COMPUTING IN SPECIAL EDUCATIONAL NEEDS & DISABILITIES SETTINGS: The Current Picture in England 2018

Catherine Elliott
Sheffield eLearning Service

Contents

Original version published at http://sheffieldclc.net/sendcomputing/ by Catherine Elliott, 5/9/18

1. Executive Summary

Computing replaced ICT as a subject in the National Curriculum in England in 2014 to better equip young people for living and working in an increasingly digital world. Although there is a greater degree of flexibility in how the curriculum is taught in special schools and similar settings, for example integrated resources and hospital schools, they still have a duty to deliver this new subject. This report aims to discover the current picture of computing provision in settings where students have special educational needs and disabilities (SEND), the confidence of teachers in delivering the subject and the perceived relevance of computing to their learners. It also collates the resources and strategies that are considered successful in teaching the subject to young people with special educational needs and disabilities.

A national survey of teachers in special schools and other SEND settings in England was conducted in the autumn of 2017, and resulted in 68 responses representing approximately 6% of special schools. The majority of respondents are computing or ICT lead teachers. On the whole the responses paint a positive picture of the subject, with confidence among teachers high and indications of a great deal of innovative work being done to adapt the curriculum for the specific needs of the young people in relevant ways. It must be noted, however, that a significant proportion of schools surveyed are still teaching ICT, and there are a number of challenges facing teachers in delivering the curriculum, many of which are common to teachers in mainstream settings. The largest barrier, however, is the lack of SEND specific resources, in terms of teaching materials, hardware and software. The majority of content is aimed at too high a level, and is often not age-appropriate for students working well below age expectations.

A number of resources were identified by respondents as successful in supporting the teaching of computing to SEND learners. This report contains a list in order to inform teachers as to what is available and works well with particular students. It must however be recognised that one student with SEND can have very different learning needs from another, and not all resources will be suitable for all learners. In addition a number of pedagogical approaches were identified by the teachers as beneficial when teaching computer science around the following themes: unplugged activities; relevant personalised tasks; physical computing; and effective chunking of tasks into smaller parts. Further research is required into the effectiveness of these approaches for particular groups of students.

A final aim for the survey was to investigate the teachers' perspective on the relevance of the computing curriculum in their setting. The responses were overwhelmingly positive about the importance of some form of computing or ICT for special needs learners. This is unsurprising, in an increasingly technology-rich world where many learners rely on technology for communication and access to learning. Teachers were also asked about the relevance of teaching programming and computational thinking skills to their learners. The teachers were split in their opinions, with many teaching in settings where the cognitive level of the students precluded meaningful programming activities. Interestingly, the more experienced and confident computing teachers consider these aspects more relevant and they have been using computational thinking as a framework for teaching problem-solving across the curriculum. There is much debate as to the transferability of computational thinking skills, but the responses indicate a number of benefits of this approach to learners with specific communication and learning disabilities and further investigation would be recommended.

2. Introduction

Computing was introduced into the National Curriculum in England in September 2014. In the National Curriculum Purpose of Study it states that a "high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world" (Department for Education, 2013). A priority for young people with special educational needs and disabilities (SEND) is to be able to influence and access the world around them, and technology is often a key part of this relationship. We should also be aspiring where possible to provide SEND learners with the same "economic and social mobility opportunities as their peers." (Wille, Century & Pike, 2017) The flipside of enabling more young people with special needs and disabilities to enter a career in a digital industry is the benefit of a more diverse workforce who can look at problems from a different perspective. (ibid.) So what is the state of computing education in special schools around the country, and how relevant is it to the priorities of SEND learners?

A number of surveys have been undertaken to establish the position of computing and progress made in mainstream schools since 2014[1], but there has been little focus on how successful the implementation of computing has been in special schools and similar settings, such as integrated resources and hospital schools. This report aims to paint a broad picture of computing delivery for students with SEND in England and it hopes to answer the following questions:

1. To what extent is the 2014 computing curriculum being taught in these settings?
2. How confident are teachers in delivering computing?
3. How relevant is the 2014 computing curriculum to learners with SEND?
4. What are the barriers to teaching computing in these settings?

The report also investigates which teaching strategies and resources work well in terms of teaching computing, with a particular emphasis on programming and computational thinking as this is the area where teachers often have less experience. The evidence gathered is based on teacher observation and experiences at this point, but it indicates areas for future action research. The report also signposts existing resources, not only for teachers in special schools, but to help teachers in other settings support their students working below age expectations or with specific needs and disabilities. As of January 2017 14.4% of pupils in England have special educational needs. The vast majority of these young people are currently educated in mainstream settings. (DfE, 2017)

One of the recommendations from the Royal Society Report, *After the Reboot,* is for "the effective sharing of knowledge between researchers, teachers and teacher trainers." (Royal Society, 2017) This report aims to be a working document for schools and teachers, to inform their practice and understanding, and also as a starting point to encourage the sharing of best-practice more widely.

3. The Survey

A survey was carried out between September and December 2017, advertised through targeted emails, social media, on the Computing At School community forums,[2] and by word of mouth. It was specifically aimed at teachers of computing in special schools or similar settings (for example inclusion centres, integrated resources, hospital schools) in England, as it is concerned with the English national curriculum subject of computing. During this time, 68 teachers completed the survey. Of these, one is an advisory teacher working for a local

[1] For example see Sentence and Csizmadia, 2016 and the teacher survey in the Royal Society report, 2017

[2] https://www.computingatschool.org.uk/

authority, and one teacher works in a school in Wales. His views were useful to gather in regard to the Digital Competency Framework (Learning Wales, 2016) and how that affects the teaching of computational thinking across the curriculum. The remaining 66 teachers represent approximately 6% of special schools in England.[3]
A smaller number of respondents were then contacted and took part in a short interview to gain more detail on certain areas. Transcripts of these interviews can be found in the appendix. These participants were selected as teachers of differing groups of special needs students who had provided examples of a number of strategies and resources for teaching computing in their setting.

3.1 Schools

One of the challenges of this piece of work is presented by the wide range of provision for students with special educational needs and disabilities, not to mention the huge spectrum of specific needs catered for. In a number of settings students are working well below age expectations and they will access a curriculum that is heavily adapted to their needs. In other settings, for example where there is predominantly a physical disability rather than learning difficulty, the students study a similar curriculum and qualifications to their peers. For an example see the interview in the appendix with Jonathan Fogg, who teaches students with visual impairment. Many of the strategies and resources he uses are identical to mainstream computing classes, and at a similar level, but will be used with specific adaptations such as using a screen reader or with audio support.

Future studies may benefit from narrowing the focus to a particular cohort of students to best identify what works well for their specific needs, although it is also important to recognise that "(e)ven among students with the same diagnosed specific disorder or subdisorder, the characteristics of the disorder vary and present in different ways" (Wille, Century & Pike, 2017). In this report, recommendations are made for approaches that may benefit students with learning difficulties and autism in particular, but the teacher should always consider the needs of the individual in adapting them.

Below is a summary of the types of settings where respondents work.

Age range of pupils in settings:

10 respondents teach in primary settings (4-11 year olds).
26 respondents teach in secondary settings (11-16+).
27 respondents teach in settings with primary and secondary age pupils.
2 teach in a setting with only post 16 students.[4]
2 did not specify.

Special Educational Needs and Disabilities:

Respondents were asked about the specific needs and disabilities of the students they taught, to gain an idea of how relevant the subject is for these young people, and the particular strategies and resources that may be beneficial to different groups.

All respondents named at least 2 from the list below, with the largest number (91%) teaching students on the autistic spectrum, followed by those with Behavioural, Emotional and Social Difficulties (BESD).

[3] 1,039 state funded and non-maintained special schools in England in 2016, https://www.gov.uk/government/statistics/special-educational-needs-in-england-january-2016

[4] Post 16 institutions are included as they can contribute to the collection of effective strategies and resources.

Special educational need or disability of students taught	**% of respondents**
Profound and Multiple Learning Difficulties (PMLD)	56%
Severe Learning Difficulties (SLD)	71%
Moderate Learning Difficulties (MLD)	66%
Behavioural, Emotional and Social Difficulties (BESD)	78%
Autistic Spectrum Disorder (ASD)	91%
Specific Communication Difficulties (SCD)	56%
Visual Impairment (VI)	54%
Hearing Impairment (HI)	56%
Other	3%

4. Findings

4.1 Who filled in the survey?

Respondents were asked about their role in school. 49 respondents to the survey (72%) are the Computing co-ordinator or ICT lead in their school (they may also hold another role in school too). Of the remaining, 5 are members of SLT, 12 are class teachers, 1 is a HLTA and 2 hold another unspecified role. Just over half (52%) of the teachers are members of Computing at School.

Ideally all respondents would have been the ICT/Computing lead, as they would have a complete picture of provision in their school. However it was important not to exclude schools where there is no specific teacher in charge of the subject, and it also provides a partial insight into the confidence of non-specialists.

4.2 Teacher Confidence

Respondents were asked how confident they felt teaching the Computing curriculum, on a Likert scale of 1 (*not at all*) to 5 (*very*). 64% rated themselves at a 4 or 5, which paints a positive picture. Only 14% put a 1 or 2. It must be noted that the willingness to complete a survey on computing may indicate a particular interest in the subject. This in turn may translate to increased confidence, and therefore this group of 68 is not necessarily indicative of the population as a whole. It is also unsurprising that the lead teachers of computing had a higher confidence level (76% rated themselves as a 4 or 5) than the remaining teachers (only 32% of these rated themselves at 4 or 5).

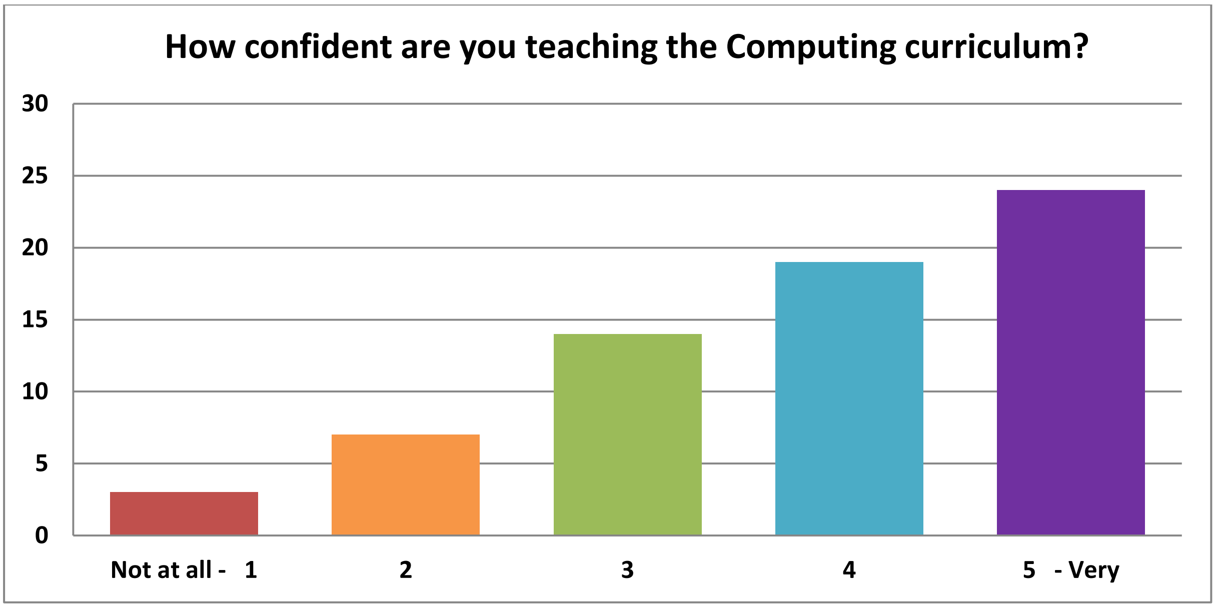


Table 1: Teacher Confidence

These results follow a similar pattern to the confidence felt by teachers in primary and secondary schools in the survey conducted by Sentance and Csizmadia (2016), with 61% rating themselves at 7 or above out of a scale of 10. The Royal Society report, After the Reboot, (Royal Society, 2017) survey found that the mean score for confidence out of 10 for primary teachers was 7.35, and 7.45 for secondary teachers across the whole phase. The results above show a mean score of 3.75 out of 5.

