Computer Science in the School Curriculum: Issues and Challenges

Mary Webb, Tim Bell, Niki Davis, Yaacov Katz, Nicholas Reynolds, Dianne Chambers, Maciej Syslo, Andrew Fluck, Margaret Cox, Charoula Angeli, et al.

Computer Science in the School Curriculum: Issues and Challenges

Mary Webb[1], Tim Bell[2], Niki Davis[2], Yaacov J. Katz[3], Nicholas Reynolds[4], Dianne P. Chambers[4], Maciej M. Sysło[5], Andrew Fluck[6], Margaret Cox[1], Charoula Angeli[7], Joyce Malyn-Smith[8], Joke Voogt[9], Jason Zagami[10], Peter Micheuz[11], Yousra Chtouki[12] and Nataša Mori[13]

[1]King's College London, UK
mary.webb@kcl.ac.uk, mj.cox@kcl.ac.uk
[2]University of Canterbury, Christchurch, New Zealand
tim.bell@canterbury.ac.nz, Niki.Davis@canterbury.ac.nz
[3]Michlala - Jerusalem Academic College and Bar-Ilan University, Israel
yaacov.katz@biu.ac.il
[4]University of Melbourne, Australia
nreyn@unimelb.edu.au, d.chambers@unimelb.edu.au
[5]UMK Toruń, University of Wrocław, Poland
syslo@mat.umk.pl
[6]University of Tasmania, Australia
andrew.fluck@utas.edu.au
[7]University of Cyprus, Cyprus
cangeli@ucy.ac.cy
[8]Education Development Center, USA
jmsmith@edc.org,
[9]University of Amsterdam, Netherlands
J.M.Voogt@uva.nl
[10]Griffith University, Australia,
j.zagami@griffith.edu.au
[11]Alpen-Adria-University of Klagenfurt, Austria
peter.micheuz@aau.at
[12]Al Akhawayn University in Ifrane, Morocco
Y.Chtouki@aui.ma
[13]University of Ljubljana, Slovenia
natasa.mori@fri.uni-lj.si

Abstract. This paper is based on analysis and discussion undertaken over several years by researchers, policymakers and practitioners from a range of countries which vary in their approaches to the curriculum for Computer Science. The discussions, undertaken predominantly within the International Federation of Information Processing (IFIP) and EDUsummIT communities were motivated by a need to examine the rationale, issues and challenges following some concerns across the globe about the position and nature of Computer Science in the school curriculum. We summarise our findings and focus specifically on challenges for the computer science education community in communicating, clarifying needs and promoting curriculum change in order to encourage Computer Science in the curriculum both theoretically and practically.

Keywords: Computer Science, curriculum rationale, international perspectives, Informatics

1. Introduction

This paper introduced the symposium: "From Curriculum Visions To Computer Science And Computational Thinking In The Curriculum In Practice" [1] at the World Conference on Computers in Education. The paper is based on an analysis and discussion of the rationale, issues and challenges for Computer Science in the school curriculum (K-12) that was initiated by the Curriculum Task Force of the International Federation of Information Processing[1] (IFIP) and continued at EDUsummIT 2015[2] as well as by IFIP meetings and conferences. We summarise and focus specifically on challenges for the computer science education community in communicating, clarifying needs and promoting curriculum change in order to encourage the realisation of the roles of Computer Science in the curriculum both theoretically and practically. The discussions have involved experts from many different countries and the analysis has focused in particular on a comparison across seven countries: Australia, Cyprus, Israel, New Zealand, Poland, UK and USA. The situation of the curriculum for Computer Science varies between these countries. In some, e.g. Cyprus, Poland and Israel, Computer Science has existed as a curriculum subject for many years. For others the curriculum for Computer Science has recently been substantially revised after a period of neglect followed by calls for reform [2-4]. Even in those countries where Computer Science in the curriculum has a long history, there are differences in approach and in the importance of various factors that affect curriculum design and implementation. Thus our discussion, based on this range of experiences led to a rich range of issues and considerations and a set of questions, some of which we were able to address and others remain as challenges.

When discussing the curriculum for Computer Science, the need to identify an acceptable working definition for Computer Science as a curriculum subject was a key consideration. Some popular definitions [5] are:

1. It seeks to answer the following questions: What is information? What is computation? How does computation expand what we know? How does computation limit what we can know? [6].
2. The study of computers and algorithmic processes, including their principles, their hardware and software designs, their implementation, and their impact on society. [7].
3. the scientific and practical approach to computation and its applications and the systematic study of the feasibility, structure, expression, and mechanization of the methodical procedures (or algorithms) that underlie the acquisition, representation,

[1] IFIP is the leading multinational, apolitical organisation in Information and Communication Technologies and Sciences; IFIP was originally set up under the auspices of UNESCO and continues to have a formal consultative status within UNESCO.

[2] EDUsummIT is a global community of researchers, policy-makers and educators committed to supporting the effective integration of Information Technology (IT) in education by promoting active dissemination and use of research.

processing, storage, communication of, and access to information (Wikipedia 2017).

In addition, definitions based on areas of knowledge can be found through widely adopted curricula, such as the ACM/IEEE Computer Science curriculum for undergraduates [8].

We favoured the Wikipedia definition as reflecting more recent consensus and current practice, so we will use that definition throughout this paper. The Wikipedia definition captures the key idea that the function of any physical computing device can be abstracted to applying algorithms to data, and hence this definition captures the key elements of the technical side of computing.

