Educ Inf Technol (2017) 22:469–495
DOI 10.1007/s10639-016-9482-0

Computing in the curriculum: Challenges and strategies from a teacher's perspective

Sue Sentance[1] · **Andrew Csizmadia**[2]

Published online: 5 April 2016

Abstract Computing is being introduced into the curriculum in many countries. Teachers' perspectives enable us to discover what challenges this presents, and also the strategies teachers claim to be using successfully in teaching the subject across primary and secondary education. The study described in this paper was carried out in the UK in 2014 where teachers were preparing for the mandatory inclusion of Computing into the curriculum. A survey was conducted of over 300 teachers who were currently teaching Computing to elicit their perspectives on challenges and strategies. From the analysis of the data, extrinsic and intrinsic challenges were identified for both teachers and students. In addition, a variety of pedagogical strategies were recommended by teachers from their own practice. In categorising approaches taken by teaching to support students five key themes emerged: unplugged type activities, contextualisation of tasks, collaborative learning, developing computational thinking, and scaffolding programming tasks. Further investigation could support whether these strategies can alleviate the challenges of teaching and learning of Computing for students and teachers. In particular developing student resilience in Computing is seen as a challenge while not many strategies are suggested. The results of this study will be useful for teachers who are new to the teaching of Computing.

Keywords Computer science education · Computing in school · In-service teacher education · Computing curriculum

✉ Sue Sentance
sue.sentance@kcl.ac.uk

Andrew Csizmadia
a.p.csizmadia@newman.ac.uk

[1] King's College London, London, UK

[2] Newman University, Birmingham, UK

1 Introduction

Computing[1] is being introduced as a new subject in the school curriculum in many countries, and as an important part of informal learning opportunities in others. This brings with it both excitement and challenges, as for any new subject. For teachers facing curriculum change, how to teach it is very pertinent. Introducing new content does not merely mean that teachers have to equip themselves with new subject knowledge, which of course in many cases they do (Brown et al. 2013; Sentance et al. 2013; Thompson and Bell 2013). Teachers also need to learn appropriate pedagogies for delivering a new subject, particularly in those aspects of computer science that relate to algorithms, programming and the development of computational thinking skills.

Recent literature relating to computer science education in school highlights a number of ways of making computer science concepts accessible, engaging and fun, and more importantly, giving students a deep understanding of these concepts.

In this study we asked a large number (over 300) of active Computing teachers how they recommend teaching the subject, in order to find out which strategies work well in practice. We balanced this with asking teachers about particular challenges they faced in teaching Computing. Our research questions are quite simply the following:

- What pedagogical strategies do teachers report work well for teaching computer science in school?
- What challenges do teachers report that they face?

Statements made by teachers who are currently teaching Computing in school have been coded, categorised and analysed, describing both successful strategies for teaching and the difficulties they face. The results reported in this paper are timely and relevant to teachers and teacher educators in the field of Computing, and we recommend that further investigation is carried out around the impact of the strategies suggested in primary and secondary education.

This paper also contributes to the area of pedagogical content knowledge (PCK) in Computing. PCK is the knowledge that a teacher has about how to teach their subject (Shulman 1986).

1.1 The ‘unplugged’ approach

Constructivist theory, based on the work of Dewey (1938); Piaget (1950) and Bruner (1996) suggests that learning is a cumulative and active process during which the student constructs knowledge and meaning for themselves as they learn, connecting with, and explaining new knowledge in terms of, what they already know. Constructivist learning theories applied to computer science emphasize the active, subjective and constructive character of knowledge, placing students at the centre of the learning process (Ben-Ari 1998). Specifically, constructivist learning, based on students’ active

[1] Computing is used as the subject name throughout this paper; it refers mainly to the computer science elements of the curriculum, and is also known as Informatics in other countries.

participation in problem-solving and critical thinking, has profoundly influenced the teaching of programming (Ben-Ari 1998).

Experiential learning that stems from constructivism describes the design of activities which engage students in a very direct way. Working with tangible real world objects is a central tenet of Papert's constructionism (Papert and Harel 1991) (which builds on constructivism). Thus, constructivist principles support the strategies of using more kinaesthetic and active approaches to teaching in the computer science classroom. In Computing this is embodied in the 'unplugged' approach.

The "unplugged" style of teaching refers to the use of activities to teach computer science concepts without the use of computers. It originated with the CS Unplugged project in New Zealand (Bell et al. 2009; Nishida et al. 2009) and has resulted in many related, kinaesthetic activities which stimulate an understanding of a concept in a very concrete and practical way. In a similar way, CS4FN (Computer Science for Fun) (Curzon et al. 2009) have generated many unplugged-style engaging activities and approaches by emphasising the importance of analogy as well as a kinaesthetic activity.

1.2 Embedding computational thinking

A key consideration in computer science pedagogy needs to be the development of computational thinking skills. Computational thinking was only recently popularised as a concept in 2006 by Wing (2006) – the original definition stems from Papert (1996) – but teachers of computer science have been facilitating these skills in their students for as long as this subject has been taught. Wing claims that computational thinking is for everyone and involves "*solving problems*, *designing systems and understanding human behaviour*, *by drawing on the concepts fundamental to computer science*". (Wing 2006, p. 34).

For teachers in England, guidelines have been developed recently suggesting how computational thinking can be explicitly taught as part of the new curriculum (Csizmadia et al. 2015). Computational thinking, according to these guidelines, is divided into a series of overlapping skills: abstraction, algorithmic thinking, generalisation and evaluation.

Many have recently suggested ways of implementing computational thinking into the curriculum (Barr and Stephenson 2011, Webb and Rosson 2013, Brennan and Resnick 2012, Lee et al. 2011, Selby 2012, Yadav et al. 2011, Van Dyne and Braun 2014, Sengupta et al. 2013). In our research we find teachers describe using a variety of activities that develop computational thinking skills in learners.

1.3 Learning to program

Programming is the aspect of computer science in school which is perceived to be the most challenging. A range of activities can be used that allow students to collaborate and construct problem solutions. As an example, the following suggestions, drawing on a constructivist view of learning, are made by Van Gorp and Grissom:

- Code walkthroughs
- Writing algorithms in groups
- Insert comments in pairs into existing code
- Develop code from algorithm in pairs
- Find the bugs in code (Van Gorp and Grissom 2001).

Reading and tracing code is also important in supporting the learning of programming (Lopez et al. 2008) and being able to do this is a pre-cursor to the problem-solving skills needed to write code (Lister et al. 2004). Tracing code refers to the process of stepping through a piece of code, often by hand, and noting the values of variables as the program proceeds. Lister later describes that novices need to be able to trace code with more than 50 % accuracy before they can begin to confidently write programs of their own (Lister 2011). In our study we were interested to see which pedagogical strategies were being used in the classroom by the teachers, and how teachers were supporting students in learning to read and write program code.

A final area of interest is the extent to which teachers provide a real world context for learning and relating it to students' interests and understanding and the value of a rich discourse regarding concepts (Grover and Pea 2013).