The following question in the survey asked teachers to explain their rating. Amongst the confident teachers, there was a wealth of experience and qualifications underpinning their practice, for example:

- *MA in Educational Technology*
- *24 years teaching experience, 17 of those as Head of Computing*
- *Computing secondary specialist*
- *Degree qualified in Computer Science with experience of mainstream and special schools*
- *I've been teaching computer science for a number of years, have a degree in the subject and have worked in industry both as a programmer and in network management*
- *I have a varied experience in industry as well as a Masters in Computer Science*
- *CAS Master Teacher*
- *I have been involved in ICT/Computing for over 25 years and lectured in it for over 15 years*

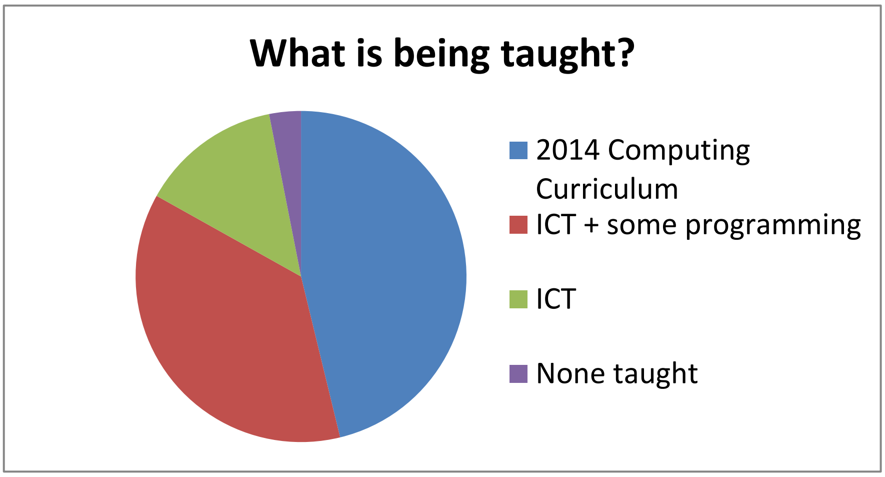
Some of the less confident teachers mentioned coding or programming as the area they found more difficult (7 mentions); others found difficulties with adapting or finding resources for SEND pupils (3 mentions). Where Computing was considered irrelevant for students, or wasn't being taught, confidence was also rated lower (5 mentions).

4.3 What is being taught?

Teachers were asked what was being taught in their settings, in terms of Computing and Information and Communication Technology (ICT). The majority (44%) are teaching the

2014 computing curriculum, but a large number (37%) describe what they teach as a mix of ICT with some programming. This perhaps reflects the way that special schools pick and choose from the national curriculum according to what is most relevant and accessible to their pupils. No ICT or Computing is taught in 3 of the schools and the remaining 9 schools (15%) are still teaching ICT.

Table 2: What is being taught?



Teachers were also asked how Computing or ICT is being taught, as this can vary from school to school depending on the cognitive level of the students and the organisation of the curriculum. 47% are teaching ICT/Computing as a discrete lesson, 12% as part of other curriculum areas, and 39% are teaching a mixture of these two depending on the ability of the students in the classes.

Finally, respondents were asked which, if any, key stage 4 or 5 qualification is offered in their school. These can be seen in the table below. This may help other schools understand the range of qualifications that are available for students with SEND. Functional Skills qualifications are by far the most popular option, but again this list reflects the diverse nature of special educational provision, with 2 settings offering A-level computer science.

Qualifications offered at KS4/5	Number of mentions
Functional Skills ICT (Edexcel, OCR, AQA, NCFE)	13
OCR Entry Level Computing	5
WJEC Entry Pathways	4
GCSE Computer Science (OCR, AQA)	4
Cambridge Nationals (Creative iMedia, ICT)	3
ASDAN	3
ECDL	3
BTEC IT	3
Digital Employability	2
IT User Skills (WJEC, NOCN, OCR)	2
A Level Computer Science (AQA)	2
Edexcel Pearson technical award in digital applications (CiDA)	1
Ascentis Finance and Online Communication	1
INGOTs	1
OCR Life and Living	1
AIM Awards – Computing, Raspberry Pi	1

EPQ Cyber Security	1

Table 3: Qualifications

4.4 Barriers

The teachers were asked which, if any, of the following they had found was a barrier to teaching Computing in their school.

Issues with the technology	45%
Lack of computing hardware	34%
Lack of confidence in teaching the curriculum	34%
Lack of training for using the technology	29%
No time allocated for teaching Computing	14%
Lack of resources for SEND	72%

Table 4: Barriers

Other barriers cited included the low level of maths and English, lack of relevance of the curriculum, lack of skills amongst support staff, the GCSE 9-1, and poor leadership or SLT simply not getting it (1 or 2 mentions each).

As with many mainstream settings, the lack of staff skills and confidence, and issues with technology working effectively are common barriers[5]. The most frequently cited barrier, mentioned by 72% of teachers who answered this question (n = 47), was a lack of resources for SEND:

> *"There is no breakdown of how these skills can be useful for students with SEND, so teachers need to research, call on each other and think of their own examples that will benefit each individual which as you can imagine is quite time consuming."*

> *"Not a lot of resources that meet the needs of SEN pupils. All resources are targeted at mainstream pupils. Schemes of work are not suitable for SEN pupils. Schools with SEN have to write their own schemes of work."*

Where this wasn't considered so much of an issue, schools had external support from the local authority, ICT teams or had a keen member of staff in school who adapts resources:

> *"We are fortunate to have a very active, knowledgeable and supportive LA school improvement service computing team and this helps to inform schools about new developments, initiatives and resources. It is difficult for a small school to have the expertise and time to lead such a specialist area and raise staff awareness and*

[5] Sentence and Csizmadia (2016) found that 'Subject knowledge', 'Students not understanding' and 'Technical problems' were the most commonly cited challenges facing primary and secondary teachers.

> *confidence in teaching these important skills and to see the relevance to our pupils without being switched off by the specialist nature of 'programming'."*

Clearly this is an area which would benefit from greater input in the future. In the Resources section I will look at places where materials can be found to support computing for SEND as mentioned in the survey responses. This can act as an initial point of reference for teachers.

5. Relevance of the Computing Curriculum to SEND Learners

5.1 General relevance of computing and computer science

As part of this research, I wanted to investigate the perceived relevance of the curriculum subject to learners with special educational needs and disabilities, and in particular the computer science elements of programming and computational thinking which are new to a number of schools. There is a concern that what is taught to learners working at very low cognitive levels can be irrelevant, and time is better used to teach essential life skills and provide experiential learning through a sensory curriculum. Ian Bean warns eloquently in his blog against adapting abstract elements of the computing curriculum for children at the earliest levels of cognition for the sake of it: "You can imagine… 'Listen to the computer beeping… Look at the numbers on the screen… Smell that data… taste that algorithm!'" (Bean, What do algorithms taste like?, 2015).

Teachers were asked two questions, providing an answer on a Likert scale of 1 (*not at all*) to 5 (*very*), followed by a long answer question to support the response.

1. How relevant do you consider ICT/Computing to be for your pupils?

The teachers were generally in agreement that ICT or Computing as an overall subject is relevant to their pupils, with 67% rating it at a 4 or 5. As with many of the questions, their response depended partly on the different needs of their students, as a blanket response didn't suit all cases – this was indicated in the long answer responses.

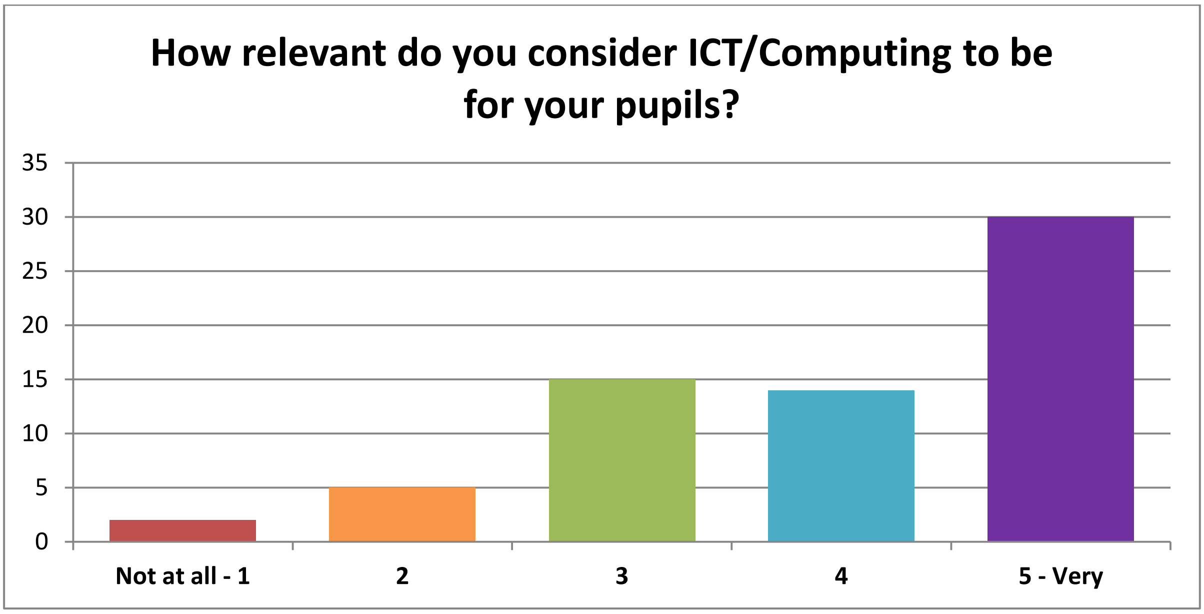


Table 5: Relevance of ICT/Computing

The main justifications for the relevance of computing and ICT in the long answer section related to the following themes:

- Technology is all around us, and enables pupils to become more independent and improve problem-solving and life skills (*21 mentions*)

"IT and Computing is a ubiquitous part of life and work. I feel that everyone should embrace technology."

- ICT/Computing can help improve literacy and is often a key tool for enabling communication (*10 mentions*)

"Computing skills provide communication opportunities for our pupils through the use of switches and AAC systems. Teaching these skills allows our pupils to have some level of independence."

- ICT/Computing is engaging and an area where some SEND pupils can excel (*6 mentions*)

"Many of our students are very engaged by ICT. It can be incredibly motivational for ASD and SEMH learners. It empowers students who struggle with social interaction to present and share their work widely."

