. In discussions with experts from the UK, it is clear that the simple ideas presented in 'unplugged'[3] activities are rarely followed up with meaningful connection to work at the computer when teaching programming (Waite, 2018). There is a real danger that Computational Thinking becomes a label for a set of enjoyable time-filling activities with little coherence and a lack of continuity and progression.

2. **Teacher readiness and research evidence -** These problems are compounded and sustained by a lack of readiness on the part of Primary teachers and the barriers in place to uptake of research evidence to inform practice. Primary teachers in Ireland are rarely qualified in Computer Science, since this is not a Teaching Council subject discipline, and in any case there are other more valuable careers on offer to those who have Computer Science. The barriers to uptake of research evidence range from teachers' personal beliefs, experiences and instincts to the external pressures of inspection, observation and tradition.

1.2.5 Conclusion

Despite these problems we would argue that we have a responsibility to equip children and teachers alike with competence in Computational Thinking, in view of the pervasiveness of technology in their lives. Children and teachers are no longer being first introduced to technology at school: it is more likely discovered and is widely used at home and leisure. Thus there is a need to ensure that children and teachers can become discerning users, as well as encouraging some to be future designers and developers of technology. For teachers to be role models for children, Computational Thinking needs to become part of their identity and practice, not simply some material to be taught, as might be observed with music teachers.

[3] Unplugged activities are those designed to investigate Computational Thinking without using technology, but instead using the more traditional paper & pencil or even human bodies and classroom environment.

2. What are the key teachable aspects?

2.1 Examples of concepts and skills

In order to fully explore the purpose of computational thinking and how it might be embedded in a Primary curriculum, it's important to clarify the constituent parts: the key concepts, skills and dispositions which are involved in Computational Thinking. An analysis of the English Barefoot Computing curriculum, the Scottish approach and a breakdown of the author's is presented here.

2.1.1 The Barefoot Computing curriculum

The most relevant analysis here is that to be found in Barefoot Computing, the English primary phase resource developed by Computing at School (Berry et al, 2018). This section breaks the Barefoot structure down further using the lens of knowledge, craft and character as described in section 1 (above). Loosely, 'concepts' maps onto 'knowledge' and 'skills' onto 'craft', but also added here is 'character' the dispositional aspects that lead to successful Computational Thinking. Barefoot define six 'concepts' and five 'approaches to working', which are analysed in tables 2.1 and 2.2 in terms of knowledge, craft and character:

Table 2.1 Analysis of Barefoot Computing 'concepts' (Berry et al, 2018)

Six Barefoot 'concepts'	**Knowledge** facts, mental-models & strategies	**Craft** skills through practice	**Character** emotions, attitudes, values
Logic - predicting & analysing	**Mental-models** of problem, programming language and notional machine[4], **strategies** of predicting, explaining and reasoning (analysing seems to be a misnomer for this)	**Evaluating** for correctness	**Imagining** consequences, **caring** about precision
Algorithms - making steps & rules	**Mental-models** of programming language and notional machine, **strategies** of decomposition and sequencing	**Creating** algorithms	**Imagining** consequences, **persevering** to correct errors

[4] The term 'notional machine' refers to the child's imagination of how a mechanism works. It is used, tacitly in most cases, to form explanations of how a mechanism has worked and predictions of what it might do in future circumstances. For example, when children play with the BeeBot (a programmable toy robot), they will imagine a 'notional machine' in the meaning of the steps they program, how they are stored and then executed when they press the 'Go' button.

Decomposition - breaking down into parts	**Strategies** of identifying parts and relationships	**Organising** information, **creating** representations of systems in diagrams	**Imagining** categorisations and causal & dynamic effects, **persevering** to revise, **caring** about precision
Patterns - spotting & using similarities	**Facts** of attributes, **strategies** of categorisation	**Organising** information	
Abstraction - removing unnecessary detail	This seems unconvincing, both in Barefoot's presentation and in Wing's which Barefoot depends on. Abstraction could be argued to be more positively about generalisation and representation e.g. using variables for sets of numbers		
Evaluation - making judgement	**Strategies** to identify efficiency, correctness, applicability	**Evaluating**	**Caring** about quality

Table 2.2 Analysis of Barefoot Computing 'approaches' (Berry et al, 2018)

Five Barefoot 'approaches'	**Knowledge** facts, mental-models & strategies	**Craft** skills through practice	**Character** emotions, attitudes, values
Tinkering - trying things out	**Strategies** for reasoning, creating and evaluating alternatives, **mental-models** of criteria e.g. effectiveness	**Creating** systems and **playing** with 'variables'	**Imagining** alternatives, **persevering** with choices
Creating - planning, making and evaluating things	**Facts** and **mental-models** about microworlds, problems, tools and languages	**Creating** designs, **making** artefacts, **evaluating** them	**Imagining** possibilities, **caring** about quality
Debugging - finding and fixing 'bugs'	**Strategies** for reasoning, **mental-models** of programming language and notional machine	**Creating** tests, **evaluating** program outputs and statements	
Persevering - never giving up, being determined, resilient and tenacious			**Persevering**
Collaborating - working with others to ensure the best result			**Collaborating**

2.1.2 The Scottish 3-15 Curriculum

In Scotland, the new 3-15 curriculum for Computer Science makes for greater focus on machines and languages: “Efforts to make Computer Science entirely about ‘computational thinking’, in the absence of ‘computers’, are mistaken, in our opinion.” (Cutts, Connor and Robertson, 2017). Nevertheless, the advice is to follow three ‘Significant Aspects of Learning’ (SALs) in sequence, but in a spiral curriculum sense (Farrell *et al.*, 2017).

The three SALs are:

1. **understanding the world through computational thinking** - Theory: Understanding the world through computational thinking and knowledge of core Computing science concepts is necessary in order to later apply that knowledge using languages and technology
2. **understanding and analysing computing technology** - Languages and Tools: Understanding of Computing technology and the programming languages that control them is essential before designing and building using these tools
3. **designing, building and testing computing solutions** - Creating: Use conceptual and technological knowledge to design, build and test.