1.4 The teachers' perspective

The implementation of Computing involves a change to teachers' practice (Thompson et al. 2013) in both their subject knowledge and pedagogical knowledge. With respect to the introduction of technology in classrooms, a different context, it can be seen that there is an intersection between teachers' knowledge, beliefs and culture (Ertmer and Ottenbreit-Leftwich 2010) and this may be the same for Computing. Teachers within a context of change face many challenges. We draw on the work of Finger and Houguet (2009) who, also working in the area of adoption of technology into the curriculum, describe a range of intrinsic and extrinsic challenges that teachers face in moving from the intended to the implemented curriculum (van den Akker 2003). Although this work is in a different domain, this analysis is relevant in this case, or perhaps in any incidence of curriculum change. Figure 1 shows an extract (the relevant parts) of the conceptual framework used by Finger and Houguet in their analysis.

Black et al. carried out a study in the UK where they asked Computing teachers how they felt they could make the subject interesting (Black et al. 2013). The key aspects that they identified were the importance to teachers of making Computing fun and relevant. In carrying out our research we were interested to see whether the teachers' comments aligned with this study; in addition we asked more specifically for actual strategies that teachers use in their classroom that they feel to be effective.

In a similar way to Black et al., this paper focuses purely on the teacher's perspective in addressing these questions. Diethelm et al. emphasise the importance of the teacher's perspective to our understanding of computer science education as the teacher "*may work on many different abstraction levels or apply very different teaching methods for the same topic of the curriculum*" (Diethelm et al. 2012, p. 167). We wish to identify what these methods are, in particular identifying common themes that may help to provide guidance for teachers new to teaching the subject, as well as providing actual examples of teachers using effective strategies as we enter a phase of education when more and more students are studying Computing in school.

Section 2 gives details of the study. In Section 3 we then report on the results of the content analysis that was used to analyse the responses of the teachers. In Section 4 we draw out how this can contribute to the general area of pedagogical content knowledge in the subject, and then suggest strategies to overcome intrinsic challenges facing teachers.

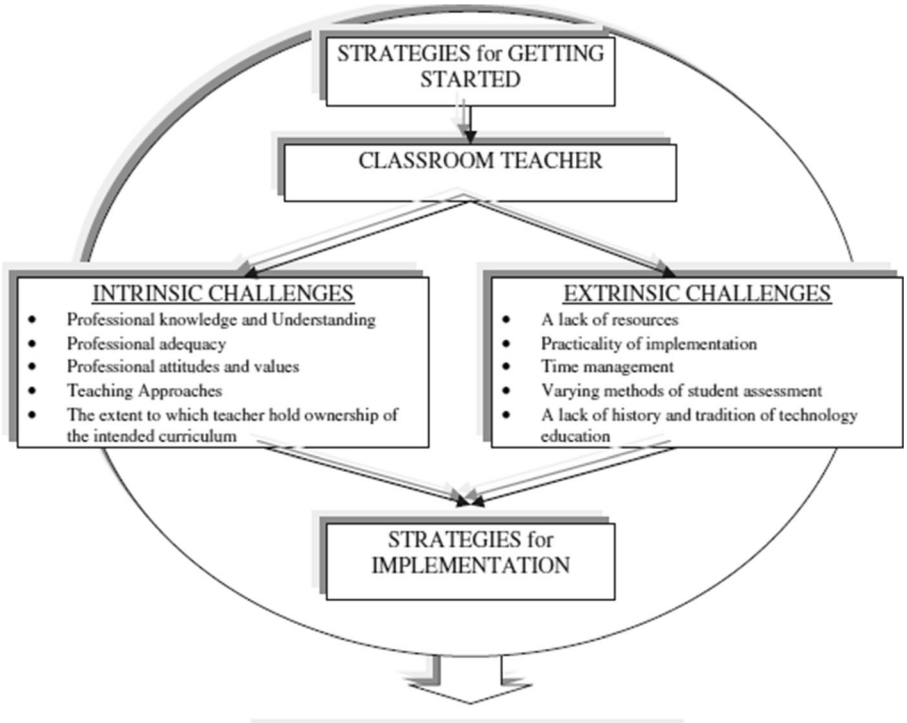

Fig. 1 Finger and Houguet (2009): Intrinsic and extrinsic challenges for teachers (Reproduced with permission from the authors)

2 The study

2.1 The context: Change in the curriculum

The UK has seen fast-paced change in the area of computer science education in the last few years (Brown et al. 2013; Brown, Sentance, Crick, & Humphreys, 2014). The state of computer science education is different in the four parts of the UK, with England having just implemented an ambitious new curriculum in Computing, to be taught from ages 5–16, and with a strong focus on computational thinking. This has been preceded by two years of preparation, as new qualifications were introduced and the draft curriculum proposed. Many schools and teachers in England had implemented elements of the Computing curriculum prior to the official starting date of the Computing Programme of Study in September 2014, as a void was left by the disapplication of the existing curriculum subject, Information and Communication Technology (ICT), in January 2012 (Brown et al. 2014).

In the UK there is a strong subject association for computer science teachers, Computing At School (Brown et al. 2013). Through this grass-roots community of practice teachers are able to share resources, share experiences and attend local events. The participants of this study were to a very large extent members of this community. In the data collected in this study, they describe the experiences, successful strategies,

and also the frustrations, of teachers who have begun to teach Computing in school over the last few years.

The Computing Programme of Study for the new English Curriculum (Department for Education 2013) is based on computational thinking principles, and thus teachers of computer science welcome guidance on how to deliver computational thinking skills, which is beginning to emerge.

2.2 The study: Methodology

A wide-ranging survey was carried out in February 2014, specifically targeting members of Computing At School (CAS), although invitations were extended widely. The survey was publicised via the CAS forum, as well as through social media channels. The survey included questions about teachers' location and work situation and also the amount of Computing they taught, the confidence they had in their skills to do so and whether they were involved in examination classes. The survey also included many other questions about participation in professional development activities and engagement with Computing At School in general, which are not reported on in this paper.

As one aspect of the survey, teachers were asked if they optionally wished to contribute free-text answers to the following four questions about their teaching.

1. What good techniques/strategies have you found for helping students to understand programming?
2. Please describe any good techniques/strategies you use for helping students to understand other aspects of Computing?
3. What difficulties, if any, have you experienced teaching programming?
4. What difficulties, if any, have you experienced teaching other aspects of Computing?

In the context of this survey, teachers in the England understand "other aspects of Computing" to be non-programming topics in the curriculum, which include learning about hardware, networking, data representation and logic (Department for Education 2013).

1417 members completed the wider survey (1126 of whom were practising teachers), with 339 teachers contributing at least one free text answer to the free text questions. In this paper we primarily focus on the 339 responses given by this self-selecting group of teachers but include reference to their other answers to survey questions where relevant.

The data were collected by an online questionnaire which was then input into qualitative data analysis software. The data consisted of the four free text questions described above, plus responses that these teachers gave to the other questions in the wider survey.