- ICT skills can help pupils in their future careers (*6 mentions*)

"It's essential both as a means to access the curriculum but also to enable them to enter the jobs market."

- Technology enables access to the rest of the curriculum (*4 mentions*)

"For my students a lot of them can only access the curriculum through Technology."

- The subject can help students stay safe online and be more digitally literate (*3 mentions*)

"ICT use and safety is important, pupils may have enough knowledge to use e.g. social media, but not enough to stay safe."

11% of respondents considered computing and ICT to be less relevant (giving a rating of 1 or 2), but in their long answers, it was generally accepted that some form of education about and with technology was beneficial:

"It's a great tool, a vehicle for teaching more important things, e.g. communication, cause and effect, pre-numeracy and pre-literacy and pre-communication skills, as Assistive Technology."

"Such a mix, vital for social interaction for some. completely disengaging for others."

"To be able to understand technology around them. To have opportunities for problem solving. Fun."

2. How relevant do you consider the Computer Science elements of the curriculum (e.g. computational thinking and programming) to be for your pupils?

Fewer respondents agreed that the computer science elements of the curriculum were relevant to their pupils, with only 37% rating it with a 4 or 5. 31% considered that these parts were not relevant.

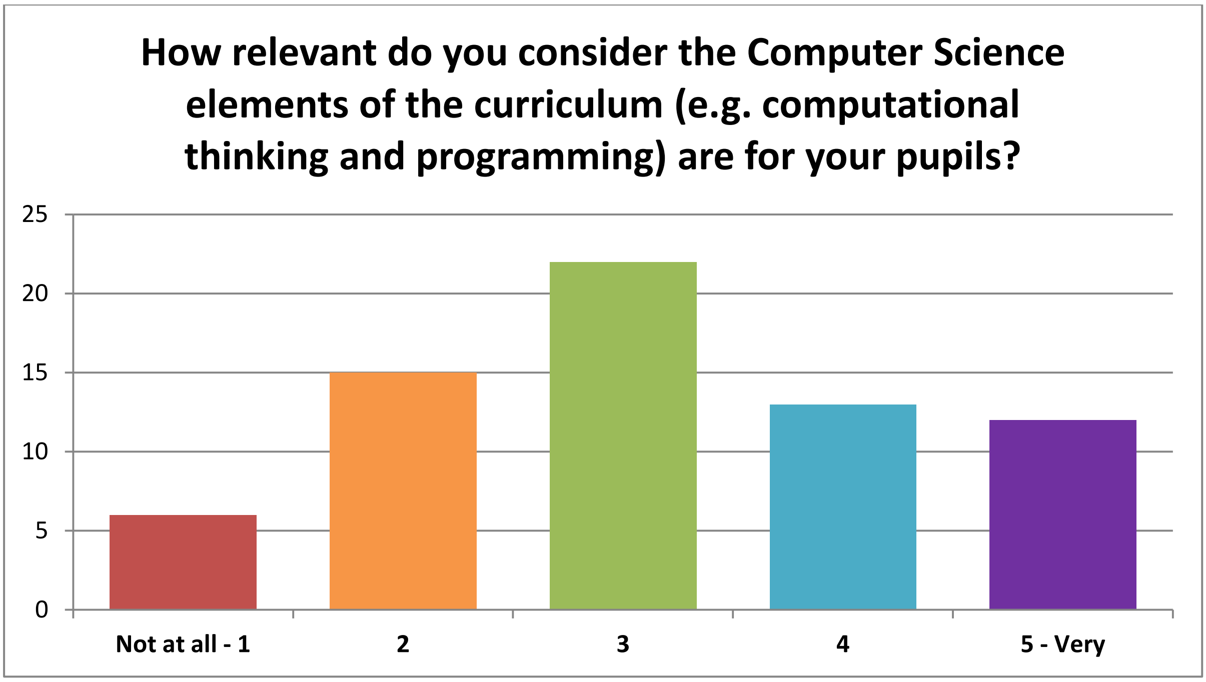


Table 6: Relevance of Computer Science

The main justifications for its relevance in the long answer section related to the following themes:

- Computational thinking and programming can help improve problem-solving, logical thinking and link to life skills (*15 mentions*)

"Because the development of computational thinking skills (problem solving) are functionally important in all daily activities and therefore impact on all areas of our children's lives."

- Meaningful links can be made to other curriculum areas, including maths and literacy (*8 mentions*)

"Pupils enjoy programming. The language taught is linked to other subjects e.g. maths. Direction, sequencing, debugging, writing instructions"

- Computational thinking and programming is engaging and an area where some SEND pupils can excel (*6 mentions*)

"Students are generally incredibly motivated by computing science. It allows them to apply literacy and numeracy skills in creative ways. They are proud of their achievements."

- This part of the curriculum provide opportunities for creativity (*2 mentions*)

"*A creative curriculum that engages and encourages self-directed, exploration, problem solving, discovery and expression is completely relevant to our pupils. The computing curriculum encourages all of these skills at many different levels."*

The nature of the students taught has an effect on the answers given to this particular question:

- Where there were no students with severe or profound and multiple learning difficulties in their school, 67% of respondents rated the relevance as a 4 or 5.

- Only 26% of respondents who teach students with severe or profound and multiple learning difficulties rated the relevance as a 4 or 5.

This is not surprising, since the main reasons given for programming and computational thinking not being relevant related to the ability of the pupils to access and understand them:

> *"As most of them have SLD [severe learning difficulties] and are working below the NC [national curriculum] they struggle to understand those concepts."*
>
> *"Many of our pupils are operating at a low level cognitively and are focusing on understanding and using cause and effect."*
>
> *"Not really relevant for over half the pupils because their literacy is so low that it becomes impossible to teach."*

Another effect on this opinion related to the confidence of the teachers in teaching the computing curriculum. Among the teachers who rated themselves as very confident, 77% considered ICT/Computing as relevant to their pupils (with a rating of 4 or 5) and 51% considered the computer science elements to be relevant. This may be because they are more confident in adapting the curriculum for their particular learners due to a better understanding of the concepts. Computational thinking in particular may be considered more relevant when teachers are shown examples relating to literacy, numeracy and life skills, such as sequencing instructions, or finding patterns in numbers and shapes. There is however a question, which I address later in section 7, about whether this kind of computational thinking is still computational in these contexts, or simply literacy and numeracy skills and general problem-solving.

5.2 Computational thinking skills to support learning in other areas

The majority of pupils in special schools will not go on to become computer programmers in the future, nor even take a qualification in computer science, and therefore some teachers may argue that a good grounding in computational thinking is not strictly necessary. The question I wanted to investigate was whether these skills can still be relevant and meaningful for pupils with moderate and severe learning difficulties, and whether they can help them make sense of the world around them or support other areas of learning. This survey aimed to find out if teachers around the country considered the teaching of computational thinking skills as useful to their students in terms of supporting learning in other curriculum areas.

The teachers were asked, "to what extent do you feel developing computational thinking skills has supported learning in other areas of the curriculum for pupils?"

Only 12% of respondents gave the answer *Not at all*, and 3 respondents stated that computational thinking is not taught in their school. 80% of the teachers considered there to be some effect on learning in other areas. See the full results below:[6]

[6] Some teachers ticked more than one box, hence the discrepancy in the total numbers – this was probably to reflect the difference in relevancy to specific groups of students within their school.

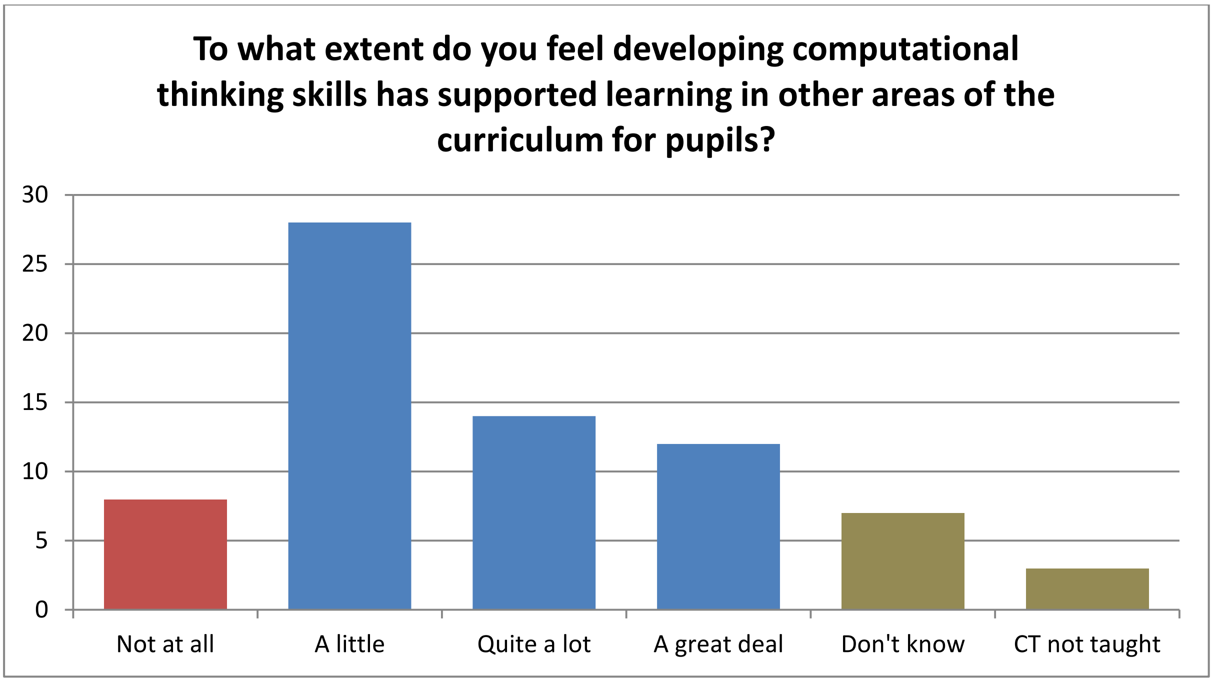


Table 7: Does CT support learning across curriculum?

Again it was clear from the long answers to the previous question that it depended on the ability level of the young people concerned, and for some of them, it simply wasn't meaningful.

There were, however, some clear examples where links were made when teachers were asked a follow up question: *Do you have any examples to support this answer?* Many also referenced wider computer science concepts and programming approaches in their answers:

> *"I have a student who has struggled with understanding coordinates in Maths. However, he was able to use coordinates when creating a game in Scratch. He was then able to complete the coordinates tasks in Maths. Variables would be another example. It makes "academic" concepts practical."*

> *"Decomposition skills useful when thinking about sequences/stages for completing a cooking task in Life Skills."*

> *"Computational thinking has been using in a variety of contexts such as DT, cooking, geography, maths and English. Through the use of algorithms it allows pupils to grasp concepts or processes from these subjects. It allows them to physically follow the process from start to finish."*

> *"Many learners are not worried about making mistakes and then fixing the problem themselves. They don't expect to be able to get the right answer to the next question because they have understood the examples on the board. They can explain strategies which could be transferred to other areas of the curriculum."*

> *"Blue/Bee Bots programmable robots and Code-a-Pillar resources have taught directional language as well as problem solving, cause and effect, following instructions, collaborative working, speaking and listening, attention, etc. This has also been combined to teach geography through the use of a combination of Google Maps and Blue Bot apps. iPad apps that teach principles of problem solving, planning, debugging, collaboration, thinking skills, etc. Steps within a dance routine have been*

> *used to teach about instructions within algorithms. Maths concepts taught through Scratch."*

It is clear from these responses that many teachers are finding ways of teaching computational thinking and programming with reference to other curriculum areas, in order to support learning in these specific instances. In section 7: Computational Thinking, I investigate further the current thinking around computational thinking and the transferability of skills.