The third SAL is where actual programming takes place, having explored theoretical meanings in SALs one and two first. This follows a traditional model of learning, which perhaps does not sit so well alongside a modern constructivist/constructionist paradigm.

2.1.3 Our approach

In this section the bigger picture of Computational Thinking is described and its relationship with the subject of Computer Science. Computer Science is not directly relevant to Primary children. It is concerned with the study of the theory, methods and effectiveness of computer systems and their applications. Computational Thinking is more concerned with overarching conceptualisation. For example, in Computer Science learners would investigate and prove the relative efficiency of alternative algorithms to sort lists of information, whereas competence in Computational Thinking would demand an awareness of this possibility, and apply that awareness in the design of a solution which needed to sort data to achieve a meaningful goal. Figure 2.1 lists some of the elements of knowledge (areas of conceptual knowledge in the form of facts, mental models and strategies) that inform Computational Thinking as a whole. These elements of knowledge will be only be understood

fully through a child's learning journey from pre-school to third level and beyond, in the sense of a spiral curriculum (Bruner, 1977). Nevertheless, some content can be addressed in simple and introductory ways at Primary level.

So this paper does not propose that all of Fig 2.1 below is to be tackled in the Primary phase, but that curriculum development should use it as guide to distinguishing Computational Thinking and Computer Science and for considering relevant ideas to be taught at a foundational, creative and playful level. Furthermore, the diagram in 2.1 does not relate to craft and character, and so demands further analysis to clarify what aspects of each topic should be expected to be practised, what character dispositions should be developed and what is to be known at the end of Primary phase.

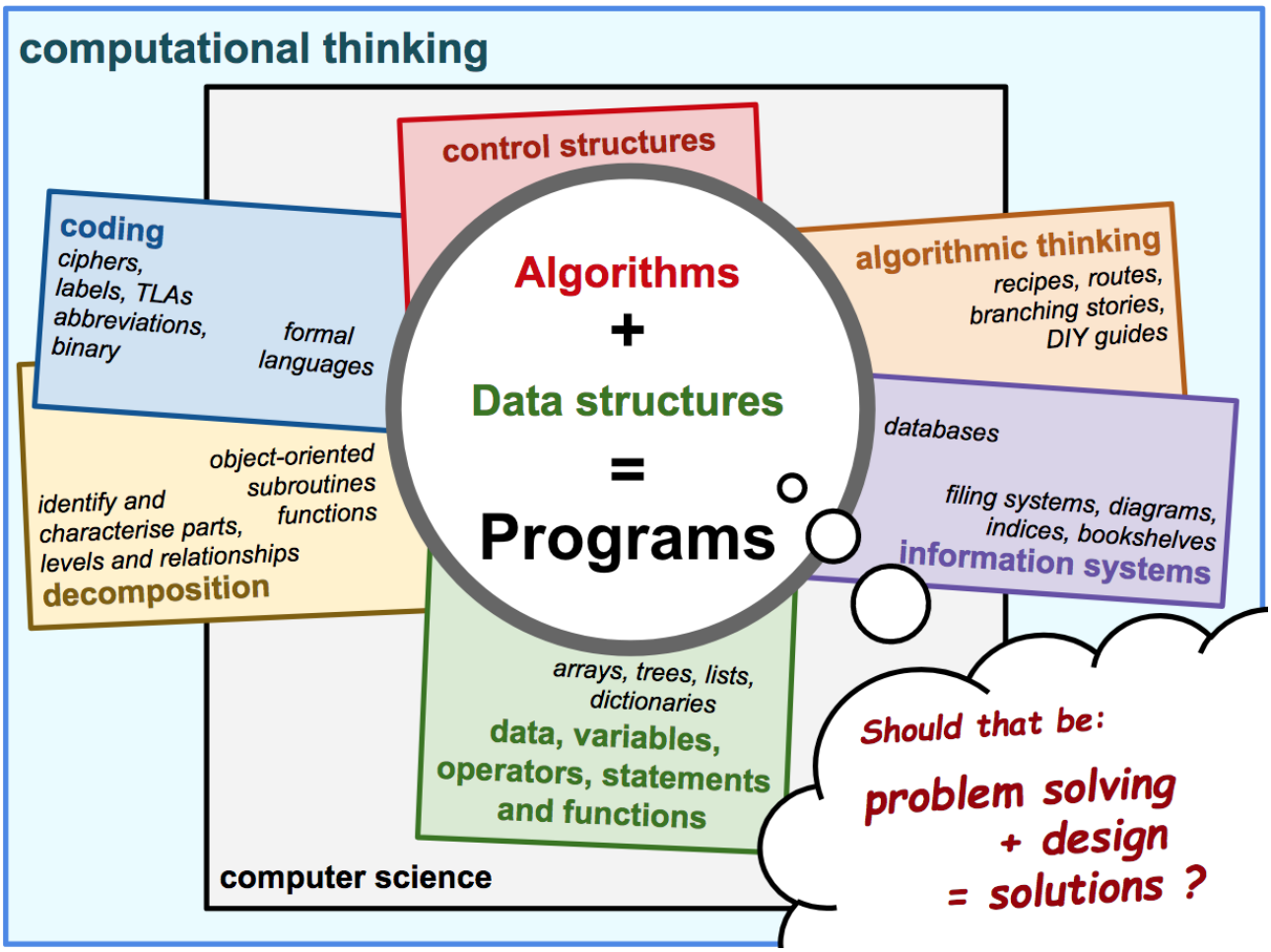


Fig 2.1 - The content of Computational Thinking (Millwood, 2016)

Fig 2.1 indicates how many of these knowledge elements (those in the outer blue rectangle) are more generic and everyday than Computer Science itself (those in the inner grey rectangle). For example, in information systems, the ways in which we organise information in filing systems or show in diagrams

on paper is an older topic and conceptually independent of Computer Science. But with the advent of widespread computing power new possibilities emerge for problem solving and design. An example from mathematics can be found in the way that with greater number-crunching power, numerical methods of problem solving by repeated calculations have supplanted methods using algebra. Another is the way that it is now most effective to develop a chart or graph using a spreadsheet package rather than drawing with paper and pencil. The has led to the growth of new and dynamic visualisations of data. Such examples are highly relevant to communicating complex ideas in many disciplines, but depend on a foundation of knowledge and craft that can begin in Primary level with understanding the power to communicate with much simpler examples. Paper and pencil methods are not helpful, since they typically mean simply following procedures - the power of computing allows a trial-and-error process more suited to problem solving and design, leading to a deeper conceptual understanding.