The data were initially coded in an inductive manner with respect to emerging themes, following the guidelines proposed in (Mayring 2000). For the challenges, the themes were then grouped to facilitate further analysis. A set of codes was then developed for the second coding. The coding scheme used for strategies is shown in Table 1 and the coding scheme for challenges shown in Table 2. The data were re-coded and verified by the two authors to ensure agreement on the interpretation of the teachers' statements. An interrater reliability analysis using the Kappa statistic was performed to determine consistency between the two researchers, separately for the coding of strategies and for the coding of the challenges (a sample of 15 % (N = 50) was used for challenges).

Although two questions were asked about strategies and two questions about challenges, the answers received in some cases did not relate to the specific question. Because of the content of the data it was not possible to report on the particular strategies for programming and those on other aspects of the curriculum. All answers under strategies were thus analysed together.

2.3 The study: Participants

We are discussing here three groups of respondents – 1417 survey respondents as a whole, of whom 1126 were teachers. We then have a self-selecting sample of 339 teachers who completed the free text questions. From this sample we identified that 97 % (N = 329) were members of CAS, as opposed to only 88 % (N = 1256) of the 1417 people that completed the survey, including those who chose not to answer questions about their practice. This may be because members of CAS may be more likely to support a piece of research that is presented to them by CAS itself, or may be that they are a more confident community by virtue of their engagement with CAS, but we cannot know this.

Of the teachers completing the survey we can see the distribution across the UK in Table 3.

Considering the type of school of our self-selecting sample we can see that the large proportion of teachers completing the survey were secondary teachers. In total 1126 teachers completed the wider survey of which 75 % (N = 841) were secondary school teachers, and of the 339 in our self-selecting sample, 77 % were secondary school teachers. Since the time of the survey there has been an increase in the number or primary teachers joining Computing At School but the figures shown in Table 4

Table 1 Coding scheme used (strategies)

Strategies – coding scheme used		
Algorithms (link to)	Games	Relate to real world activities
Block to text strategies (use of)	Give feedback immediately	Robotics
Break down/decomposition	Interactive lessons	Scaffolding
Celebrate progress	Keep simple	Start children young
Code manipulation	Learn theory through coding	Tangible interfaces
Collaboration	Learn through examples	Team coding/pair programming
Computational thinking	Learn through mistakes	Tutorials
Contextualisation	Lots of practice/Little and often	Unplugged strategies
Demonstration and modelling	Make it achievable (break down)	Use of videos
Develop troubleshooting skills	Make it fun	Use of examples
Differentiation	Mathematical skills	Use of hands-on experiences
Discussion and questioning	Minimise syntax	Use of pseudocode
Emphasise problem solving	Online learning	Use of simple IDEs
Emphasise similarities	Peer mentoring	Use of variety of activities
Flipped classrooms	Provision of support	Use of visual prompts
Flowcharts	Reference to particular software	Work at own pace

Table 2 Coding scheme (Challenges)

Challenges – coding scheme used		
Teachers and assessment	Teachers' lack of time	Students and documentation
Teachers and coding focus	Teachers recruiting students	Students and expectations
Teachers and differentiation	Teaching approaches	Students problem-solving
Teachers and digital literacy	Teachers are under-ambitious	Students starting too late
Teachers and dry topics	Students and resilience	Digital divide
Teachers and funding	Students and instructions	Gender issues
Teachers and lack of training	Students and literacy	Students not suitable
Teachers and large classes	Students and Maths	Choosing resources
Teachers and programming	Students and practice	Finding quality resources
Teachers and results	Students and remembering	Lack of resources
Teachers and subject knowledge	Students lack confidence	Physical Computing issues
Teachers and troubleshooting	Students not engaged	Technical problems
Teachers' lack of support	Students not understanding	

roughly equate to the balance of membership within CAS at that time. Other categories of school are not as common as primary and secondary schools so we would expect to see fewer teachers from those groups. Some teachers teach across the whole age range.

Teachers were asked to identify the key stages that they were currently teaching, where the key stages (KS) are labelled from 1 to 5 according to the ages of the pupils, KS1 being the youngest group at age 5–7 and KS5 being aged 16–18. Table 5 shows the numbers of teachers reporting that they teach the different age groups.

From Fig. 2 we can see that our self-selecting sample have a similar profile to those in the overall population of the survey respondents, with a slightly higher percentage of teachers teaching post-16 students.

We asked the teachers how much teaching of Computing they were currently doing (Fig. 3). Note that this survey was undertaken in the academic year prior to the introduction of the Computing curriculum from age 5 although the previous subject of ICT had already been disapplied so teachers were starting to introduce Computing in preparation; examination subject teachers at KS4 and KS5 were already teaching Computing, and in the case of KS5 teachers had been doing so for many years.

Table 3 Distribution of respondents across UK

UK	Whole survey population		Self-selecting sample	
England	1028	91.3 %	314	92.6 %
Northern Ireland	7	0.6 %	3	0.9 %
Scotland	47	4.2 %	12	3.5 %
Wales	16	1.4 %	3	0.9 %
Outside UK	28	2.5 %	7	2.1 %
Total	1126		339	

Table 4 Type of school and number of teachers

Type of School	Age group	Whole survey population		Self-selecting sample	
Primary	4–11	229	20 %	55	16 %
Middle	8–13	16	1 %	5	1 %
Secondary	11–16 or 11–18	841	75 %	260	77 %
Sixth Form College	16–18	24	2 %	12	4 %
Further Education	16–19	16	1 %	7	2 %
		1126		339	

Here it can be seen that our self-selecting sample differs from the overall respondent population of teachers as they are teaching more hours per week overall, with 69 % teaching at least 5 h a week of Computing. This sample includes primary teachers who teach many other subjects as well and would not be likely to teach more than one or two hours of Computing every week.

Teachers were asked to rate their confidence in being able to deliver the new Computing curriculum on a Likert scale from 0 to 10. This self-selecting group were largely confident in their Computing teaching, with 85 % rating their confidence at 6 or more out of 10. Their confidence overall was greater than the confidence levels of the wider population completing the larger survey, as can be seen in Fig. 4. The general confidence in the teaching of Computing will have contributed to their willingness to participate in a free text questionnaire on their practice and also will have a bearing on the content on their responses. This indicates that they may not be `typical' of the whole teacher population, but represent teachers who are more comfortable teaching Computing.

3 Findings

The findings of the study will be divided into Challenges and Strategies. In the study we asked questions about strategies before the "challenges" questions, but present challenges first here as it is then possible to see that teachers are suggesting strategies that can overcome the challenges faced by themselves or others.