6. Teaching Strategies and Resources

In this section I will investigate the strategies and resources that have been useful to teachers in teaching computing to their students. Again it is important to highlight that these strategies may not be effective for all students with SEND. Hopefully this will provide a basic guide to what works to other teachers looking for guidance, and act as a starting point for further research into effective strategies and technologies.

6.1 Strategies for Teaching Programming and Computational Thinking

Respondents were asked which strategies they found successful when teaching programming and computational thinking to their students. The following four areas were indicated as being particularly successful.

Unplugged activities:

These activities originated with the CS Unplugged project (Bell, Alexander, Freeman, & Grimley, 2009; Nishida et al., 2009), and describe tasks that take place away from a computer, to model key concepts in different ways, often with a physical element. This can be especially true with students with learning difficulties or autism.

Nearly a third of teachers (*19 mentions*) mentioned using practical, active tasks or unplugged activities, as a way of helping pupils understand a concept and develop logical thinking skills. The majority of these related to sequencing instructions or steps in a task, with some relating to a specific programming concept (e.g. directional language introduced unplugged, then practised with the Bee-Bots):

> *"We always start with unplugged activities to understand the concept."*
>
> *"Unplugged algorithms linked to specific topics in other subject areas, this has allowed them to develop their logical reasoning skills and computational thinking skills away from an ICT setting."*
>
> *"At my previous school, I had the pupils 'programme' me with instructions as an introduction to computing; we then moved onto the Beebot - both were very entertaining and involving for the pupils!"*
>
> *"We generally start with "Simon says". Students produce sets of instructions for others to follow. ASD students are generally very literal - so we will set up tasks and they will follow them. We transfer the instructions to cards - which they can move around. This leads on to floor robots."*
>
> *"The simple use of laminated images with velcro applied to a laminated velcro strip for sequencing in programming for example."*

The unplugged approach is generally considered useful amongst the respondents, but as one teacher pointed out, *"this is only relevant to the minority of our children that have the requisite language and conceptual skills."* Another talked about introducing unplugged activities such as sequencing brio train track and moving around in PE, but felt it *"hard to then translate it to programming computer equipment."* For the unplugged activities to be fully effective as a computing activity, they should relate to a computational task in order to illustrate the concept being taught. Students with specific needs often struggle to make connections between learning in different contexts, and will need explicit instruction to make the links. See further information on this point in section 7 on Computational Thinking.

Relevant, personalised tasks:

Other teachers (*9 mentions*) talked about the importance of making activities fun and relevant to students' interests and methods of accessing knowledge:

> *"ASD students can be motivated by linking it to their special interests. For example, if a student likes raisins the robot can be programmed to collect a raisin from a specific location. If they like buses we can produce a bus costume, and the bus can be programmed to complete a specific route."*
>
> *"A range of strategies both visual/non-visual etc. An eclectic approach to teaching matched to a pupil's needs and personality."*

Motivation and engagement plays a large part here, but there is also a benefit of teaching complex or abstract concepts through a familiar context in order to reduce the amount of new knowledge that is required for a learner to process. A range of teaching methods also ensures that students with different needs can access the learning, for example using visual support for autistic learners, or audio support for visually impaired learners.

Small chunks of activity:

Interest is also maintained through short bursts of activity, and complexity is reduced by decomposing problems into smaller parts (*2 mentions*):

> *"Activities are in short bursts to maintain interest."*
>
> *"Keeping it as simple as possible and ensure that it has all been chunked into bitesize pieces."*

Students with learning difficulties can struggle with multi-step problem solving, or sustained concentration on one task, and therefore it helps to break up activities into shorter stages that can be tackled individually. More able students can start to break up problems themselves by applying the computational thinking skill of decomposition.

Physical computing:

In terms of programming tasks there was a preference for using physical computing devices, such as the Bee-Bot, BBC micro:bit, Code-a-pillar and Lego robots (27 mentions):

> *"Tangible coding such as osmo code, code-a-pillar and pro bots."*
>
> *"The BBC Micro:bit is an excellent device to help develop programming skills particularly for my visually impaired students. The device enables students to receive immediate feedback both audio and visual. It also enables students to develop an understanding of communication between devices using the radio communication facility. It also enables students to gain an understanding of IoT and radio control."*
>
> *"Using Lego or a robot for teaching python. The students like to see an outcome from their programming."*

Seymour Papert referred to the original floor turtles as "objects-to-think-with" (Papert, 1980), and the effectiveness of learning through active exploration of the world, including physical objects was developed as part of his constructionist theory of learning. Floyd et al (2016) describe digital making as "a tangible representation of computational thinking". Physical computing devices provide a tangible interface and a physical output for the code, such as movement, sound or lights, which are desirable features when working with young people who may require multi-modal ways of accessing learning. The immediacy of output for the code is also motivating and can help to engage learners in the process of programming, although in some cases downloading the code to the device can be more convoluted.

Ryan Hayes discusses the benefits of physical computing in the follow-up interview (see appendix for full transcript):

> *When I first started teaching here I tried to do more complex stuff like Scratch, HTML, MS Logo and different things like that, and the pupils just weren't interested whereas things where they can actually build it, or program it with an instant effect seems to keep their interest and keep them motivated much easier.*

In a literature review of the pedagogy of computer science in schools, Waite concludes that there is limited empirical research as to the efficacy of approaches with physical computing, but a number of the studies mentioned indicate that the tangibility of a device such as the BBC micro:bit or Lego WeDo robot can support understanding and the development of literacy and numeracy concepts (Waite, 2018). This is an area that will benefit from further research and investigation of the benefits in the SEND-specific context.

It is not surprising that these strategies align with basic teaching approaches across the curriculum to support SEND learners. For example, the advice from Jisc for teachers of students with learning difficulties includes the following rules:

- Make learning participative
- Encourage peer learning
- Break tasks down into smaller steps that will incrementally build into the task objective
- Use learners' own words, language, materials and personal context - be clear about activity purpose and how it relates to the skills needs of the learner (Jisc, 2014)

These examples of good practice can easily be incorporated into the computing lesson, in particular through unplugged activities and physical computing projects.

The strategies mentioned in the survey are also not dissimilar to the findings of the 2016 survey of computing teachers (the majority of whom work in mainstream schools) conducted by Sentence and Csizmadia. They identified 5 main themes: unplugged-style activities, computational thinking, contextualisation of learning, scaffolding programming tasks and collaborative working (Sentence & Csizmadia, 2016). There was only one mention of collaborative learning in this survey of SEND settings, which could be a reflection on the nature of the young people being taught, but may provide great benefit if it can be developed as a strategy for learning.

Specific approaches to teaching programming:

There was little mention of more specific approaches to teaching programming in the survey responses, with teachers mainly providing examples of useful resources and general teaching approaches. More detail was often given in the follow up interviews, where teachers were specifically asked about effective strategies for teaching programming.

Ryan Hayes talks about providing code to students, for example using the Crumble Controller, which is in the wrong order, or with errors for them to correct: "S*o we give them code, they debug it and sequence it themselves. They're really progressing and enjoying it,* *especially pupils who you wouldn't expect to be bothered about code, going 'oh sir, you've got this wrong' which is great."*

Darren Craddock, working with SEND classes in a mainstream setting, uses "*Parson Programming (putting code blocks into the correct sequence rather than just asking children to come up with lines of code). Best strategy I think is to give students something fun/interesting/relevant already made which they can tinker with*."

There are a number of approaches that have been proven successful in mainstream that could be adopted for use with special needs learners. For example the PRIMM approach by Sentence and Waite (2017) which involves the following stages:

- Predict
- Run
- Investigate
- Modify
- Make

This would provide a number of points of access to a young person, from a highly scaffolded activity where they simply run code and observe what happens, to predicting the outcome of a piece of code, modifying existing code and eventually making their own programs without support. Learners with poor working memory or difficulties with written language can still investigate how a program works, discuss what different parts of the code do, and create their own version by adapting pre-existing code, without having to remember where to find blocks or how to spell specific commands.[7]

Wilson and Brown propose ten quick tips for teaching programming. The following four are particularly relevant to a SEND context, and complement the PRIMM approach:

- **Have students make predictions** – predicting what the code does provides an opportunity for students to engage with code without having to create their own program, reducing cognitive load. Furthermore, research shows that students how make a prediction retain more knowledge than those who simply see how the program works.
- **Use worked examples with labelled subgoals** – remixing a worked example can provide scaffolding for learners who may struggle to type in a large amount of text, or to create a block-based program from first principles. In addition, labelling the parts with consistent descriptions help students to make connections and spot patterns between different programs.
- **Use authentic tasks** – providing programs with a specific, real life context can help motivate and engage learners who have a limited set of interests, and reduce cognitive load if the subject matter is already familiar.
- **Don't just code** – an example is given of using Parson's Problems where the code is provided in the wrong order, sometimes with distractor elements, for students to sequence into a working program. There is no need for students to have to remember how to write the specific code, but it requires an understanding of how a program is put together. (Brown & Wilson, 2018)[8]

These approaches may provide an effective starting point for teachers in determining what works well with their individual pupils.

[7] You can read more detail on this approach at https://blogs.kcl.ac.uk/cser/2017/09/01/primm-a-structured-approach-to-teaching-programming/.

[8] For further detail, Mark Guzdial discusses the prediction, Parson Problems, projects in context and sub-goal labelling in a video in the blog post here: https://computinged.wordpress.com/2018/04/20/teaching-computational-thinking-across-an-entire-university-with-guest-blogger-roland-tormey/.

6.2 Resources

Below is a list of all the different resources used by the teachers to teach computing, and the number of mentions in the survey.

Physical Computing Devices

As mentioned in section 5, physical computing devices are very popular, with a tangible interface that helps students make connections between their actions and the results. These devices also provide a variety of sensory outputs, for example sound, images, light and movement, which can help with retention of information, engagement and provide immediate feedback. Physical computing devices also lend themselves to creative projects with cross-curricular links, for example building a Bee-Bot maze on an underwater theme and creating a sea creature costume for the robot, or using a BBC micro:bit or Code Bug to demonstrate conductivity in science using different materials.

Physical computing devices	Mentions
Floor robots, e.g. Bee-Bots, Blue-Bots	18
BBC micro:bit	5
Code-a-pillar	4
Remote control cars and toys (e.g. TTS)	4
Pro Bot / Probotix	3
Sphero	3
Lego Robots	2
Cubetto	1
Eye Gaze	1
Osmo	1
Crumble controller	1 (interview)
Microsoft Torino	1 (interview)

Table 8: Physical Computing devices

The Bee-Bot and related floor robots are by far the most popular devices, with a low level of understanding required. Unplugged resources, such as command cards, will be useful to plan out programs and provide opportunities for debugging and discussion. Newer physical computing devices can provide a more tangible representation of the program through shaped command blocks or sequencing components, e.g. Code-a-pillar or Cubetto. Microsoft's project Torino is an inclusive physical programming language designed specifically for children with vision impairments, using physical components linked together, and is currently still in beta.[9] See also the interview with Jonathan Fogg in the appendix for how it is being used with their students.