2.2 Progression

In this section, the idea of progression is examined through the kinds of activities proposed to develop Computational Thinking and the role of the computer in learning, taking into account the potential for playful and bodily experience and the new dimension that working with a computer offers. Broadly, it is anticipated that a spiral curriculum (Bruner 1960) approach is begun in the Primary stage, moving from direct concrete play and problem solving to apparently more abstract activity using the computer to design solutions. The vital insight is that the computer makes the abstract concrete in the sense of providing responsive, tactile and aesthetically pleasing experiences, thus supporting independent learning at the same time as being the embodiment of the content to be learnt.

2.2.1 Practical and meaningful projects

Computational Thinking depends on developing crafts of user-centred design[5], data analysis and programming through developing skills in practice (hands). This means children must engage in practical projects that attempt to create solutions for meaningful problems. Such problems do not

[5] 'user-centred design' means consciously using empathy and consultation with the anticipated users of a design. It is consider best practice for developing effective and pleasing design products.

need to be meaningful in the sense of real-world relevance - playful, imaginative and fantastical contexts can provide them too. For example, at an early stage children can solve play problems related to shopping or travelling from home to school using a floor map. They may begin by navigating physical space with their own bodies and subsequently by programming toy robots. At a later level embedded systems can be used to design solutions to robotics problems around the creation of toys and playthings as seen in the Robot Petting Zoo (*TechHive - At the Lawrence Hall of Science*, 2018). Robots here may be a dinosaur that opens its mouth when presented with food - much more than simple engineering solutions, but toys which are founded in play and fantasy, unleashing the imagination to create playful artefacts.

2.2.2 'Unplugged' and 'plugged'

Learners may begin with 'unplugged' activities, designing solutions to problems that exercise algorithms and data structures applied to themselves and their surroundings. A common example is to invite children to play as robots, speaking or writing instructions for each other to navigate space. But the link must be made in terms of knowledge and craft as they progress to using technology. Essential character development includes dispositions of empathy, inquiry, imagination, perseverance and concern for quality (heart). These can be explicitly addressed through reflection on the way learners have dealt with the challenges set.

Key strategies of problem solving as described in the Barefoot Computing approach can be developed over time by remembering to start with modest aims. For example 'decomposition' can be to identify first and second steps in a sequence - "this comes before that". Progression to longer sequences can be attempted when this simplest of decompositions has been mastered.

Knowledge required includes the way in which information systems and computer technology work, are programmed and may be debugged through facts, mental models and strategies (head). Four essential mental models that must be developed for programming are shown in Fig 2.2.