Both researchers coded the data as described above and the interrater reliability for the researchers was found to be Kappa =0.601 for coding of strategies and 0.684 for coding of challenges. 0.41–0.60 is moderate agreement; 0.61–0.80 is substantial agreement (Viera and Garrett 2005) so these figures can be seen to be on the

Table 5 Number of teachers for different key stages (age-groups)

Key Stages	KS1	KS2	KS3	KS4	KS5
Ages	5–7	7–11	11–14	14–16	16–18
Teachers from whole survey	152	257	809	798	560
Teachers from self-selected sample	43	74	258	253	194

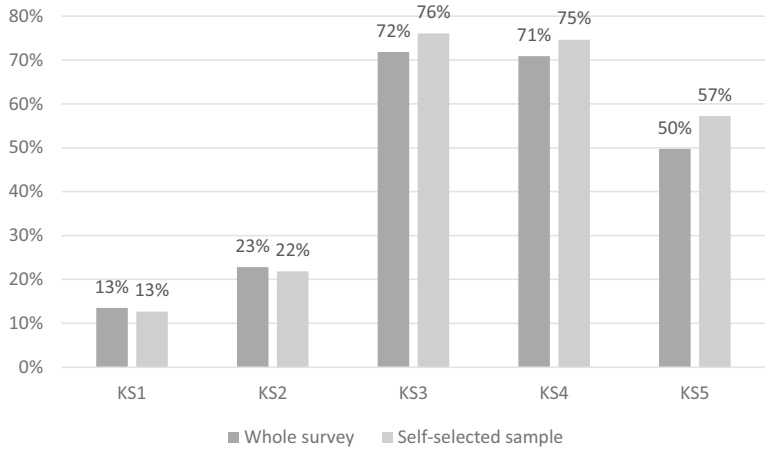

Fig. 2 Key stages (age-groups) taught by teachers

moderate/substantial agreement boundary. During the analysis of the challenges one of the codes "teaching and coding focus" was removed due to a lack of common understanding of the code.

3.1 Challenges

In this section we report on the challenges that teachers report on with respect to the teaching of Computing. A number of themes emerged from the analysis of the data.

Overall it can be seen that all of the challenges coded fell into the categories that we had broadly identified as teacher, student and resource-related. Of 855 codings, 40 % (N = 342) related to challenges directly experienced by the teacher, 38 % (N = 325) related to challenges which could be seen to difficulties experienced by the student, and 16 % (N = 138) related to resources. In 50 cases, just 6 %, teachers reported to have no problems with respect to teaching Computing.

Considering these three areas in more detail, Table 6 shows the most commonly coded categories with regards to the challenges and difficulties described by teachers. The challenges categorised as relating specifically to teachers were subject knowledge, differentiation, approaches to teaching topics, assessment, lack of support and having to

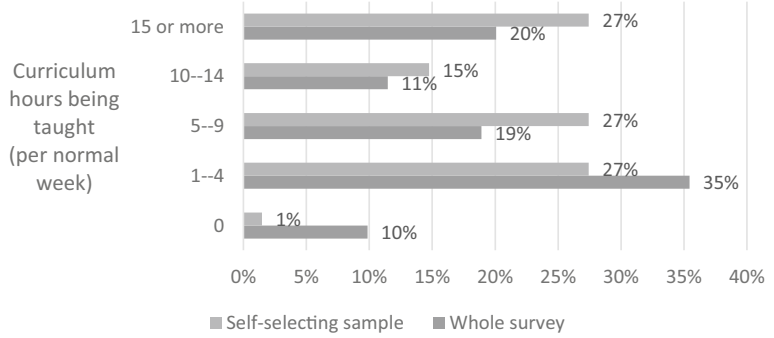

Fig. 3 Number of hours of Computing being taught per week

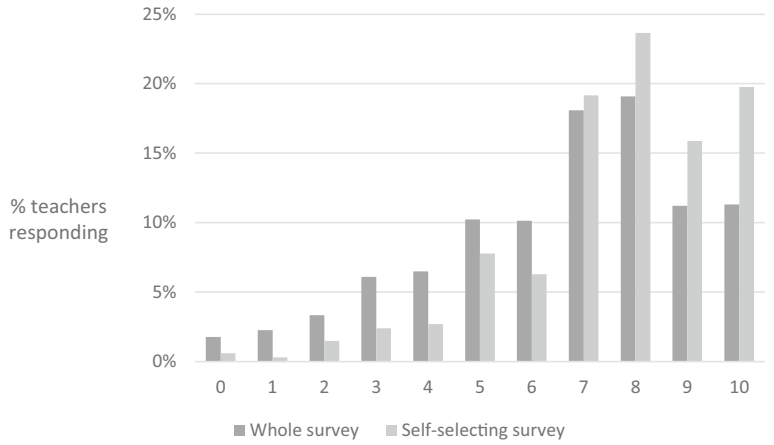

Fig. 4 Responses to "How confident are you in teaching Computing (on a scale of 0–10)

teach "dry" topics. Challenges relating specifically to students included difficulty understanding topics, problem-solving skills, resilience, and mathematical aptitude and literacy skills. In terms of resources, the problems seemed to be the difficulty finding resources, although this has got much better in the time that has passed since the survey was carried out, but also being able to identify what was a good or appropriate resource. Technical difficulties around networks, installation of software and flexibility of technicians were described as challenges by many teachers. Across these categories, the five most commonly-mentioned challenges facing teachers facing Computing are as follows:

- Teachers' own subject knowledge
- Students lack of understanding of content
- Technical problems in school
- Differentiation to meet different levels of ability
- Students willingness or ability to problem solve

Typical quotes from teachers illustrate these points.

3.1.1 Challenges for teachers

The analysis of teachers' qualitative responses indicates that teachers were concerned about the depth and breadth of their own Computing subject knowledge, and in particular that of computer science and programming. Subject knowledge skills are obviously one of the areas that we are focusing on in England (Sentance et al. 2013) in order to support teachers in feeling confident about their subject knowledge. Teachers express the worry "…*that my own subject knowledge is not always secure*". They also report that they have spent hours of their own time trying to upskill in the subject:

> "*...the sheer time involved in learning the language, skills. I do self CPD daily, and have given easily 100+ hrs of my own time to building my own skill set up...* "

Table 6 Challenges: most commonly occurring themes

Challenges	Teacher challenges	Number of cases with mentions
Challenges relating to teachers	Subject knowledge	97
	Differentiation	59
	Lack of time	53
	Approaches to teaching	52
	Dry (difficult to teach) topics	33
	Assessment	25
	Lack of support	20
Challenges relating to students	Students not understanding	76
	Students and problem solving	59
	Students' resilience	46
	Students not engaged	40
	Students ability in Maths	26
	Students literacy skills	14
	Students not remembering	13
	Students not practising	13
Challenges relating to resources	Technical problems	61
	Lack of resources	43
	Finding good quality resources	22

Teachers report that they have attended many training courses to build up their knowledge but still lack confidence in solving problem that students would come across:

"*At the moment it is my own underpinning knowledge about the construction of solutions to problems. I have worked through several training booklets and courses but it is just the ability to solve problems that the students would come across in the system that they are using.*"

Differentiation is also a concern for teachers. In some cases this is because students have differing experience of programming:

"*The different abilities of students especially when they come in from primary… some are well-versed in graphical programming environments.*"

In other cases, teachers referred to some students progressing faster than others and the gap between their students widening, making more differentiation necessary:

"*The gap between those that engage and achieve very quickly grows at an alarming rate. When introducing block coding in Scratch to my class, all were unfamiliar with anything like this, I had some pupils baffled and many self-exploring. I have found the ability gap to be much bigger than any other subject or topic and it seems to be down to the way in which the children think.*"

"*Pupils understand at different pace so trying to keep the high flyers occupied and engaged without losing the less able pupils...*"