Programming Apps (on mobile devices)

Apps such as these can be useful for consolidating knowledge and encouraging independent learning. Learners may still benefit from planning programs away from the tablet prior to inputting the commands, in order to develop sequencing skills rather than relying on trial and error. This activity also creates a record of the program which is essential

[9] For more details see https://www.microsoft.com/en-us/research/project/project-torino/.

if learners are to identify and correct any errors (i.e. debugging). This can be done with laminated command cards, recordable buttons with image and audio prompts or simply with pen and paper. Some of the apps mentioned are not programming apps per se (marked in italics), but are great for cause and effect work and problem solving. This can help develop an initial awareness of a pupil's influence on technology, and may be more appropriate for children working at lower cognitive levels.

Apps	Mentions
Scratch Jnr	6
Bee-Bot	2
Blue-Bot	1
Tickle	1
A.L.E.X.	1
Daisy the Dinosaur	1
Lightbot Hour of Code	1
Kodable	1
Swift Playgrounds	1
Osmo	1
Cargo Bot	1
Amazing Alex	*1*
Inclusive Technology apps	*1*
Petsons Inventions	*1*
Fix Machine	*1*

Table 9: Computing apps

Scratch Jnr is the most popular app with symbol-based coding blocks, and provides good opportunities for cross-curricular learning and activities based around student interests.

Software and Websites

Many of these websites can be used to support independent learning working through a number of levels, where literacy levels are sufficient to understand the instructions, for example the Hour of Code website. Others are examples of open-ended programming environments, with a mix of block-based (e.g. Scratch and Kodu) and text-based languages (Python, Small Basic and Sonic Pi). These cater for a wide range of abilities and will need to be evaluated for suitability for specific students.

Some of the websites contain no actual programming environment (Inclusive Technology and HelpKidzLearn), but they can be used for students to learn about controlling technology and predicting the outcome of their actions at the start of their computing journey.

Software & Websites	Mentions
Scratch	12
Hour of Code – code.org	6
2Go - Purple Mash	5
Discovery Coding (Espresso)	5
Python	5
Kodu	4
Education City	2
J2Code	2
Minecraft	2

Busy Things	1
Codecademy	1
Python Tutor	1
Sherston	1
Mozilla Thimble	1
Snakify.org	1
Logo	1
Little Man Computer (LMC)	1
Small Basic	1
Sonic Pi	1
Visual Studio 2015	1
Flowol	1
Notepad++	1
HelpKidzLearn	1
Inclusive Technology software	1

Table 10: Computing software & websites

Scratch is by far the most commonly used programming software, and it has a great deal of flexibility in the types of activities that can be created, incorporating sound and images. It has huge scope for cross-curricular projects and activities for specific students. The programs can include a number of different input methods, for example the keyboard, mouse, microphone and video input, and it is also very effective when combined with a MaKey MaKey to create bespoke controllers. It does, however, remain inaccessible to students with poor literacy or visual impairment.

Teaching resources

As indicated in the question about barriers, there is a lack of SEND-specific sites and resources for teachers of computing. The Barefoot website has a section containing specific SEND resources around computational thinking, aimed at pupils working below national curriculum level. The Sheffield eLearning Service has published a SEND Computing Scheme of Work, plus there are a number of freely available resources for teaching Scratch to SEND learners on their website. The eLearning and Information Management (eLIM) service in Somerset has specific resources and guidance for teachers of students with special needs.

Teaching Resources	Mentions	URL
CAS Barefoot Programme	8	https://barefootcas.org.uk/
Sheffield eLearning Service	3	http://sheffieldclc.net/
Raspberry Pi Foundation	2	https://www.raspberrypi.org/
Crash Course videos – Computer Science	2	https://www.youtube.com/user/crashcourse
BBC Bitesize Computing	2	https://www.bbc.com/education
Computing at School	2	http://www.computingatschool.org.uk/
Nichola Wilkin Ltd	1	https://www.nicholawilkin.com/
Teach-ICT for	1	http://teach-ict.com/

Computer Science		
UK Bebras computational thinking challenge	1	http://www.bebras.uk/
eLiM planning	1	https://slp.somerset.org.uk/sites/edtech/SitePages/Home.aspx
TES	1	https://www.tes.com/
Code-It by Phil Bagge	1 (interview)	http://code-it.co.uk/

Table 11: Teaching resources

7. Computational Thinking

This is not intended as an in-depth review of Computational Thinking, as there is already a wealth of research addressing this subject. Rather it is an attempt to clarify what computational thinking is for teachers of students with special educational needs, how it is relevant to their learners.

7.1 What is Computational Thinking?

Computational thinking was first coined in 1980 by Seymour Papert, but it was Jeanette Wing's 2006 article that sparked off renewed discussions about its relevance in computer science education. Her definition of computational thinking (CT) "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." This is not thinking like a computer, but a way of thinking in order to solve problems in computational ways. She believes that "(c)omputational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability." (Wing, 2006). It was this definition and call to action that initially formed my thinking around computational thinking as a meaningful set of skills for learners with special educational needs. Could this thought process and set of concepts lead to a better understanding of computers and how to program them, at the same time as helping learners to make sense of the world around them?

One focus for the SEND Computing survey, therefore, was to find out whether computational thinking is being taught in SEND settings, how relevant teachers considered it is to the young people in terms of their learning, and whether it has benefits for learning in other subject areas. You can see the results in section 5.

7.2 What does Computational Thinking consist of?

There are a number of definitions of what CT consists of, in terms of approaches and concepts. There is a collection of these resources on the Computing at School website (http://community.computingatschool.org.uk/resources/252/single) for interested parties that want further reading. Many teachers in the UK use Computing at School definition (CAS, 2015) that identifies the following concepts as central:

- Logical reasoning
- Algorithmic thinking
- Decomposition
- Generalisation (Patterns)
- Abstraction

- Evaluation

This builds on the work by Wing and a number of other researchers, and it also forms the basis for the Barefoot 'Learn Computer Science' materials at https://barefootcas.org.uk/ which are aimed at non-specialists to help their understanding. This resource is widely used among teachers in the UK and was signposted from the survey for participants to refer to before answering the questions on computational thinking.

A number of approaches or practices are also indicated in the Barefoot resources, namely:

- Tinkering
- Debugging
- Creating
- Persevering
- Collaborating

Further explanation about these concepts and approaches can be found on the Barefoot website, including a 'Computing for SEND – Guidance Document' with examples of what this might look like in the special needs classroom.[10]

7.3 How transferable are Computational Thinking skills to other areas of the curriculum?

Wing writes about Computational Thinking as "a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use." (Wing, 2006) A number of other programs since then emphasise the generality of the skills. For example the introduction to Google for Education's computational thinking course for educators states "CT is essential to the development of computer applications, but it can also be used to support problem solving across all disciplines, including the humanities, math, and science." (Google for Education, n.d.). Grover and Pea consider it "reasonable to argue that it is in all the contexts outside of CS (computer science) classrooms that CT truly shines with its generativity" (in Computer Science Education, ed. Sentence et al 2018).

Grover and Pea, however, go on to explain that "transfer of learning across contexts does not happen automatically", and that explicit connections need to be made between the original and transfer learning contexts (ibid.) Tedre and Denning consider that it is easy to overemphasise the significance of CT, and that it may not be the most relevant problem-solving approach or method of thinking in other areas or contexts. They have examined the research and found that there has been no evidence to support the automatic transfer of CT skills to other areas (Tedre & Denning 2016).

We should therefore be cautious in ascribing too much influence to CT to improving learning automatically in other subject areas. Tedre and Denning also warn against "losing sight of computational models" (ibid.) – something which is prevalent in primary and special needs classrooms where there are fewer computer science specialists. Teaching pupils how to sequence a set of pictures representing the activities involved in getting ready for school does not necessarily constitute computational thinking. It should be linked explicitly to the importance of giving precise instructions to a computer or digital device in the correct order to elicit a specific outcome. Without this link and the use of the specific Computer Science language, it can be argued that pupils are simply being taught a literacy skill. However a large number of responses from the survey indicate that there are benefits of using

[10] This can be found at https://barefootcas.org.uk/activities/sen/ - free account required for access.

computational thinking as a framework for solving problems with special needs learners in areas outside of computing.

7.4 How relevant is Computational Thinking for SEND learners?

As we have seen above, it is important for students working at a sufficient level to write computer programs that explicit links are made to computing when undertaking computational thinking activities in other subject areas. Many learners with learning difficulties and other special needs struggle to make connections between what they are learning in one context and another, and “teachers need to spend time helping students connect new knowledge, skills and strategies to different contexts rather than expecting that transfer will happen spontaneously” (Westwood, 2013). Activities such as creating a class dance algorithm can be very effective for teaching about core programming concepts, for example sequence, repetition and selection. These concepts can then be referenced in context when creating a program to move a Bee-Bot, animate sprites in Scratch, or to light up LEDs on a BBC micro:bit, in order to reinforce understanding.

Nearly a third of teachers in the survey referred to the use of ‘Unplugged’ activities to help teach computational thinking. They often “engage learners in different modalities” (Caldwell, 2016), for example involving manipulating objects, physical movement, sound and music, images.

This multi-modality provides a strong argument for the use of unplugged activities to introduce such concepts with all learners, but it is particularly relevant with learners with special educational needs and disabilities, who may need to access learning in particular ways. The cross-curricular nature of these activities can also be hugely motivating for students, particularly autistic learners whose set of interests may be limited (DCSF, 2009). Curzon et al. explain that “if topics can be set in a context that relates to a student’s interests and pre-existing knowledge and understanding, then that interest can drive their learning.” (Curzon et al. in Computer Science Education, ed. Sentence et al, 2018) It is one fewer barrier to overcome if the context and prerequisite knowledge about that context is already familiar to the child. Computational thinking activities, as with programming projects, can harness a wealth of contexts at a cognitive level suitable for the student that are also age-appropriate.

Curzon et al also detail how unplugged activities can help students make sense of abstract concepts through physical objects that can be touched, manipulated and described. “This can make it much easier to explore the concepts involved and makes it easier to ask questions about things that aren’t understood. […] By providing a physical representation, the learner can point to and ask the question at the level of the analogy rather than having to fully verbalize it at the technical level.” (ibid.) This is of great benefit to learners with specific communication or learning difficulties who struggle with abstract concepts and require a multimodal approach. (NASEN, 2015 / Grandin, 2002)

Reponses to the survey indicate that the use of the specific language to describe computer science concepts and introduction of specific strategies to approach problems can also be useful to learners with SEND to help them describe and frame their learning. Davis et al list a number of examples of good practice in terms of teaching strategies for pupils with speech, language and communication needs including “(a)n emphasis on teaching language and cognitive process, and the strategies needed for effective generalisation through varying degrees of structure designed to match the child’s needs”. (Davis et al, 2004). Learning about decomposition and algorithms resonates with advice to break down activities into smaller parts and provide clear, step-by-step instructions, ideally with visual support, to complete a task for autistic or dyslexic learners (NASEN, 2015). Through computational thinking, however, the students can take ownership of these approaches and have the vocabulary to be able to talk about their learning in a more structured way. Thinking and talking about learning more explicitly is a metacognitive approach that has proven to be beneficial to learners in improving their attainment (Education Endowment Fund, 2018). This is not to say there aren't alternative models of thinking that could also be used for this purpose, but CT may be one option that engages and resonates with specific learners who have difficulty with choice, problem-solving and decision making (DfE 2009).