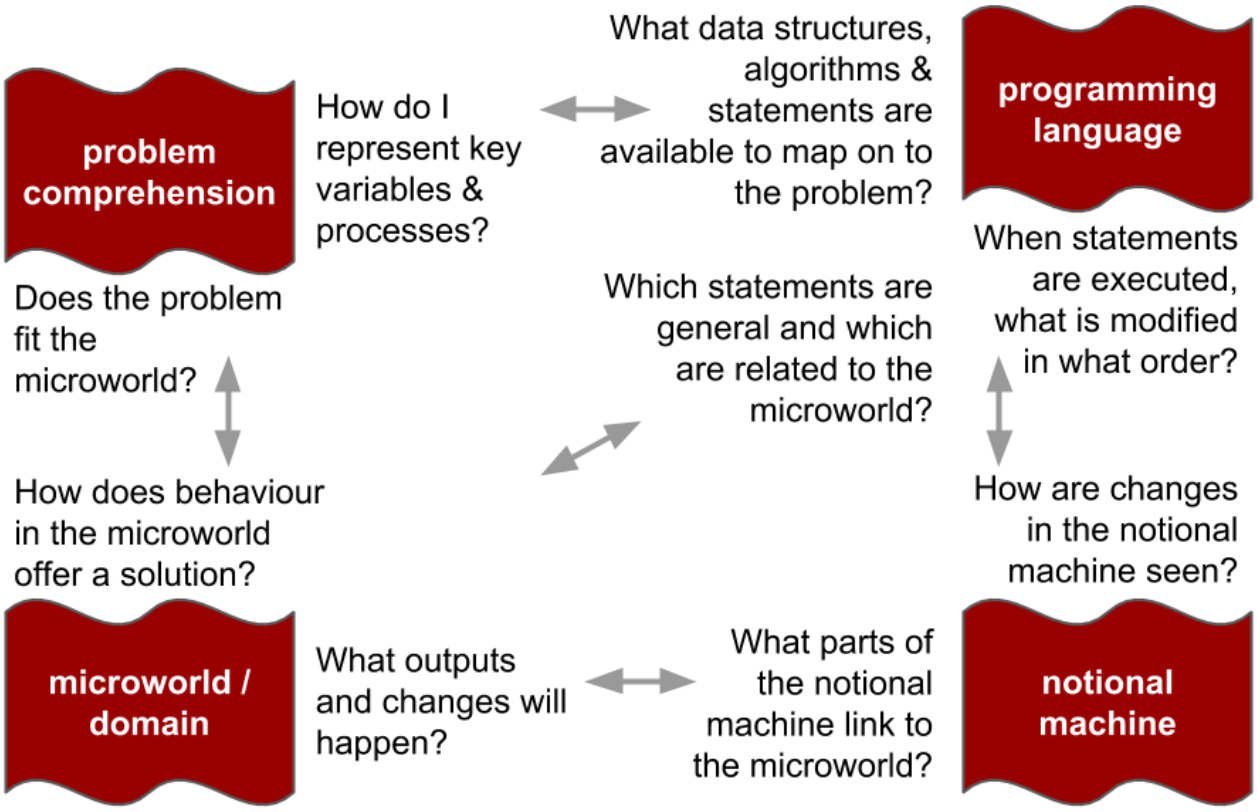


Fig 2.2 Programming: Four areas of mental model

For example, a simple project for children is to animate on a tablet the telling of a Knock-knock joke. It is important to develop problem comprehension, by inviting children to recall such jokes, select one and write down the script as a play. If they are then to write a computer program, they must develop an idea of the program language elements - in Scratch, these are found categorised by colour on the screen and their 'grammar' is discovered through dragging and dropping like jigsaw pieces onto the screen. Furthermore, they must build a mental model of the notional machine - how the language is executed in the computer. Scratch helps by highlighting which statement is active as the program is run. Finally they must also know the features of the microworld or domain on which the program is acting - with Scratch, this includes the idea of a stage and sprites that can move, interact, play sounds and change shape.

Ultimately, designing data and algorithms to create programs through coding is an essential skill, which is attainable by Primary learners aided by sophisticated but easy to begin programming

languages such as ScratchJr, Scratch and Snap. These languages can be revisited with increasing levels of sophistication as the learner develops.

2.2.3 Why computing is important

An important potential strength through working with the computer is that its interface offers an additional learning cycle of expression and evaluation as described in the expressive constructivist model (Millwood, 2014) - see Fig 2.3.

Fig 2.3 - The learning model of Expressive Constructivism (Millwood, 2014)

An example from three children exploring a computer game can be found in the script below. In this example Sasha is expressing his thinking about the way the game works explicitly using natural language to the others. After several turns through a loop re-expressing on the basis of his own evaluation by listening to his own words (although seeking the others' evaluation), eventually his brother evaluates his words.

> Patrick - I'll die if I go down there!
>
> Sasha - Like getting damaged. Getting all the way damaged do you mean? Getting damaged.

Sasha - Do you know when you die? You die when you get all damaged, is what it means, when it all gets red or the green turns into red.

Sasha - The red is damage and the green isn't damage. Do you understand?

Sasha - When you get all damaged then you die, is that right Patrick?

Patrick - Yes, yes that's right.

The additional cycle of learning through trying in the game and seeing what happens had already occurred, as Sasha reports in this conversation:

Sasha - That's right, when you get down there you can just go from there and then straight down to there without a single damage.

Sasha - And do you know how I know that?

Sasha - Because I tried it several times - that's how I know.

Patrick - That's very good, thanks Sasha!

The independent learning cycle offered by the computer game itself is the powerful important potential strength proposed above.

Such potential is realised if the computer environment enhances expressivity and at the same time increases evaluative power, which may also come from the evaluation of peers, teachers or parents able to share the screen (Millwood, 2012). In programming, a formal expression is written, often using aesthetically pleasing tools such as Scratch, and the computer helps evaluate their validity by attempting to execute the expression. The visual results are often informative to the learner, especially as they develop debugging skills. A record of past expressions can be saved and reviewed and used as a basis to collaborate with other learners. Furthermore, tools can be used to analyse the complexity or sophistication of such expressions. Thus much independent learning can take place, although this can be maximised by working in pairs or small groups and through the facilitation and expertise of the teacher.

3. Computational Thinking in the primary curriculum

3.1 Integration

3.1.1 Embedding for transfer

Computational Thinking can be introduced and developed in all subjects, and indeed is already present in Primary practice, and so embedding Computational Thinking is also about recognition, not simply innovation. As described in section 2, evidence suggests that teaching programming in isolation does not lead to transfer of competence. Using Computational Thinking to support challenges across the curriculum should therefore be the starting point, in order to embed it as a transferable competence.

Computational Thinking depends on acquiring crafts of user-centred design, data analysis and programming through developing skills in practice (hands). Essential character development includes dispositions of empathy, inquiry, imagination, perseverance and concern for quality (heart). Knowledge required includes the way in which information systems and computer technology work, are programmed and may be debugged through facts, mental models and strategies (head). At its heart, designing data and algorithms to create programs through coding is an essential skill, which is attainable by children in Primary school aided by sophisticated but easy to begin programming languages and other tools such as spreadsheets and databases. These tools can be revisited with increasing levels of sophistication as the learner develops.

3.1.2 Examples where it can be included

The following specific aspects of Computational Thinking from Fig 2.1 above were chosen:

- coding
- decomposition
- algorithmic thinking
- information systems

These aspects were explored against the strands and strand units of the Irish primary curriculum (NCCA, 2016) and the following examples of using Computational Thinking as an across-curriculum competency are presented in Table 3.1

Table 3.1 Examples of using Computational Thinking across the curriculum.

Computational Thinking	**Competence**	**Precursor**	**Curriculum focus**	**Activity**	**Progression**
Algorithmic Thinking - understand the concept of sequence	Explaining 'what do you see when you go on your journey home'. Predicting the success of a sequence of steps to go home.	Understand cause and effect between two events	**English** - 11. Retelling and elaborating Tell and retell stories and personal and procedural narratives of increasing complexity to familiar and unfamiliar audiences using appropriate sequencing, tense and oral vocabulary. TF11, C1+2	It's your birthday, draw a map/explain to your friend how to get from the school to your house and what steps/stages in the journey.	Send your beebot home: Draw a map of your locality and enter the commands necessary for the beebot to navigate the map.
Decomposition, Alorithmic Thinking - co-ordinating events in correct sequence	Analysing a narrative for the steps in a sequence, then formally making the steps that will represent its performance.	Understand the concept of a sequence.	**English** - as above	With a partner recall a knock knock joke. Write a play script outlining your knock knock joke.	Using Scratch write a set of commands between two characters that will result in the telling of a knock knock joke.
Coding, Decomposition - generalising structures	Analysing patterns in human and computer languages, creating new expressions	Know the elements of language and their names: verb, noun, adjective etc.	**Gaeilge -** . Sentence structure and grammar Use coherent sentences of increasing complexity with correct tense, word order and sentence structure, while using	Compare syntax of irish language (Action, person/object, FB, place, time) with syntax of Formal Coding languages.	Write program to generate nonsense sentences, but correctly grammatically structured.