Teachers need to find pedagogical approaches that support their students. They express the challenge of developing, promoting and sustaining problem-solving strategies and techniques amongst the students they teach. They demonstrate their desire to find ways to help students work through problems rather than give up:

"*…finding ways to encourage pupils to logically think through their problems, rather than ask for assistance at the first sign of difficulty.*"

Teachers expressed concern regarding how they should formatively assessing students and prepare students for summative assessment tasks:

"*I have very little experience of teaching that will prepare pupils for an exam... nearly everything I have ever taught has been coursework based, so I am not confident my lessons will arrive in pupils' memories!*"

Some teachers were worried that the mechanisms for assessing progress were not being embedded with students following instructions to learn programming:

"*There is little guidance on assessment and I fear that many schools will just head down a 'death by scratch' approach with some children simply following instructions*"

Other teachers expressed the view that some topics were difficult to teach because they were dry. If the teacher has recently learned the topic themselves they may not have the background knowledge to bring a topic to life that a more knowledgeable practitioner would have:

"*…some of the theory is quite dry so finding ways to make it interesting to students is a challenge.*"

3.1.2 Challenges related to students

In many of the responses teachers describe that students lack a basic understanding and can want to progress to work on projects beyond their ability:

"*In addition, many students want to jump before they can crawl and want to be developing complex programs without any understanding of the steps en route*"

Another variation of this problem is that students cannot apply what they have learned to new problems or tasks:

"*Linking the theoretical concepts to the practical application of those concepts*; *students tend to try to learn* '*rules*' *and then cannot apply the theory to their practical work.*"

Some teachers were very specific about exactly what students did not understand, in this case the concept of a variable:

"*Getting across the concept of a variable and why / how variables are used is always a challenge - simple metaphors help*, *but this is perhaps the biggest hurdle pupils have. Bridging the gap from graphical programming* (*Scratch etc.*) *to text-based programming is a challenge …*"

Teachers also related the fact that students sometimes struggled with the problem solving:

"*Pupils find it very difficult to think computationally. Breaking a large problem in to smaller* '*chunks*' *is not something which comes easily to them.*"

Teachers were unhappy with students' problem-solving skills when it came to a new problem that they had to solve independently:

We also find students' problem solving skills are under-developed. So we could as a student to code a FOR loop to iterative for 1000 times. Or to create an IF statement, *etc. They are fine with everything individually. However*, *ask them to create an algorithm to solve a problem*, *which involves everything they have learnt*, *it can be a problem.*

Although not frequently mentioned, some teachers referred to literacy difficulties as a stumbling block when learning to code:

"*Literacy is a big issue when teaching Computing*; *this has been the main stumbling block when trying to introduce variables*, *functions etc.* "

Fewer teachers mentioned students not understanding the theory than the programming, but there were still some concerns about topics and students' engagement:

"*Keeping pupil interest going and maintaining their attention*; *weaker pupils coping with mathematical concepts*, *such as binary*; *tying the different elements together to ensure pupils understand how they mesh*, *e.g. relevance of binary code*, *logic gates.*"

Gender was also highlighted by a few teachers, in terms of engaging more girls in Computing:

"*We need to let students know there is more to Computing than* [*company name*] *and controlled networking environments*, *they need to learn to be responsible for their actions on the computers and understand how their actions*

can be traced - because one day that might be the job they are doing. I feel this will get more girls interested and keep more boys focused and give students a much more realistic view of what Computing is all about. It is a very practical, hands on subject and we need to be reflecting this in our teaching style."

Many teachers identified the problems that their students had with having the resilience to keep trying when something did not work:

"*Students giving up easily. Not wanting to check their code. Finding it too difficult and not being prepared to "find out for themselves". Students are too spoon fed in ICT, and other subjects, and appear not to have the thinking skills required for this subject. I have even had quite a few students leave my CS course because they found it hard, and gave up!*"

Teachers talked about the perceived challenge of motivating students who lack confidence, capability and competence in mathematical concepts (for example, Boolean Algebra, logical operators) and manipulating numbers. Also, teachers expressed the challenge of encouraging students to successfully manipulate different numbers systems independently (i.e. binary and hexadecimal) which they have not encountered previously:

"*Maths abilities of students also appears to have a big influence on their understanding of logic and sequencing.*"

The analysis of teachers' qualitative responses highlights the challenges of students reading, analysing and synthesising problems in order to abstract the essential data to solve a problem. In addition, a number of teachers expressed their concern about students not being able to read a section of code and detect grammatical, logical and syntactic errors that exist within that code. Teachers identified the need to encourage students to develop, embrace and articulate a common Computing vocabulary so that students are aware of technical keywords and use them correctly.

Teachers expressed the challenges of identifying successful strategies to engage students in the Computing classroom, emphasising an understanding of the ubiquitous nature of Computing and the issue of disaffected and disengaged female students who do not see the relevance of the subject to them.

We also see the challenge of encouraging students to develop their mastery of the subject, and in particular coding, outside of lesson times.

"Pupils struggle to remember what has gone before without lots of practice"

Teachers are concerned about how to manage the expectations and aspirations of their students in what they can practically achieve in creating digital artefacts with the subject knowledge, capability and competence that students possess.

"*Students do not realise nature of subject before they start and sometimes find it very challenging*"

3.1.3 Challenges relating to resources

The analysis of teachers' qualitative responses highlights a variety of resource-related challenges which include possessing adequate hardware and software resources to teach the subject, sufficient funding to purchase resources for a new subject, and software resources correctly installed, configured and maintained to run correctly on the platform that the school operates. For example, a teacher commented that there is a "*Lack of resources but CAS is helping to change that*".

Teachers expressed their frustration of a perceived lack of support and understanding from their managers of the complexities of teaching Computing and unwillingness on the part of technical staff to contemplate installing software that they consider could compromise the integrity and security of the school's computer network.

A number of teachers conveyed their concerns regarding technical staff reluctance to maintain and troubleshoot installed software on either the computer network or standalone computers, for example describing "*technical difficulties with getting software to work on the school network*".

Overall, there were some differences between secondary and primary teachers in their responses about difficulties. More primary teachers (40 %) mentioned lack of subject knowledge in computer science being a difficulty than secondary (26 %). More secondary teachers (17 %) mentioned differentiation being problematic compared to primary (7 %). More secondary teachers (10 %) said their students lacked resilience than primary (5 %).

3.2 Successful strategies used by teachers

In this section we address the strategies that teachers report that they use. A set of coding themes was developed as shown in Table 1, and, of these, Table 7 shows the 20 most commonly occurring themes.

Table 7 shows that teachers emphasised unplugged, hands-on, contextualised activities and the importance of lots of practice. Approximately the same number of teachers mentioned working on tasks away from the computer as mentioned a particular software package that they used. The study looks entirely at free text comments with suggestion within the question; there are themes emerging quite clearly from this data around using activities away from the computer that promote understanding. These will be discussed in more depth in the next section.