In the extended interview with Ryan Hayes, a teacher in Wales where computational thinking is taught across the curriculum as part of the Digital Competency Framework (Learning Wales, 2016), he talks about links made with other subjects such as maths:

> *"We did a whole topic before the summer based on algorithms, so I had several different classes making algorithms. The pupils then took that into maths and started to create their own algorithms for doing angles, for example, in triangles, and area in squares. They created their own algorithms, sequenced them, passed them to another pupil, debugged them then took it back and amended it. And it really did help in maths, as they got the process from computer science, and took it to maths and thought the steps to do these equations are basically algorithms."*

A recent article by Craig Smith also draws the conclusion from observing his students that teaching coding, and by extension CT, can help autistic children make sense of the world around them, and provide a strategy for solving problems beyond the computer:

> *"Learning to code is all about learning how to solve problems, work with others in creative ways, and think in a new language. Teaching children with autism employs the exact same skills."* (Smith, 2018).

Teachers should remain mindful that explicit links should be made between, for example, creating algorithms in maths and writing effective computer programs. If taught effectively then the repetition of key computer science vocabulary and computational thinking approaches in other subjects may help with the retention of knowledge and understanding of these concepts, by providing a familiar, more concrete context to reference in a programming activity. The added benefit is the opportunity for using computational thinking approaches as a framework for approaching everyday problems, such as the sequencing of a life skill or decomposing a complex problem into manageable parts.

7.5 SEND learners working below national curriculum level

Many teachers are working with students working below national curriculum level, who will complete few, if any, programming activities, and may not have the language to describe or think about their learning. Computational thinking is clearly not relevant to these young people as a specific set of skills and language, as the links between a sequencing activity and a computer program will never be made. However, it is common in special schools for classes to include students with a wide range of abilities and specific needs or disabilities. As such an experienced SEND teacher is used to adapting the curriculum at a number of different levels for their students in order to provide a flavour for meaningful activities (see Bean, 2015).

In the interview with Kirsty McCann, Deputy Head at a school for children with Profound and Multiple Learning Difficulties (PMLD) she talks about how they adapt the curriculum to best serve their learners:

> *"So we still follow the National Curriculum, but we bring it down to a very sensory level, as we have a sensory-based curriculum. So what we do is we have taken all the core strands of the computing NC program of study and taken the key elements from it. For*

> *example if we teach algorithms, it is about a set of instructions, so we make it very practical and relevant to our young people but we also use assistive technology, e.g. switch-adapted technology, so that our pupils can still get the idea of input and output."*

The Computing national curriculum program of study aims to equip "pupils to use computational thinking and creativity to understand and change the world" (Department for Education, 2013). For young people with PMLD, the understanding that they can press a switch to elicit a favourite sound or turn on a light is extremely powerful in terms of being able to influence their own environment.

What computational thinking offers, and in particular the unplugged approach to teaching it, is a way of involving all learners in computing in a context that is relevant to the learning needs of these children. Being able to sequence the actions of a life skill, e.g. getting dressed, is important to a learner with severe learning difficulties. It can also serve as impetus for a discussion around the order of instructions in an algorithm with more able students in the same classroom. As long as the outcome of a task is relevant, then it is meaningful to include computational thinking type activities in the special needs classroom with a range of learners.

8. Conclusion

It is reassuring to see the breadth and depth of experience in computing among teachers in special needs and disabilities settings around the country, and how the different elements of the computing curriculum are being embraced as a context for teaching key knowledge and digital skills across all subjects. Some form of Computing or ICT education is considered relevant and meaningful for students with special educational needs and disabilities to help them communicate more effectively, access the wider curriculum, and to teach digital skills that are essential for their future lives, whether in the workplace or at home.

Coverage of the 2014 National Curriculum subject is not consistent across all schools, and of course this survey only provides a limited snapshot of the current situation. There are, however, a number of common themes in the survey responses, demonstrating some agreement on what works with a range of special needs students. Physical computing devices offer an engaging way of teaching programming, providing support for learning through a range of input and output methods, and the immediacy of feedback. Unplugged activities provide effective ways of introducing programming and computational thinking to a range of learners with meaningful links to other subject areas through familiar and engaging contexts. Computational thinking also appears to offer additional benefits to these young people as a framework for solving wider problems, and in providing the language to talk about their learning more explicitly. Hopefully this report can provide a starting point for further research in schools into effective strategies for teaching computing and the benefits of introducing computational thinking skills to SEND learners.

Finally, the responses from the survey indicate a passionate community of educators, with a collective resource of experience, imagination and creativity that is used to adapt the curriculum meaningfully for their particular learners. They can learn a great deal from their colleagues in mainstream settings about effective approaches to teaching the subject, but it may be that there is much to be gained from a two-way conversation and sharing of experience in order to benefit students with special needs, or simply a different way of engaging with learning, in primary and secondary computing classrooms. After all, as the mantra goes, good practice for SEND is good practice for all students.

9. Acknowledgements

The author wishes to thank the teachers who took time to complete the survey, in particular those who consented to a follow up interview. Thanks must also be given to Sue Sentence, formerly of King's College London and currently the Raspberry Pi Foundation, for her advice, encouragement and feedback, without which this report may never have seen the light of day.

References

Barefoot, CAS (2014). Computational thinking. Available online: http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/

Bean, I. (2015). What do algorithms taste like? Available online: https://www.ianbean.co.uk/what-do-algorithms-taste-like/.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. New Zealand Journal of Applied Computing and Information Technology, 13(1).

Brown NCC, Wilson G (2018). Ten quick tips for teaching programming. PLoS Comput Biol 14(4): e1006023. Available online: https://doi.org/10.1371/journal.pcbi.1006023.

Caldwell, H., Smith, S. (2016). Teaching Computing Unplugged in Primary Schools. Sage Publications Ltd.

Computing at School (2015). 'Computational thinking: A guide for teachers.' Available online: http://community.computingatschool.org.uk/files/6695/original.pdf.

Davis, P., Florian, L., Ainscow, M. (2004). Teaching Strategies and Approaches for Pupils with Special Educational Needs: A Scoping Study. Available online: http://dera.ioe.ac.uk/6059/.

Department for Education (previously Department for Children, Schools and Families - DCSF) (2009) 'Supporting Pupils on the Autism Spectrum'. Available online: https://www.gov.uk/government/publications/inclusion-development-programme-primary-and-secondary-supporting-pupils-on-the-autism-spectrum

Department for Education. (2013). National curriculum for England: Computing programme of study. London, England: Department for Education.

Department for Education, (2017). 'Special educational needs in England: January 2017' available online at https://www.gov.uk/government/statistics/special-educational-needs-in-england-january-2017

Education Endowment Fund (2018) 'Metacognition and Self-regulation'. Available online: https://educationendowmentfoundation.org.uk/evidence-summaries/teaching-learning-toolkit/meta-cognition-and-self-regulation/

Google for Education (n.d.) 'Exploring Computational Thinking'. Available online: https://edu.google.com/resources/programs/exploring-computational-thinking/

Grandin,T. (2002). 'Teaching Tips for Students with Autism'. Available online: https://www.iidc.indiana.edu/pages/Teaching-Tips-for-Children-and-Adults-with-Autism

Israel, M., Pearson, J. (2015) 'Empowering K-12 Students with Disabilities to Learn Computational Thinking and Computer Programming'.

Jisc, (2014) 'Meeting the requirements of learners with special educational needs'. Available online: https://www.jisc.ac.uk/guides/meeting-the-requirements-of-learners-with-special-educational-needs

Learning Wales. (2016). 'Digital Competence Framework'. Available online: http://learning.gov.wales/resources/browse-all/digital-competence-framework/

Namukasa, I. K., Kotsopoulos, D., Floyd, L., Weber, J., Kafai, Y., Khan, S., Yiu, C., Morrison, L., Somanath, S. (2016). 'From computational thinking to computational participation: Towards Achieving Excellence through Coding in elementary schools'. Available online: http://researchideas.ca/coding/docs/CT-participation.pdf

NASEN (2015) ‘Supporting Pupils with Specific Learning Difficulties (Dyslexia) in Secondary Schools’. Available online: http://www.nasen.org.uk/utilities/download.286D7D63-9AA4-4E39-85E3A04B72D0C618.html

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

Royal Society (2017) After the reboot: computing education in UK schools. Available online: https://royalsociety.org/topics-policy/projects/computing-education/

Sentance, S. & Csizmadia, A. (2015). Teachers’ perspectives on successful strategies for teaching Computing in school. In IFIP TC3 Working Conference 2015: A New Culture of Learning: Computing and Next Generations Vilnius, Lithuania.

Sentance, S., Barendsen, E., & Schulte, C. (2018). Computer Science Education: Perspectives on Teaching and Learning in School. Bloomsbury Academic.

Sentance, S. and Waite, J. (2017). PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In Proceedings of WIPSCE 2017, Nijmegen. ACM.

Smith, C. (2018). ‘How Coding Can Teach Life Skills to Kids With Autism’. Available online: https://themighty.com/2018/04/coding-life-skills-autism/

Tedre, M. & Denning, P. J. (2016). ‘The Long Quest for Computational Thinking.’ *Koli Calling* 2016, 120-129. doi: 10.1145/2999541.2999542.

Proceedings of the 16th Koli Calling Conference on Computing Education Research, November 24-27, 2016, Koli, Finland: pp. 120-129.

Waite, J. (2017) ‘Pedagogy in teaching Computer Science in schools: A Literature Review’.

Westwood, P. (2013) Learning and Learning Difficulties: Approaches to teaching and assessment. Routledge.

Wille, S., Century, J. & Pike, M. (2017). ‘Exploratory Research to Expand Opportunities in Computer Science for Students with Learning Differences’ in Computing in Science & Engineering, vol. 19, no. 3, pp. 40-50, May-June 2017.

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

Appendix

Transcripts of follow up interviews with teachers:

Interview with Philip Crompton, Manor Academy - 4/5/18

1. Can you tell me a little bit about the school and your students?

We have approximately 150 pupils, with an intake of ADHD (Attention Deficit & Hyperactivity Disorder), EBD (emotional, behavioural difficulties) and ASC (autistic spectrum condition) children. We have an autism centre for 20-25 pupils, and we are KS3 to KS5 provision. Most pupils have moderate learning difficulties (MLD), and we have approximately 5 pupils with Downs Syndrome who have one-to-one support.

2. Can you tell me how computing is being taught in your school?

Computing is taught as a discrete lesson, taught by myself and my colleague. Everyone does computing, in every year group. There is a KS4 computing option where they do the OCR Entry Level Computing qualification. KS3 do the Edexcel Key Skills ICT award in year 9, and non-option students do the Functional Skills ICT in KS4.

3. When teaching coding, what resources and strategies work well with your pupils?

I use an eclectic approach, best fit. If it works for the pupils, then do it. For KS3 programming we use the Hour of Code resources – very visual, very gamey and they engage with it. I do the start with physical activities – I'll get them to instruct a pupil to do a task and we will sequence it. We start with the mundane, make a cup of tea, but everything must be exact, to understand it's not random, it's sequenced. The curriculum I teach is what would be covered in mainstream, just not as high a level, but I still cover everything. We use Flowol, look at the notion of algorithms. They can see the mimics, the traffic lights etc.

Last year I purchased a little robot, about 2 foot high, with handheld controls for the ASC pupils who are low ability. You press the buttons in order and then run the program. And they were really engaged with it, moving between cups on the floor. They were interested in that and they were able to do it. The next thing I bought was The Foos online program, with cartoon characters and bits of coding, and again very visual. They got the hang of it, but without individual support all the time it was a problem. They knew use a Repeat to perform an action a number of times, but they tended to use forward 5 times instead. They could complete the task just not efficiently.