			connectives and producing compound and complex sentences to elaborate appropriately. TF4, C2		
Algorithmic Thinking - identifying patterns	Making by hand, analysing and making an algorithm from patterns	Know appropriate colours, able to draw straight lines and fill in shapes with colour	**Art -** Paint/ drawing – Making drawings, looking and responding to drawings.	Follow an algorithm to draw an 'automated' landscape'	Using Scratch, engage in a paired programming activity to research and re-create a famous painting.
Decomposition- identfying key elements and relationships	Creating and making an interactive narrative	Know the story of a past event, its context and main actions	**History -** use imagination and evidence to reconstruct elements of the past	Research a historical event and context identifying the actions in sequence and the appearance of key elements	Develop a narrative using an interactive 3D environment to tell the story
Decomposition - identfying key elements and relationships	Analysing and explaining a system	Know concept of biological cells and their lifecycle	**Science** - Living things - Human life	Identify and characterise the 'actors' in the antibody reaction to infection	Create an antibody reaction simulation using Scratch
Information systems	Analysing, explaining and predicting information	Understand measurement of temperature, wind strength, direction	**Geography -** Investigating and experimenting • carry out simple investigations set by the teacher, make observations and collect data	Record weather data each day using a spreadsheet / database	Create reports of weather using charts, contrast with media predictions

The final column in the table suggests activities that might develop Computational Thinking further through the use of computing technology and which build on the concepts already tackled in the curriculum in other subjects,

but not labelled as Computational Thinking. It is argued that the greater depth of understanding through the use of technology will create a foundation for Computational Thinking through creative and playful activities.

4. How do we develop teachers' knowledge-base?

This section focuses on Continuing Professional Development for teachers in service. Clearly there is also scope for change in the preparation of teachers in initial teacher education. However, the conditions are different, with face-to-face predominant and greater opportunity to lay down a foundation rather than building on existing teachers' experience.

4.1 What competence is needed?

4.1.1 Pedagogical and content knowledge

Teachers aiming to to develop competence teaching Computational Thinking concepts, can benefit from engaging in reflection on their current practice and to identify existing alignment with Computational Thinking objectives. The Pedagogical Content Knowledge framework initially proposed by Shulman (1986) lends itself as an approach that teachers can engage with in order to develop a deeper understanding of how their current practice aligns with the teaching and learning of Computational Thinking. Shuman argues that in order for teachers to develop in depth understanding of teaching and learning they need to obtain expertise in pedagogical knowledge and content knowledge and furthermore develop an understanding of how and where both pedagogical knowledge and content knowledge align. The PCK framework and the more recent TPACK framework proposed by (Mishra and Koehler, 2006) both provide useful frameworks for teachers to explore combinations of technology knowledge, pedagogy knowledge and content knowledge and how and where they intersect for in-depth Computational Thinking teaching and learning experiences. In recent studies in Ireland at secondary level, teachers who participated in a Lesson Study[6] team identified deep relationships between their pedagogy knowledge and their content knowledge and an alignment between their pedagogy content knowledge during their evaluation of teaching and learning activities they had co--created and implemented in a formal setting (NCCA, 2016, Ní Shuilleabhain, 2016). The research suggests that the Lesson Study model provides a valuable methodology for engaging teachers in activities that aim to raise their awareness of how pedagogy and content knowledge align. Furthermore, the research suggests that teams of teachers could support one another to develop

[6] Lesson Study is an approach to professional development where teachers act in teams to address lesson planning and reflect collectively on outcomes

common practices for Computational Thinking. However, It is important to consider, that competences are “complex combinations of knowledge, skills, understanding, values and attitudes” (EC, 2013) *a*nd, as such, simple knowledge of the alignment between their existing practices and Computational Thinking may be insufficient contributors to teacher professional growth. Hence, an over-reliance on teachers only demonstrating knowledge of where and how Computational Thinking aligns with current practice will repeat many of the mistakes of past reforms.

4.1.2 Building on existing knowledge

In conjunction with a knowledge based approach, a ‘bottom up’ approach where teachers build on their existing ‘know how’ and their current experiences of teaching and learning can be of benefit to individuals willing to develop competence in Computational Thinking teaching and learning. It is argued that teachers’ prior experiences of teaching and learning are significant contributors to the epistemologies, beliefs and values they bring into the classroom and these experiences afford or construe the likelihood of teachers’ successes with achieving new reforms. The literature is full of examples of teachers, who, following professional development, engage in ‘gung ho’ attempts at implementing reforms before, gradually, retreating to traditional teaching and learning methodologies when external pressures, including parental expectations, pressures from colleagues and exam focus conflict with the reforms. If Computational Thinking is to take its place within the curriculum, then easily accessible resources or activities which enhance the teaching and learning of current curriculum learning outcomes would be of great benefit for teacher's practice, and where these resources align with teacher's existing knowledge-base of pedagogy and content knowledge they will, in theory, be more likely to be used.

4.2 What alternatives are there?

Since the introduction of Computational Thinking may be novel for many teachers, approaches to Continuing Professional Development should be diverse and adapted to fit different teachers’ experience, preference and life circumstances. Online collaborative approaches designed to fit within the Irish Cosan and Digital Strategy frameworks would be particularly appropriate to cater for rurally isolated teachers. Exploitation of national conferences, regional Teachmeets and existing Communities of Practice will ensure sustainability alongside more traditional approaches.