Most of the individual strategies suggested by teachers could be grouped into a series of five themes, which are (in no particular order):

- Learning away from the computer (unplugged-style)
- Collaborative working
- Computational thinking
- Contextualisation of learning
- Scaffolding programming tasks

Table 7 Strategies: most commonly coded themes

Strategy	Number of cases coded at this node	Strategy	Number of cases coded at this node
Unplugged strategies	70	Peer mentoring	32
Reference to particular software	55	Use of examples	32
Relate to real world activities	49	Algorithms (link to)	29
Use variety of activities	49	Demonstration and modelling	27
Lots of practice/Little and often	47	Learn through examples	27
Use hands-on experiences	45	Contextualisation	25
Break down/decomposition	40	Use of videos	25
Code manipulation	38	Team coding/pair programming	24
Scaffolding	38	Develop troubleshooting skills	23
Emphasise problem solving	32	Collaboration	22

The way we have moved from coded themes to these five areas of focus can be seen in Table 8.

Each of these areas of focus will be exemplified in turn.

3.2.1 Unplugged-style or kinaesthetic activities

A significant proportion of teachers mentioned, unprompted, that they try to support students' understanding by using physical, or unplugged-style activities in the classroom. One teacher gave two examples of teaching different topics using physical visual-aids to support the learning:

> "*For example I use clear plastic drinking cups as memory locations and label them as variables or when demonstrating an algorithm like bubble sort add data* (*on pieces of paper*)."

Many of these activities are designed to promote both collaboration and computational thinking skills. In fact, whether the activity takes place on the computer or not may not be what is interesting. The key link between the statements made by teachers seemed to be their impression that actually physically being engaged in the activity was conducive to the students' learning. This is an area which would benefit from further research.

3.2.2 Collaborative working

The analysis of teachers' qualitative responses highlights a variety of collaborative working strategies that they use within the classroom and would promote to other computer science teachers. These collaborative strategies include: team work, peer mentor, paired programming and collaboration. These strategies resonate with the concept of computational participation (Kafai and Burke 2014) and

Table 8 Coded themes linked to key strategies

General theme	Coded theme	Number of cases
Unplugged/practical activities	Unplugged strategies	70
	Hands-on experiences	45
Collaborative work	Team coding/pair programming	24
	Peer mentoring	32
	Collaboration	22
Computational thinking	Break down/decomposition	40
	Problem solving	32
	Algorithms	29
Scaffolding programming tasks	Scaffolding	38
	Code manipulation	38
Contextualisation	Relate to real world activities	49
	Using examples	32
	Learn through examples	27
	Contextualisation	25

strategies proposed to develop this within the classroom. In addition individual teachers commented on the positive motivational impact that collaborative working has on individuals, small groups and the class itself.

"*…Developing digital leaders in students who can support others…*"

In this context, digital leaders are pupils who are good at working with technology and can support others in the school. Teachers spoke about working on problems together and also programming together:

"*Decomposing sample problems together as a class then team-coding …they can use peers for discussion of specific problems. …*"

3.2.3 Computational thinking

Analysis of teachers' qualitative responses indicates a number of computational thinking concepts and processes that teachers want to promote and develop their students' competence in through using a variety of teaching and learning activities. These concepts and processes include: logic (algorithmic) thinking, decomposition, problem solving and abstraction (Brennan and Resnick 2012).

There were many references to breaking down problems in the analysed data, for example:

"*Breaking down the problem then breaking it down again then breaking it down again... …*"

In some cases, computational thinking skills such as abstraction were explicitly mentioned by teachers:

"*Organise the learning so that the pupils develop their programming skills using decomposition and abstraction. ….*"

3.2.4 Contextualisation of learning

Teachers talked about relating Computing content to other aspects of the curriculum; they give examples of both relating what is being learned in Computing to other subjects taught at school and also to concepts from home (so relating to real-life). The quote below is a typical example:

"*Scale it back to basics and use real-life examples for the activities e.g. making tea. Use lots of visual aids to help pupils and online resources to help scaffold activities.*"

3.2.5 Scaffolding programming tasks

Closely related to the theme of computational thinking are the strategies that teachers use to help their students understand program code. One teacher described a range of types of strategies used to support students learning programming, that involve:

"*… giving code on paper not electronically, so they have to type it in, think about what they are typing and fix the errors that occur when trying to compile the program …*"

Another teacher described using trace tables to help students understand the flow of control and changing value of variables within a program:

"*Discussion of what a specific algorithm does, then running trace tables on small programs …*"

Other strategies described included "scaffolding" as the student is given part of a program to extend, and programs to debug. Typing in code to give more chance that the program would work, but involving debugging errors caused by transcription errors is another supportive strategy for early programmers reported by teachers.

4 Discussion

Teachers reported a range of different challenges that they faced when teaching Computing. Some of the challenges mentioned relate to the teachers' own difficulties – for example, not being confident in the subject matter or not being able to differentiate

sufficiently for a mixed-ability group, and other comments focus on the fact that the students have difficulty understanding the material and in problem solving.

4.1 Intrinsic and extrinsic challenges

The work of Finger and Houguet (2009), described in Section 1.4, around intrinsic and extrinsic challenges helps us to analyse the findings from our teachers. Some of the challenges for teachers are extrinsic such as lack of resources; others such as their own subject knowledge of understanding of appropriate pedagogy are intrinsic. We find that also we can also divide the areas of challenge for *students* as intrinsic or extrinsic to the students too. Teachers report that students may be challenged by low mathematical ability (intrinsic) for example, or lack of opportunity to practise (extrinsic). This gives us a framework in which to examine the challenges that we identified. This can be seen in Table 9 where we show a range of perceived or encountered challenges reported by teachers.

Teachers may be working to improve on the challenges that they face that are intrinsic to them, for example their own subject knowledge and teaching approaches, but be frustrated by the challenges over which they have less control, such as time in the curriculum or technical support. Teachers report on challenges that we have defined as intrinsic to the students; in the way that these are presented, they appear to be described as extrinsic to the teacher. The teachers report the students' lack of understanding of the subject or lack of engagement as a problem to them. However the development of strategies by the teacher to overcome the difficulties experienced by the student may be within their reach as their experience of teaching Computing increases.

The question remains as to what *schools* can do to address these challenges. Clearly teachers need more training (Sentance et al. 2013) and to develop confidence in their pedagogical skills with relation to Computing. However there is much that could be done by schools to alleviate the challenges facing teachers in the context of curriculum change. Teachers need more support from their headteachers and leadership teams to implement new courses and develop resources. They need access to robust technical solutions through their school's technical support departments. Students need time to

Table 9 A framework for viewing challenges reported by teachers

	Challenges	
	Intrinsic	Extrinsic
Teachers	Subject knowledge Differentiation (skills in) Approaches to teaching topics (pedagogy)	Assessment Resources Lack of support Lack of time Technical problems
Students	Mathematical aptitude Literacy skills Resilience Problem-solving skills Not understanding Engagement	Time to practise School and others' expectations

practice the new skills without initially facing unrealistic expectations while the new subject is bedding in. All these things are possible, although there may be financial restrictions on schools' ability to implement them.

However building students' resilience and ability to learn from mistakes may take more time. Developing problem-solving and other computational thinking skills is challenging when the subject is mandatory for *all* students in the school rather than a small self-selecting group as an extra-curricular option. Building learners' confidence is key and understanding and then celebrating small successes will be important.