I have used mainly Scratch with KS4, and Kodu, also with the y7 and 8s. They like to create games in Kodu – they can create something with an immediate reward whereas with Scratch they can accomplish much more but as it gets more complex their engagement with it changes. They find the reward tends not to equate with the effort, there's no intrinsic motivation – that's much better with Kodu than Scratch.

We use micro:bits – at the beginning it was new and novel but it was a passing thing. You have to plug it in, create the program, download it and they can't be bothered. Interest waned after 4 or 5 weeks. I can link the micro:bit to Kodu and Minecraft, but need to work on that in the holidays. But my initial thought was using the micro:bit as a controller, that would make it more appealing. We also use Lego Mindstorms, which are more interesting but a little more complicated. Without the internal motivation they just aren't engaged in the long term. The option group engaged with it more.

We also have Spheros, which we use with the ASC pupils. I bought two BB8 spheros and they liked those, but they were small. The 2 foot robot was much more physical and that interested them more. Any text-based coding would be really difficult as they can't read and write. I thought about Python and used Code Combat, but only the gifted and talented pupils engaged with it.

It's the simplest, the quickest, understandable from the get-go that works. There's got to be a return, they're not interested in learning for an exam, it's got to have some personal motivation for them. Based on what their interests are, their hobbies, to get that initial hook and get them engaged.

4. Where do you go to find resources?

No one place for resources, as it's such an eclectic approach. What works – modify and adapt. If there is something I can adapt, even if not a computing activity, I'll use it. I use Lego a lot, in terms of building up an algorithm, for sequencing. Anything they can use to engage with a concept then apply it to computing is good. The notion of making a cup of tea for example – use a familiar task to teach the concept first, but it must be clear and precise.

5. How transferable are the skills they learn in computing?

In desk-top publishing we create a cover sheet for a school magazine. My biggest problem is context so I give them blocks of text, and they add the headlines and titles. I dropped in an English classroom and they were looking at newspapers and headlines, and when I said to them in computing, you've done this before, they said, 'no we haven't!' They needed prompting that they had done it in English. They don't have the opportunity to get it either, as the teachers don't know about sequencing and algorithms in other subjects.

Interview with Ryan Hayes, St. Christopher's School - 1/12/17

1. Can you tell me a little bit about the school and your students?

Our school is the biggest one in Wales, the biggest one for additional learning needs in Wales. We've got 320 pupils at present, and we take pupils all the way from 7 to 19. The school's split into four areas really: you've the main body of the school which is moderate learning difficulties, behavioural difficulties, and autism; then we've got an autistic provision, where pupils are kept in one class, and it's very structure based; and we've also got a behavioural provision; then we've got an independent living area which is PMLD (profound and multiple learning difficulties) and severe learning difficulties. So I teach in the main body of provision, and some of the independent living pupils.

2. Can you tell me how computing is being taught in your school, and particularly with your learners, how you've made it relevant?

So at present the way we're trying to do it is very much cause and effect. So we program something and something happens. We've gone down the route of Spheros, coding in Osmo Play, Crumbles – so it's dead simple block programming, very much tangible programming, so something you can either get hold of or program on a screen that will instantly have some sort of impact. When I first started teaching here I tried to do more complex stuff like Scratch, HTML, MS Logo and different things like that, and the pupils just weren't interested whereas things where they can actually build it or program it with an instant effect seems to keep their interest and keep them motivated much easier.

3. And is Computing taught as a discrete subject or cross-curricular?

A bit of both at the moment. Basically in Wales we've had the Donaldson Report and we're in limbo at the moment, about to have a new curriculum in 2018 or 2020. In school at the moment we predominantly teach it through IT, so I'll teach computer science. The way we've structured our schemes of work and our year group, we do some kind of functional IT, so like your word processing stuff, in the Christmas term, then we do the computer science from January up to Easter, so we do a little on Scratch Jnr, we do a little bit on Osmos. It's basically progression so we start off on the Code-A-Pillars, with really simple loops, simple cause and effect, simple debugging and things like that. Then we will move onto some unplugged programming, program the teacher, a bit of Phil Bagge's Code-it – the Jam Sandwich and things like that. We then work all our way up, and it stops based on ability. So some years I will have groups that get to micro:bit, other years I have groups that stay on really simple coding using block programming – so do something, it moves, things like the Pro-Bots.

And then it's quite cross-curricular ours, as we've got the DCF in Wales, the Digital Competency Framework, which is split into Citizenship, Producing, Interacting, Collaborating and Computational Thinking. It's a bit like in 2014 we had the Literacy and Numeracy Framework introduced which is a cross-curricular document, a framework that every teacher has to try and get in their lessons, and it's very much a similar principle to that. As a school we ask the teachers in their planning to pick particular strands and areas, so predominantly, the way that we've gone this term is Computational Thinking with the majority of the teachers doing it. The main way that they do it is with Algorithms. So in Maths, English, Food Tech, all those different areas, we've got them teaching algorithm, so simple instructions to do things. In PE it's the same principle, simple instructions for how to kick a football etc. So as I said it's kind of interwoven because of the DCF which helps my job because I have to teach algorithms and the real basics such as loops, and debugging, and because they are learning it in maths and English and food technology, it makes my life so much easier when we get to it. I've seen this year, compared to previous years, pupils are really beginning to understand the basic concepts of debugging and sequencing which saves me lessons where I previously had to teach that.

4. So I suppose it's almost the other way round here, that learning in other subjects is supporting computing. And do you have any specific examples of how learning in other areas has been supported by what has been taught in computing? Do you think it does support other areas?

Yes, especially in maths we did a whole topic before the summer based on algorithms, so I had several different classes making algorithms. The pupils then took that into maths and started to create their own algorithms for doing angles, for example, in triangles, and area in squares. They created their own algorithms, sequenced them, passed them to another pupil, debugged them then took it back and amended it. And it really did help in maths, as they got the process from computer science, and took it to maths and thought the steps to do these equations are basically algorithms. It worked really well and we ended up laser cutting a load of algorithms for the maths teacher that she can use – these are based on a block programming type thing, with a basic start and finish, like in Scratch Jnr, and pupils have started using them in lessons to help them do the work.

5. And do they make the link when they are creating block-based programs from the work you've done with algorithms?

I don't think so, I think they compartmentalise it. I think it's the same as every subject. They will do something and it will go in that little box, then they will something else and that will go into another little box, but hopefully with the DCF and the computational thinking element of it, as they start to do more and more in other subjects and starting to do more complex elements of it. So for example there is Boolean searches, all sorts of loops, and complex computational ideas, so hopefully as they start to do more and more we should start to see a shift from them putting it into that little box and see the point to it. So we are starting to see it a little bit, in English we've started to do algorithms, and next week we will do programming in computer science then go into English and write it up.

6. You have mentioned a number of resources already, is there anywhere else in particular you go to find resources?

There is the CS Unplugged website – that's brilliant, we use that quite a bit. Obviously we dilute it. We've used your website a few times (www.sheffieldclc.net), to point us in the right direction, particularly with the unplugged stuff when we've done it previously. Also CAS and Barefoot, but when I've looked at them, I don't think they're as much use, they're not as comprehensive as I'd have liked. So we tend to just make our own. So we use Code-It (http://code-it.co.uk/) or CS Unplugged (https://csunplugged.org/en/) and water it down, take out key bits. I tend to focus on the Year 1, year 2 in primary school resources and then I'll start to change them slightly to make them more relevant to our older pupils. You can't go far wrong with the Code It stuff as a starting point.

7. Would you recommend the Sphero, Code-a-pillar and Osmo to other schools working with SEND learners?

Definitely the Code-a-pillar, it's so simple, but basically we've used it all the way through. I've used it with my older pupils, because they really like the idea of a challenge, to get it in and out of tables, from one part of the room to others. But also with 7 year olds with severe learning difficulties have been using it as it is so simple, hands on, tangible coding, just press the button and go. And you have the expansion packs to start doing loops. So you can do quite complex bits of code without really realising. The Osmos are great, because the pupils are playing a computer game and don't even realise they're coding which is great for me. I can go round and help them and they say, 'no, I don't need any help' – and these are pupils who in mainstream would be really disengaged and wouldn't like computer science, but as soon as I get the Osmo Code out I can't help them, they're doing it independently.

We've used the Spheros previously with Eye Gaze, which is really handy, so one of the pupils plays football with the Eye Gaze on it. Our PMLD teacher has done a lot with the Eye Gaze and Sphero, so using the Sphero apps on a Windows device. Have you seen the Big Life Fix program? There was a pupil on that with a switch playing different games, all open – source. We're looking at the minute to use Bluetooth switches to control devices to move forwards, backwards, right and left, and then hopefully start to use things like that with the PMLD pupils.

8. When teaching coding, what strategies work well with your pupils?

We started off with them mimicking the code, so I gave them the code to copy out and they were debugging it themselves. Then slowly, as time goes on and the skills of the pupils improve, we've started giving them wrong code. So on the Crumbles at the moment, we're giving them the wrong code, they go through it, going 'that's not worked, why has that not worked? That's the wrong thing there.' So that's the strategy of where we are at the minute. So rather than creating complex code from scratch, we start with the forwards, backwards, left and right with the ProBots, for example, and other different devices, then we give them loops and different code. So for example with the Crumble, then putting the code above the loop, so they work out that it's in the wrong place. So we give them code, they debug it and sequence it themselves. They're really progressing and enjoying it, especially pupils who you wouldn't expect to be bothered about code, going 'oh sir, you've got this wrong' which is great.

9. In an ideal world, what resources or training would you like to have (useful resources: hardware, software, materials)?

From our point of view, obviously everybody would like more money, more tech. We've been doing a bit of work with the Welsh government on the i5 framework and implementing new tech into different schools in America through MIT, and the big thing that they pointed at was they all wanted more technology and hardware – whatever school you go to they always want more. We'd love to buy every single pupil a Windows tablet, and have Crumble on it and all the apps that I'd like to use on it, Pivot etc. But that's not real life - so I'd like to see more teaching materials, especially for SEN provision. To be able to say to the maths teachers, the English teachers etc., look you've been doing algorithms, here's a really good way to put a bit of code in that one, or here's a website with resources and templates you can use. And with that you'd have to have your teacher CPD. There's such a big black hole in computer science special needs resources. If we're going to push computer science with special needs, additional learning needs, then having a baseline of materials for people to dip their toe in is so important.

Interview with Kirsty McCann, Bleasdale School - 14/12/17

1. Can you tell me a little bit about the school and your students?

We are a school for PMLD, that is profound and multiple learning difficulties, and associated challenging behaviour. We are an all-age school, 2-19 and we have 30 pupils on roll. So we have a range of physical disabilities, cognitive disabilities, and a range of complex physical needs. We are a residential school, with 6 pupils who board with us termly.

2. Can you tell me how computing is being taught in your school, and particularly with your learners, how you've made it relevant?

So we still follow the National Curriculum, but we bring it down to a very sensory level, as we have a sensory-based curriculum. So what we do is we have taken all the core strands of the computing NC program of study and taken the key elements from it. For example if we teach algorithms, it is about a set of instructions, so we make it very practical and relevant to our young people but we also use assistive technology, so switch-adapted technology, so that our pupils can still get the idea of input and output. To code we use Bee-Bots etc. So we bring it really down to make it meaningful to them, and practical, so you've got to be very innovative in how we deliver the message, the core essence of what computing is about.