Accordingly, CPD experiences aimed at successful reform will require approaches tailored specifically for the needs of the teacher, which can be achieved through elearning initiatives guided by branching

logic for example, or the provision of a diverse range of activities which recognise and cater for professional development episodes which afford teacher professional growth. If we accept that for the most part, teacher growth is incremental and gradual, while recognising that teachers can experience gestalt shifts in practice, then it is likely that teachers will be prepared to engage in different types of CPD during different stages of growth.

Where teachers are novices or newcomers to topics such as Computational Thinking then traditional instructor directed activities will enable the development of initial knowledge about how and what Computational Thinking is. These examples may include facilitator driven presentations of what Computational Thinking is, how it can be integrated and examples of successful strategies which teachers can take away and use in their own classroom. These could also be supplemented with online modules where examples are delivered step by step similar to online course providers such as Udemy and Futurelearn. Where teachers have moved beyond a novice stage and are prepared to experiment with new teaching and learning strategies then attendance at events including Teachmeets, conferences or maker meets, for example would enable them to view examples and make connections with like-minded individuals engaging in professional experimentation with Computational Thinking initiatives. For those who are in a master stage of development, then joining or leading communities of practice may align with their stage of professional growth. In these cases, models of CPD which aim to engage teacher's as participants in professional learning communities will be of value. Existing models of CPD which align with this stage of development would include Lesson Study, Bridge 21 and Design Thinking with teachers. Each of these have been seen to be effective at second-level: Lesson Study in Mathematics, Bridge21 in Computer Science, Design Thinking in Information and Communications Technology. These models have also been employed in academic courses for Irish primary teachers such as the MSc Technology and Learning at TCD, and because they go beyond exposition of knowledge but also foster the development of the teacher's craft and character by involving the teacher in systematic professional dialogue, there would be good reason to believe they would be effective at primary level more generally.

Over time, collaboration between teachers can move from face-to-face meetings to the online environment. Particularly in rural Ireland, teachers are geographically dispersed inconveniently, meaning face-to-face meetings can require cumbersome journeys for participants, which may be interrupted by a range of life events, duties of care or adverse weather conditions. Consequently, the ultimate goal of CPD aimed as long term sustainable growth should consider the provision of online environments where teachers can engage in constructive dialogue, lesson prototyping and evaluation and reflection on professional experimentation with co-created teaching and learning approaches to

Computational Thinking. Here cluster models, such as those currently being proposed by the Digital Excellence fund initiative will be highly relevant.

When this reality can be achieved, then CPD will become both teacher-centred and self-directed. While it is important to recognise that there is no argument for teachers to pass through stages of professional growth in a linear fashion, the provision of a range of CPD experiences, such as those listed above, which can cater for teachers at different stages of professional growth can provide a user experience which aligns with the four dimensions for teacher learning outlined by the Cosán framework (2016).

1. Formal and informal
2. Personal and professional
3. Collaborative and individual
4. School based and external.

4.3 How can this be sustainable?

But for sustainability, it is crucial to consider the continuity and progression of the teacher's competence alongside the whole-school approach being taken. To this end it is recommended that school self-audit and planning is combined with individual teacher audit to identify strengths and weaknesses and guide strategic and personal planning.

Regardless of any CPD initiative, or learning process which is identified, the gatekeeper to any reform is the teacher themself. Teacher professional growth takes place on two planes, the psychological and the social and it is important to recognise that any teacher will require a range of both internal and external supports which will make it worthwhile for them to implement reform. Where internal factors including beliefs, attitudes, knowledge and skills will contribute to the competence of the teacher, the social domain must be negotiated in order for reforms to become sustainable. Traditionally, external resistance to reform comes from a range of factors including parental expectations, school culture and anxiety surrounding student performance on exam subjects. As it is, initially, unlikely to be able to prove the impact of Computational Thinking skills learning on students performance through traditional standardised testing approaches to assessment at primary level in particular, success will

have to be communicated to the wider school community through different means. Subsequently, a balanced strategy where both the individual teacher and the whole-school approaches are in synergy can contribute to reform initiatives which will be sustainable. Accordingly, the school eco-system will play a significant role in how successful any reform will be. While the teacher is the gate-keeper to any reform, as they ultimately afford or construe its implementation, the wider system, including parents, stakeholders, colleagues and students will all play a significant role in how sustainable any reform initiative is. Computational Thinking initiatives, therefore, can benefit from adopting a whole school approach, by school we mean the participants listed above, where the beliefs, attitudes, knowledge and skills of the school community are evaluated before formulating a strategic plan to achieve reforms. One such model includes, the Educational Positioning System (EPS) (Wenmoth, 2008) school evaluation process which focuses on the whole school community as a contributor to direction and vision. The EPS can help teachers and stakeholders to identify their communities needs in order to plan and measure successful reforms. The EPS examines how the whole school community, including parents, teachers, staff and stakeholders interact thereby providing an in depth picture of the socio-cultural environment of the school and what challenges there are likely to be to reforms. The EPS accomplishes this by measuring school progress against three key dimensions which include six elements each.

The three key dimensions include:

Philosophical frameworks.	Exploring the fundamental nature of educative purpose, learning, knowing and knowledge.
Community and culture.	Addressing the development of a learning culture and learning community.
Strategies and structures.	The tools to implement the philosophical frameworks including the design of, the use, and allocation of people, time, space and place.

(Wenmoth, 2008)

The value of this type of approach is that the beliefs attitudes, skills and knowledge of the individual teachers, the parents, students and staff can be collected, measured and ‘planted in the ground’ as reference points for evaluating the success of any reform initiative.

Conclusion

This report argues that Computational Thinking is the right focus in primary education and can and should be delivered through activities in every subject. Simpler frameworks can help teachers see the whole picture to develop competence in children. ‘Unplugged’ approaches are useful, but must be clearly linked with progression to ‘plugged’ activities. In both cases, playful and meaningful approaches should be used to maintain interest and zest in pupils. Professional Development approaches must be creative and collaborative as teachers develop their personal competence as well as understanding the pedagogical and content knowledge to be taught to children. This can best be started through linking self-audit and whole-school audit to recognise where Computational Thinking is already taught.