Suggested strategies to address these challenges will be discussed in the next section.

4.2 Strategies adopted by teachers

Examining the statements of teachers as they report what **strategies** work well for them in teaching Computing has enabled us to draw out particular themes. Section 3.2 described five key themes identified in the analysis of teacher responses:

1. Unplugged type activities
2. Contextualising activities
3. Collaborative learning
4. Developing computational thinking
5. Scaffolding programming tasks.

Ben-Ari (1998) advised teachers: "*Do not run to the computer*", and it seems that teachers are taking this advice in using a variety of other strategies to get concepts across. In addition, the use of collaborative work, peer mentoring, pair programming and other strategies is helping teachers to establish computational thinking skills in young students. What is clear is that there is a change for teachers.

4.2.1 Change in teaching strategy

One of the main outcomes of this study is that the ways in which computer science elements of Computing are taught are different to methods previously used in delivering ICT. Teachers have provided a range of approaches that they use that are different to those that they previously used delivering ICT in the curriculum. They also describe that students also have to adapt to new ways of teaching and different types of content, particularly older students who had been used to the same teachers delivering a different style of lesson.

One teacher described being a teacher in this subject as requiring a whole new style of teaching:

> [*It is*] " *a whole new style of teaching as for the past 10 years I have been demonstrating skills in... software, and getting pupils to show evidence they have learnt those skills. With Computing I actually have to teach, and get difficult concepts across, and need to get out of the habit of simply showing pupils how to do something and then asking them to do the same thing. I am finding it hard not to just show how to solve a programming problem, and instead teach pupils to think for themselves*."

In our particular context, teachers who are delivering Computing are mostly those who have been delivering ICT[2] for many years, which has focused on learning software applications, and on individual work at the computer producing digital artefacts and evaluating existing products. Clearly the teaching style for teaching computer science principles is different for teachers. It is also different for students too, and a teacher reported that "*Some students find the change from the old ICT a massive leap*" with another reporting that the problem-solving element had not been present in many ICT lessons:

> "*Often the subject is viewed as ICT and most students do not know the difference. For many it comes as a shock that there is a lot of problem solving as opposed to ICT*"

Some of the teaching methods suggested by teachers can be identified as constructive activities, based on the discussion in Section 1. Constructivism, based on students' active participation in problem-solving and critical thinking, has profoundly influenced the teaching of programming (Ben-Ari 1998). It implies a need for authentic and meaningful experiences to support learning based on prior experiences and models of the world. The following activities, drawn from the suggestions by teachers, would all support a constructivist pedagogy:

- Active learning experiences which involve the student (for example, unplugged, kinesthetic activities)
- Learning by exploration (open-ended tasks, exploring programming environments)
- Learning by solving problems (self-directed projects, problem-solving)
- Using examples that are relevant to students' own experiences (relating to real-world experiences)
- Open-ended discussion and working in groups (group tasks, team problem-solving).

Whether the teacher is teaching history, science or computer science, these teaching methods would form part of a teacher's pedagogical content knowledge. One of the teachers indeed commented on borrowing strategies from other subject areas:

> "*I've observed people teaching other subjects e.g. Maths and History and found that they have great strategies for managing learning content!*"

4.2.2 Developing computational thinking

There are some aspects of computer science which may require specific approaches to teaching that do not necessarily transfer as well from other teaching experiences. Primarily this relates to supporting students with their computational thinking skills. Some guidance on computational thinking developed for the English curriculum (Csizmadia et al. 2015) suggests a range of strategies including "*breaking down into*

[2] Information and Communication Technology

component parts (*decomposition*), *reducing the unnecessary complexity* (*abstraction*), *identifying the processes* (*algorithms*) *and seeking commonalities or patterns* (*generalisation*)" (p. 9).

It may be useful to give this type of guidance to teachers on how to develop computational thinking skills in students. There is clearly an overlap between the constructivist influences on pedagogy, which have a long history, and the more recent emphasis on facilitating computational thinking skills through the teaching of computer science: in essence the suggestions for actual teaching activities derive from different motivations but may result in similar or identical activities in the classroom. Types of specific activities that would form part of specific pedagogical content knowledge to computer science teaching may include:

- Manipulating/comprehending/tracing code (teaches evaluation and generalization)
- Breaking down code/concept (teaches decomposition)
- Developing algorithms (teaches algorithmic thinking, abstraction and evaluation).

4.2.3 The need to develop resilience

Debugging or troubleshooting skills are also important to computer science. To be able to develop these, students need more than just computational thinking skills. They need patience and particularly resilience, which is something that many teachers who participated have highlighted was lacking for their students:

> "*If there is a problem*, *they want the teacher to correct it for them... they do not have to get it correct first time*, *but they do need the skills to be able to correct it and see why there was a mistake in the first place.*"

This is one area of difficulty that was not readily addressed in strategies suggested by teachers. Being able to "deal with adversity", "keep trying" or "learn from our mistakes" is obviously a life skill for all of us. However, this is particularly important when learning computer programming (perhaps less so in other areas of computer science theory). As all programmers know, learning to troubleshoot errors, and learning from errors, is the prime way of making progress in the acquisition of programming skills. This is a mindset that is alien to a lot of children in school, who are taught to seek success, not failure.

In this study only three of 336 teachers commented that they had strategies to support students in this type of activity. The three teachers suggested how they personally dealt with students needing to build up their resilience to failure:

> "*I encourage pupils to have a go*, *make mistakes and then try to work out for themselves what to do when things don't work the way they want*"

> "*Students tend to understand the process better if they work through wrong answers*, *rather than being given program code each time*"

> "*Letting them have a go and allowing them to get it wrong. They need to become problem solvers, not just the teacher telling them what to write. The students must code for themselves without too much intervention from the teacher in terms of theory. There has to be some of course, but after they have tried to figure it out first*

Given that lack of resilience is a commonly mentioned challenge, it could be recommended that teachers focus more on strategies that build resilience in learners when they teach programming.

4.3 Limitations of the study

The intention in the study was to identify particular strategies for teaching programming and those for teaching other aspects of Computing, including theory, and similarly with challenges. However we were not able to do this, due to the nature of the data, as reported in Section 2.2. Some teachers described how they taught programming under the non-programming question and vice versa. This is a limitation because it means we have not been able to reliably compare strategies for different areas of the curriculum. Thus the proportion of answers relating to scaffolding programming tasks, which did not generally occur as answers to the non-programming question, may have been even higher had we been able to look at programming strategies alone. This could have possibly been avoided by more explicit signposting in the survey.

The teaching approaches described by teachers in this study are not claimed to be representative of all teachers teaching the Computing curriculum in England. The survey was promoted primarily through the Computing At School (CAS) website which has a large number of teacher members who teach Computing, in addition to a direct email to Computing At School members and use of social media. The four questions reported on in this paper were optional and teachers were asked if they wished to take part in a particular part of the research before being shown these questions. This means that the group of teachers answering are not only self-selecting, but they are also a group of teachers that are enthusiastic about teaching Computing and we have already reported that they demonstrate self-efficacy. Thus this study does not in any way attempt to claim that these strategies and challenges relate generally to all teachers of Computing.