3. Do you think computing is relevant to your pupils?

I think yes, basically I do, because for our young people, technology forms a very big part of their communication and their ability to engage with others and access the world. With having such complex physical disabilities, obviously none of our children are vocal, we're a non-verbal school, so they need to have a voice, and technology is what provides them with a voice and the choices for our young people. So it's really important that they have access to a progressive computing curriculum, it's not just about activities to hold them, and keep them busy, but it's actually moving their skill-set forward at their level so they are then able to engage with other people, so it's really important for them.

4. Can you give any examples of how learning in other areas has been supported by what has been taught in computing?

We have developed a switch progression document which is basically using the assistive technology and progressively making the skills more challenging for our young people so that they are moving through a series of skills to do with their switching ability. So it may just be to press the switch to respond to the reward, then to be able to choose between two switches to find an active switch, to use your head, to use your hands, to use your knees, to then move onto real assistive technology, where we have electronic devices that have a voice output, apps on the iPad such as Widgit Go, and the skills that we teach in those computing sessions are then embedded everywhere throughout our computing curriculum. So for our children, if they are cooking, they are using switch-adapted technology to control the food-processors, the cookers etc. So everything becomes switch-adapted for our children so they really need to have a set of skills to use switches.

5. What have been the most useful resources: hardware, software, materials?

Nearly everything that we buy is from Inclusive Technology, which is a really good website for us, as they have a lot of switch-adapted technology. Our best purchase recently is called a Magic Carpet, which is an interactive floor-projector, where it responds to very sensitive, small movements. So our children can lie on the floor and if they just move their elbow, it interacts with the floor projector, so for example it may be a pond, so if they move their elbow, the fish move, and it's encouraging them that it is cause and effect all the time. So that's been a really useful piece of kit. But the adapted switches are brilliant. You've got head switches, mounted switches, dome switches, you name it, we have it and it allows pupils to access whatever their ability is. We find resources to adapt that are specific to them. All the Inclusive Technology, Help Kidz Learn is a website we use quite a lot with switch-adapted games for the children, but we do struggle with age-appropriate content and teachers have to work really hard to come up with new, interesting ideas to get the pupils engaged because we are all age. So it's brilliant for the 2 year olds, as they have sing song nursery rhymes that they can switch, but as you get older, 16, 17, 19 – it gets really difficult to find things that are appropriate to them at their age, but that they can access at their level. So we work really hard together to make sure that the age-appropriate content doesn't detract from the skill, but the skills are embedded in what they do. Each child has a target each term relating to computing to make sure they are striving forward and can challenge themselves.

6. You mentioned Bee-Bots, are you using those with switches?

No, we've got some more able young people who are just working at what was P6-P7, so for our school, our demographic that's quite able, and they're able to press the buttons on a Bee-Bot and look at coding. We've also got coding apps on the iPad – that basic idea that if I tell you to do something, something will happen. I know that some teachers have used the BBC Bitesize Computing video clips as a starting point for their session and can then elaborate on those, which have been quite useful: the idea of what is a network, what is an algorithm etc., and that has been useful for the pupils and the staff, for them to understand the concept as well as being able to deliver it meaningfully to the children. The teachers take the essence of what's in the video clip and then really adapt it and make it sensory for our children. So you might have watched a clip, for example on networks, and there was a BBC Bitesize clip of two birds sending a message from nest to nest, so what they did was start to watch that, pause the video then they had string with hula hoops, which they could send the message through to each child in the wheelchair, so they could pass the message around the room, and the idea of the network being there, take away a piece of the network and you can’t send the message – so we have to adapt it to make it very meaningful, very sensory, very tactile for our children.

7. With the more able students, how do you support their programming?

Coding has to come down to the very basic idea of programming, so I tell you what to do and you do it back. So the children dress up as a robot, I tell you what to do and you do what I say. And then you apply that to the machine. So if you’re telling the robot to go straight ahead you’re telling it to go ahead once, and we use symbols to do that.

8. In an ideal world, what resources would you like to have?

Age appropriate content would be wonderful. We piloted some new touchscreens, with Android apps built into them, which are wonderful, but not one of them was for anyone older than early years. Anything above early years was encyclopaedia, that kind of thing, or very difficult to control apps, there was nothing sensory or for teenagers working at a very low level and I find that very difficult and it puts a very big strain on staff to try and think of all these new ideas. But our staff are very skilled and very creative in how they do this, so as a centre we’ve got a lot of expertise that we share with other schools, and as a school we’re quite well-resourced and well-versed in what we are doing. But there is still a lack of PMLD software out there.

Interview with Daren Craddock, Newfield School - 15/12/18

Daren teaches in a mainstream school, but has classes made up predominantly of students with special needs.

1. Can you tell me a little about your students?

The students who are in our low ability sets are students with a very low reading age, and we have a number of EAL students too, whose first language is not English at home so you have that communication issue as well. It's predominantly kids who have a high degree of dyslexia, very poor literacy skills, and that's a real barrier. So we are having to find ways of supporting them, and I want to give them the same opportunities as every other student.

2. Which strategies do you use to help your students access the programming and computational thinking elements of the curriculum?

The best thing bar none that we have been using recently are the BBC micro:bits, and having that physical, that kind of tangible thing to touch and you've got the excitement factor too, especially if it's new. And the thing with coding and programming, even with the more able students, it's so abstract and the great thing about the BBC micro:bit is the physical thing. Last week we had an enrichment day and my year 8s were having a taster and I had mixed ability sets for the day, and what they loved, we spent the day using the micro:bits. The connection between what they see on the screen - this is something real happening, something that they can touch. The great thing about the micro:bit you can very quickly use drag and drop Blockly on the screen, which they are familiar with through scratch, and although Scratch and Blockly can be wordy, they can recognise colours and shapes - but seeing something simple on the screen become physical in front of them on the display. And what they absolutely loved was using the radio facility and by the end of the session they were sending messages around the classroom, and I think that was the moment, 'I get it now, I can see how it works'. We didn't get much beyond the problem solving stage, click here, click that. And that's what I'm interested in, how do we get beyond 'pick that block, put that there'. One way we're working towards that is entering every student into the UK Bebras computational thinking competition, prior to that we did all the practice quizzes we spent the whole lesson the week before, and with the lower ability sets we entered them for a lower year group test, but there was an issue with the age on their certificates.

I would like to do more unplugged stuff as well with them, but it's the time. But what I've found works, is something physical, with quick results, with certification. The lowest ability sets also love games, so that's something else we do - we teach them typing skills using typing games, we use Typing Cat. It's also go to be pacey, with a lot of stop-starts, short bursts of activity.

<u>Interview with Jonathan Fogg, New College Worcester – 2/2/18</u>

1. Can you tell me a bit about your school and your students?

It's a specialist non-maintained school - so it's not a private school, we are a state school and we take students from all over the country and they're all either blind or visually impaired. So that's their primary disability and they're aged from 11 to 19, 19 is usually the oldest that we go to and they follow a mainstream curriculum as far as possible up to A level. We do have some that have got other needs and so some students might be on the autistic spectrum, some with moderate learning difficulties but they've all got to have as their primary need visual impairment. And we are a residential school.

2. How computing is being taught in the school and how do you make it relevant to your particular learners?

We teach it from year 7 upwards. So currently my year 7s are working with Microsoft on their project Torino, which is a physical programming environment that they're developing, so that's quite exciting. I will continue to use micro:bits and not just with year 7 but all the way through school as a physical programming element. And we go all the way up to A level computing where we use Visual Basic or Python programming environments. The common one is that we steer clear of are things like Scratch. That's a very visual environment that doesn't interact with the screen reading software that most of the students use.

3. Can you talk a bit more about the Microsoft Torino project?

I think as an entry into computing it's really nice it assumes a very physical way of engaging with the concepts of computing, it's quite limited in what you can do. So it's made up of play pods, loop pods, pause pods and decision pods, and you plug these pods into each other and depending on the combination you plug them in they will do different things. So it has to be connected to the laptop and the laptop controls what they are able to do, so you can have a MIDI player so it can play musical instruments. There's a speech element to it where you can tell a story- you can buy pods to play a story in a particular order or you can buy it to play music in a particular order. So using it to solve problems using a limited number of physical components that you plug together and the order you put them together generates the program for you. You can then look on the computer of the program that's being generated by the pods. It's very simple. You don't have to worry about that what's happening on the computer at all. The student doesn't, and there's a little hub that links up via Bluetooth to the software on the computer and you've got a single play pod you plug into the hub. It's got two dials on it. One changes the note the other dial changes the duration of the note. Through the hub you've then got a play button and a stop button. And if you've got the loop pod you plug that into the hub and it will control how many times it goes round the loop. So as a physical activity it's really nice for the students to be able to track what is happening. We're doing loops at the moment so they can track which module is being played at which particular time - they can put their hands on it and follow along as the program plays.

4. And in terms of the BBC micro:bit, it would be very interesting to hear why they are so suitable. What works very well with your students?

Starting with the things that didn't work, the programming environments are completely inaccessible. So that wasn't great and even the Python programming environment that was developed had some aspects of it that made it accessible but it wasn't accessible to speech. So we had to come up with our own programming environment for it and we use Python on it. And one of the nice things about it is that you can get speech out of it using Python language but also you can very easily get sounds out of it. So the students can get very immediate feedback and they tend not to use the visual aspects of the micro:bits. So I've made some of my own bits to plug into the headphones so there aren’t buzzers going off in the classroom. So yeah, they are nice little devices, very immediate feedback as I say. We had to use a program called Sixpad ++ and then we wrote a little bit of Python to allow it to interact with the micro:bit, and you do get some of the feedback from the micro:bit into the program.

5. Would you say that teaching computing opens up other areas of the curriculum for learners or any transferable knowledge?

I think the computational thinking is probably the bit that is most transferable rather than the actual programming skills, because I think teachers in other areas certainly in secondary are scared of introducing programming into their subject areas but the computational thinking I think is something that as it works through the students in the college that will certainly aid in all sorts of areas.

6. How do you teach the computational thinking? Do you teach it explicitly or just use it as a way of approaching problems?

We talk about it explicitly but just as a way of approaching problems really and one of the nice things about VI students is they do this all the time. It's the way they do it. All students do problem solving all the time, but particularly blind students are constantly having to think in abstract ways to solve problems. And so it doesn't come as a huge challenge to them or even anything that is that difficult to explain to them really.

7. Are there any other places you go to for teaching ideas or resources?

I go all over the place for ideas but I don't find a great deal. There's a great deal of modifications of other people's materials. So if I find a Python scheme work it tends not to fit our students requirements so it has been modified so that the programs that they're developing they will get something from. So yeah there's a lot of modification. I'll look on the CAS website for resources. Occasionally TES but not so much anymore. Various sites on the Web.

8. Are there any particular teaching strategies for teaching programming that work well for your students?

The only thing we do is that's any different really is to make it all very auditory rather than visual. I mean one of the things that we do find is that those students that do have mild learning difficulties is that all of the resources that are out there even when you get to exam level or entry level stuff, there's an assumption that it will be the literacy elements that they will struggle wit . And so they make everything very visual instead, with so much graphical elements to it which obviously doesn't fit particularly well with our students. So that's one of the biggest challenges - finding appropriate courses for the students if they're not on the mainstream path, e.g. GCSE.