References

Anderson, L. W. et al. (2000) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Complete Edition. 2Rev Ed edition. New York: Pearson.

Barr, V. and Stephenson, C. (2011) 'Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?', ACM Inroads, 2(1), pp. 48–54. doi: 10.1145/1929887.1929905.

Bruner, J. S. (1977) *The Process of Education*. Harvard University Press.

Clarke, D. and Hollingsworth, H. (2002) 'Elaborating a model of teacher professional growth.', Teaching and teacher education, 18(8): 947-967.

Crick, T. (2017) FINAL DRAFT: Computing Education: An Overview of Research in the Field. Royal Society. Available at: https://royalsociety.org/~/media/policy/projects/Computing-education/literature-review-overview-research-field.pdf.

Cutts, Q., Robertson, J. and Connor, R. (2017) 'Keeping the machinery in Computing education', Communications of the ACM, 60(11), pp. 26–28. doi: 10.1145/3144174.

Department for Education (2014) National curriculum in England: Computing programmes of study, Publications - GOV.UK. Available at: https://www.gov.uk/government/publications/national-curriculum-in-england-Computing-programmes-of-study (Accessed: 28 April 2015).

Department of Jobs, Enterprise and Innovation (2014) ICT Skills Action Plan 2014-2018. Available at: https://www.education.ie/en/Publications/Policy-Reports/ICT-Skills-Action-Plan-2014-2018.pdf (Accessed: 1 December 2018).

Ertmer, P. A., et al. (2012) 'Teacher beliefs and technology integration practices: A critical relationship', Computers & Education, 59(2): 423-435.

Europeia, C. (2013) 'Supporting teacher competence development for better learning outcomes, Comissão Europeia.'

Farrell, K. et al. (2017) Teach Computing Science - A Guide for Early Years and Primary Practitioners, p. 70. Available at: http://teachcs.scot/wp-content/uploads/2017/05/TeachCS.pdf (Accessed: 14 February 2018).

Feaster, Y. et al. (2011) ‘Teaching CS unplugged in the high school (with limited success)’, in. Proceedings of the 16th annual joint conference on Innovation and technology in Computer Science education, ACM, pp. 248–252.

Gibson, J. P. (2012) ‘Teaching graph algorithms to children of all ages’, in. Proceedings of the 17th ACM annual conference on Innovation and technology in Computer Science education, ACM, pp. 34–39.

Graham, S. and Latulipe, C. (2003) ‘CS girls rock: sparking interest in Computer Science and debunking the stereotypes’, in. ACM SIGCSE Bulletin, ACM, pp. 322–326.

Grover, S. and Pea, R. (2017) ‘Computational Thinking: A Competency Whose Time Has Come’. Available at: https://www.researchgate.net/profile/Shuchi_Grover/publication/322104135_Computational_Thinking_A_Competency_Whose_Time_Has_Come/links/5a457813a6fdcce1971a5ce5/Computational-Thinking-A-Competency-Whose-Time-Has-Come.pdf.

Guskey, T. R. (2002) ‘Does it make a difference? Evaluating professional development’, Educational Leadership, 59(6): 45.

Guskey, T. R. (2014) ‘Planning professional learning’, Educational Leadership, 71(8): 10.

Lapan, R. T. et al. (2000) ‘Seventh graders’ vocational interest and efficacy expectation patterns’, Journal of Career Development, 26(3), pp. 215–229.

Livingstone, I. and Hope, A. (2011) Next gen: transforming the UK into the world’s leading talent hub for the video games and visual effects industries : a review. National Endowment for Science Technology and the Arts (Great Britain). Available at: http://www.nesta.org.uk/sites/default/files/next_gen_wv.pdf.

Mannila, L. et al. (2014) ‘Computational Thinking in K-9 Education’, in. Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference, ACM, pp. 1–29.

Mano, C., Allan, V. and Cooley, D. (2010) ‘Effective in-class activities for middle school outreach programs’, in. Frontiers in Education Conference (FIE), 2010 IEEE, IEEE, p. F2E–1.

Mayer, R. E., Dyck, J. L. and Vilberg, W. (1986) ‘Learning to program and learning to think: what’s the connection?’, Communications of the ACM, 29(7), pp. 605–610.

Millwood, R. (2008) ‘An Analysis of Delight’, Richard Millwood, 15 May. Available at: http://blog.richardmillwood.net/2008/05/15/an-analysis-of-delight/ (Accessed: 11 January 2018).

Millwood, R. (2012) ‘How does technology enhance learning?’, A New Learning Landscape, 30 June. Available at: http://blog.richardmillwood.net/2012/06/30/how-does-technology-enhance-learning/ (Accessed: 14 February 2018).

Millwood, R. (2014) The Design of Learner-centred, Technology-enhanced Education. University of Bolton. Available at: http://phd.richardmillwood.net/ (Accessed: 4 May 2016).

Millwood, R. (2018) ‘Competence = knowledge + craft + character’, Richard Millwood, 11 January. Available at: http://blog.richardmillwood.net/2018/01/11/competence-knowledge-craft-character/ (Accessed: 11 January 2018).

Mishra, P., M. J. K. (2006) ‘Technological pedagogical content knowledge: A framework for teacher knowledge’, Teachers college record, 108(6): 1017.

NCCA (2012) Key Skills of the Junior Cycle. National Council for Curriculum and Assessment. Available at: http://www.juniorcycle.ie/NCCA_JuniorCycle/media/NCCA/Documents/key_skills_oct_2012_WEB_FINAL.pdf (Accessed: 8 February 2018).

NCCA (2016) ‘Primary Language Curriculum: English Language 1 and Irish Language 2’. National Council for Curriculum and Assessment. Available at: http://curriculumonline.ie/getmedia/524b871d-1e20-461f-a28c-bbca5424112d/Primary-Language-Curriculum_1.pdf (Accessed: 1 March 2018).

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

Pea, R. D. and Kurland, D. M. (1984) ‘On the cognitive effects of learning computer programming’, New ideas in psychology, 2(2), pp. 137–168.