The teachers participating in the survey are, in the majority, members of CAS, and as such have access to a lively and supportive grass-roots community of teachers with whom they can exchange ideas and classroom resources. The presence and nature of this community of practice may well have an impact on the commonality between the approaches teachers are successfully using, as resources are freely shared and adapted and strategies for teaching discussed in face-to-face sessions such as hub meetings. The possible limitation of this is the emphasis on unplugged activities in particular may be influenced by the amount of resources, posts and magazine articles on this topic that CAS members have access to. The community suggests good practice and teachers act on their suggestions. However, the fact that teachers mention use of these strategies implies that they have found them useful in the classroom in supporting learning.

We are not able to provide evidence for which of these suggested approaches is more effective in helping students to learn without more empirical research; thus

another useful angle following on from this study would be to examine students' own perspectives on how Computing is taught. This is recommended as an area of further investigation.

As another area of further investigation, we suggest that more research is carried out on the implementation of these strategies for learning Computing and the impact on students' learning. The five themes identified here offer a clearly defined set of approaches to examine further.

5 Conclusion

This study has reported on a survey of Computing teachers in 2014 where a large self-selected sample (N = 339) answered free text questions about successful teaching approaches that they used for teaching Computing and challenges that they faced. The participants were self-selecting and mostly reported themselves as being confident in their delivery of Computing - thus the data gives us reports of good practice.

We have been able to identify a number of themes emerging from this data and also pinpoint where challenges may be intrinsic or extrinsic to the teacher. We hope that the strategies suggested here may help those teachers who are still moving into Computing teaching themselves. The discussion in this paper can contribute to work in teacher education to support teachers who are beginning to teach Computing – as more countries move to introduce the subject in the curriculum.

Overall, given that we would like this study to be able to offer guidance to teachers on how develop their Computing teaching skills, we can suggest that one specific focus should be on supporting students to develop resilience and the ability to learn from mistakes. Together with other strategies suggested by the teachers whose comments are reported here, this may lead to an easier acquisition of programming skills, and subsequently computational thinking skills.

References

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads, 2*, 48.

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), 20–29.

Ben-Ari, M. (1998). *Constructivism in computer science education. Proceedings of the twenty-ninth SIGCSE technical symposium on computer science education*. Atlanta, Georgia, United States: ACM.

Black, J., Brodie, J., Curzon, P., Myketiak, C., McOwan, P. W., & Meagher, L. R. (2013). Making Computing interesting to school students. *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (*ITiCSE* '*13*). Canterbury, UK. 255. doi:10.1145/2462476.2466519

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, *Vancouver*, Canada., Vancouver, Canada.

Brown, N., Kölling, M., Crick, T., Jones, S. P., Humphreys, S., & Sentance, S. (2013). Bringing computer science back into schools: Lessons from the UK. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (*SIGCSE 2013*), 269–274.

Brown, N., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: the resurgence of computer science in UK schools. *ACM Transactions on Computing Education, 14*(2), 9.

Bruner, J. (1996). *Towards a theory of instruction*. Cambridge, MA: Harvard University Press.

Csizmadia, A., Curzon, P., Dorling M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. (Available at: http://community.Computingatschool.org.uk/resources/2324). Unpublished report.

Curzon, P., McOwen, P., Cutts, Q., & Bell, T. (2009). Enthusing and Inspiring with Reusable Kinaesthetic Activities. *Proceedings of the ITICSE* 2009,

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

Dewey, J. (1938). *Experiential education*. New York: Collier Books.

Diethelm, I., Hubwieser, P., & Klaus, R. (2012). *Students, teachers and phenomena: Educational reconstruction for computer science education*. Koli, Finland: ACM.

Ertmer, P. A., & Ottenbreit-Leftwich, A. T. (2010). Teacher technology change: how knowledge, confidence, beliefs, and culture Intersect. *Journal of Research on Technology in Education, 42*(3), 255–284.

Finger, G., & Houguet, B. (2009). Insights into the intrinsic and extrinsic challenges for implementing technology education: case studies of Queensland teachers. *International Journal of Technology and Design Education, 19*(3), 309–334.

Grover, S., & Pea, R. (2013). *Using a discourse-intensive pedagogy and Android's App Inventor for introducing computational concepts to middle school students*. Proceedings of the 44th SIGCSE technical symposium on Computer science education. ACM.

Kafai, Y. B., & Burke, Q. (2014). *Connected code why children need to learn programming*. Cambridge MA: MIT Press.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37.

Lister, R. (2011). Concrete and other neo-piagetian forms of reasoning in the novice programmer. *Proceedings of the Thirteenth Australasian Computing Education Conference - Volume 114*, Perth, Australia. 9–18.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, Leeds, United Kingdom. 119–150.

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. *Proceedings of the Fourth International Workshop on Computing Education Research*, Sydney, Australia. 101–112. doi:10.1145/1404520.1404531

Mayring, P. (2000). Qualitative content analysis. *Forum of Qualitative Social Research, 1*(2), Art. 20. http://nbn-resolving.de/urn:nbn:de:0114-fqs0002204

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., & Kuno, Y. (2009). A CS unplugged design pattern. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (*SIGCSE '09*), 231. doi:10.1145/1508865.1508951

Papert, S., Harel I. (1991). *Constructionism*. NY: Ablex Publishing.

Papert, S. (1996). An exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95–123.

Piaget, J. (1950). *The psychology of intelligence*. Cambridge, MA: Harvard University Press.

Selby, C. C. (2012). Promoting computational thinking with programming. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (pp. 74–77). ACM.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351–380.

Sentance, S., Dorling, M., & McNicol, A. (2013). Computer science in secondary schools in the UK: Ways to empower teachers. In I. Diethelm, & R. Mittermeir (Eds.), *Informatics in schools*: *Sustainable informatics education for pupils of all ages. Lecture notes in computer science* (pp. 15–30) Springer-Verlag, Berlin.

Shulman, L. (1986). Those who understand: knowledge growth in teaching. *American Educational Review, 15*(2), 4–14.

Thompson, D., & Bell, T. (2013). *Adoption of new computer science high school standards by New Zealand teachers* ACM.

Thompson, D., Bell, T., Andreae, P. and Robins, A. (2013). The role of teachers in implementing curriculum changes. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM

VAN DEN Akker, J. (2003). Curriculum Perspectives: An Introduction. In *Curriculum Landscapes and Trends* (pp. 1–10). Springer Netherlands.

VAN Gorp, M. J., & Grissom, S. (2001). An empirical evaluation of using constructive classroom activities to teach introductory programming. *Computer Science Education, 11*(3), 247–260.

Viera, A. J., & Garrett, J. M. (2005). Understanding interobserver agreement: the kappa statistic. *Family Medicine, 37*(5), 360–363.

Webb, H., & Rosson, M. B. (2013). Using scaffolded examples to teach computational thinking concepts. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 95–100). ACM.

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

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011, March). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 465–470). ACM.

VAN Dyne, M., & Braun, J. (2014). Effectiveness of a computational thinking (cs0) course on student analytical skills. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 133–138). ACM.