Piaget, J. (1953) The origin of intelligence in the child. New York: Routledge & Kegan Paul.

Priestly, M. (2016) A Perspective on Learning Outcomes in Curriculum And Assessment. Dublin, Ireland: National Council for Curriculum and Assessment of Ireland, p. 13. Available at: https://www.ncca.ie/media/2015/a-perspective-on-learning-outcomes-in-curriculum-and-assessment.pdf (Accessed: 11 January 2018).

Rodriguez, B., Rader, C. and Camp, T. (2016) ‘Using student performance to assess cs unplugged activities in a classroom environment’, in. Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ACM, pp. 95–100.

Shuilleabhain, A. N. (2016) ‘Developing mathematics teachers’ pedagogical content knowledge in lesson study’, International Journal for Lesson and Learning Studies, 5(3):, p. 212–226.

Shulman, L. S. (1986) ‘Those who understand: Knowledge growth in teaching’, Educational researcher, 15(2): 4-14.

Sysło, M. M. and Kwiatkowska, A. B. (2014) ‘Playing with Computing at a children’s university’, in. Proceedings of the 9th Workshop in Primary and Secondary Computing Education, ACM, pp. 104–107.

Taub, R., Ben-Ari, M. and Armoni, M. (2009) ‘The effect of CS unplugged on middle-school students’ views of CS’, ACM SIGCSE Bulletin, 41(3), pp. 99–103.

TechHive (2018) TechHive - At the Lawrence Hall of Science, TechHive - At the Lawrence Hall of Science. Available at: http://www.techhivestudio.org/ (Accessed: 11 February 2018).

Tedre, M. and Denning, P. J. (2016) ‘The Long Quest for Computational Thinking’, in Proceedings of the 16th Koli Calling International Conference on Computing Education Research. New York, NY, USA: ACM (Koli Calling ’16), pp. 120–129. doi: 10.1145/2999541.2999542.

The Royal Society (2107) After the reboot: computing education in UK schools. The Royal Society, p. 60. Available at: royalsociety.org/computing-education (Accessed: 22 February 2018).

Turner, S. et al. (2008) ‘Gender differences in Holland vocational personality types: Implications for school counselors’, Professional School Counseling, 11(5), pp. 317–326.

Waite, J. (2017) Pedagogy in teaching Computer Science in schools: A Literature Review. Royal Society. Available at: https://royalsociety.org/~/media/policy/projects/Computing-education/literature-review-pedagogy-in-teaching.pdf.

Waite, J. (2018) ‘Primary computational thinking - experience in the UK’.

Wenmoth, D. (2008) ‘CORE Educational Positioning System 2.0’, 24 October. Available at: https://www.slideshare.net/dwenmoth/eps2-presentation (Accessed: 22 February 2018).

Wilson, C. et al. (2010) ‘Running on empty: The failure to teach K-12 Computer Science in the digital age. Association for Computing Machinery’, Computer Science Teachers Association.

Wing, J. M. (2006) ‘Computational Thinking’, Comm

Appendix 1: The development of the report

The authors of this report formed a team under the supervision of Dr Richard Millwood, who took overall responsibility for content and editing. Other experts also provided input, all of whom are acknowledged here.

Dr Richard Millwood is Visiting Research Fellow in the School of Computer Science and Statistics in Trinity College Dublin. He is currently facilitating a new community of practice for teachers at all levels in Ireland called CESI•CS which is addressing the need for mutual support regarding the introduction of computing in schools. Richard has been working for the last five years in Dublin to direct the Masters in Technology and Learning supervising teachers' own research at all levels. His doctorate was awarded in "The Design of Learner-centred Technology-enhanced Education". Apart from overall direction of this report, Richard contributed the sections on definition, concepts & skills and on progression. He also presented the report to the NCCA Board for Early Years & Primary and to the NCCA Council.

Nina Bresnihan is an Assistant Professor in the School of Computer Science and Statistics in Trinity College Dublin. Her doctoral study is in computational thinking and she is Principal Investigator on the SFI-funded OurKidsCode project which is developing workshops for families to bring parents and children together in creative use of computers. Nina has lectured for many years on the Masters in Technology and Learning and supervised teachers' own research. Nina worked on much of the section justifying Computational Thinking and its rationale.

Dermot Walsh is an acting Principal in Roundfort School in Co. Mayo. His doctoral study is in professional development and he has been recently awarded grants from the Teaching Council's Research Support Framework and from the Digital Strategy for Schools. His background has been in Primary teaching and thus contributed much of the work on curriculum and professional development.

Joy Hooper is a consultant who has a recent background in working for the UK's Joint Information Systems Committee (Jisc) and formerly the Education Ministry in New Zealand in the area of technology enhanced learning. Her experience as a Primary School teacher informed her overview of the work and her vital contribution was to manage the work, proof read and bring coherence to the report.

Glenn Strong, Assistant Professor in the School of Computer Science and Statistics in Trinity College Dublin and Dr Stephen Powell of the Centre for Excellence in Teaching & Learning at Manchester Metropolitan University provided review and feedback as the work developed.

In developing the report, the team held two conference calls with experts from UK organisations:

Dave Smith and Amanda Jackson form the Computing and Online Safety advisory team at HES - Havering Education Services in London, U.K. Dave and Amanda helped to develop the Rising Stars' Switched on Computing curriculum. Their knowledge and experience in curriculum development and implementation of professional development gave us many helpful ideas.

Jane Waite, who is London Regional Project Manager of the Computing at School organisation for teachers of Computing and also a learning resource developer in the Barefoot Computing team, who have created a computing curriculum for Primary schools in England. Jane has worked for over a decade as a Primary teacher, following twenty years working in computing for large organisations. Her recent work has been to help develop Barefoot Computing, resources for teachers delivering the Computing curriculum in Primary schools in England. Jane's doctoral study investigating the continuity and progression in the computing curriculum and her honest critique of the state of play in England was a vital contribution to our report.

Tony Riley and Arlene Forster at the NCCA provided continual support, encouragement and critique as the work developed.