In our exploration of the definition of Computer Science as a curriculum subject we also recognised that the name given for the curriculum subject varies across different countries. For example, Informatics, as a curriculum subject, is slightly broader than Computer Science, but is the term used widely across Europe to refer to this discipline. The Joint Informatics Europe & ACM Europe Working Group on Informatics Education use the term Informatics to "cover the entire set of scientific concepts that make information technology possible" [2 P. 9]. We also recognise the importance of other aspects of computing and computer use such as digital literacy, which may be linked into a curriculum grounded in the academic discipline of Computer Science. In this paper we have focused only on the underlying scientific discipline in order to examine its importance and we have used the term Computer Science in order to restrict the definition.

In this paper we first summarise the challenges and solutions identified at EDUsummIT 2015. Then we explain: the rationales for incorporating Computer Science in the curriculum; arguments regarding the position of Computer Science in the curriculum; the nature of curriculum design and the specific major challenges for designing and implementing a Computer Science curriculum.

2. Key challenges and issues

Table 1 summarises the issues and solutions identified at EDUsummIT 2015 for advancing understanding of the roles of Computer Science in the curriculum. While the order of challenges shown in Table 1 represents a logical progression for considering curriculum rationale and design, the order of priority and difficulty will vary across contexts. Furthermore there are interrelationships between the issues identified in Table 1 as discussed in subsequent sections.

Table 1. Challenges and Solutions for Advancing Understanding of the Roles of Computer Science in the Curriculum [adopted from 9, p64]

Challenge	Solution/Recommendation to P, E, I, R
Key: Policy maker (P), Educator (E), Industry partners (I), Researcher (R)	
1. Lack of clear understanding (outside the field of Computer Science) of Computer Science as an academic discipline.	(a) Adopt a globally agreed statement of Computer Science as a discipline in its own right (P, I, R, E). (b) Articulate the nature, importance and relevance of Computer Science to society and education (P, I, R & E).
2. A need for Computer Science as a distinct subject in school curricula is controversial and poorly understood.	Disseminate and communicate a clear rationale to different stakeholders about the need to have Computer Science as a distinct subject in school curricula (P, I, R & E).
3. Computational thinking, a core component of Computer Science, is considered to be important in 21st century skills, but due to its complexity, it is difficult to implement in schools.	Promote computational thinking through the means of a Computer Science curriculum, which aims at making computational thinking commonplace (P, R & E).
4. The development of Computer Science school curricula is impeded by insufficient empirical evidence of student learning to support content definition and sequencing.	Design Computer Science curricula based on a content analysis, and continue to research students' learning difficulties as well as the effects of different pedagogical approaches. (E & R).
5. Previous ICT curricula deliveries poorly prepared students for Computer Science in further/higher education or professional employment.	Facilitate better smart partnerships between education systems and industry/professional associations. (E & I)
6. Integrating Computer Science across other subjects in school curricula has been ineffective.	Identify clear learning outcomes, assessments and standards for Computer Science. (E, I, P & R)
7. Teachers' professional development in a newly introduced Computer Science subject is a challenge in quality and quantity for many countries.	a) Encourage more Computer Science graduates to become teachers. (P, I & E) b) Add Computer Science specialisation to pre-service training for primary teachers. (P & I) c) Make Computer Science professional learning a requirement for periodic teacher re-accreditation/licensing. (P) d) Schools need resources to free teachers to undertake the professional learning and preparation for a new Computer Science subject. (P)
8. Identifying and allocating the additional resources for teaching Computer Science is a challenge.	(a) Some of Computer Science can be taught without computers. But computers can enhance the learning experience. (P, E, I & R) (b) Teacher training needs to provide skills in using the available resources in the most efficient way. (E) (c) Identify, and if not available, commission teaching support materials in mother-tongue language especially for younger students (P,E,I).

3. Rationales for Computer Science in the curriculum

Arguments for the inclusion of Computer Science in the curriculum are compelling, as evidenced by many countries adopting it as a mandatory part of their curriculum. The reasons were summarised at EDUSummIT 2015 as economic, social and cultural [5, 9]. The economic rationale rests not only on the need for a country to produce computer scientists to sustain a competitive edge in a world driven by technology but also on the need for Computer Science-enabled professionals in all industries to support innovation and development. The social rationale emphasises the value in society of a diverse range of active creators and producers rather than just passive consumers of technology. Such capability provides people with power to lead, create and innovate within society. The cultural rationale rests on enabling people to be drivers of cultural change rather than having change imposed by technological developments. Alongside these three rationales, we frame the inclusion of Computer Science in school curricula with respect to two further dimensions for evaluating its contribution [5]:

1. the beneficial context: the individual learner; society; humanity and the ecology upon which we all depend; and the wider universe.
2. the timescale for the benefits of the learning to be experienced: immediately; the lifetime of the individual learner; years within transformation of a social system; the expected duration of humanity or the lifetime of the universe.

Furthermore we claim that Computer Science is necessary for education because of its increasing importance for knowledge generation in a range of important areas of human endeavour. Computer Science is heralding new developments in many areas of science and technology and data science in particular, which links machine learning with programming skills. Moreover it is providing new methods for knowledge discovery [see for example 10].

Immediate broad educational benefits for students in learning Computer Science include potential benefits for thinking and problem solving. There is a long history of research into the benefits of learning programming for developing general thinking and problem solving skills [see for example 11]. The issue remains controversial and the debate has been enlivened recently by the revised focus on computational thinking [12, 13] which we argued is best developed through Computer Science including programming since 1) programming makes an excellent vehicle for students to explore the concepts in a concrete way and 2) implementing the concepts in a program provides a means for students to check their thinking. As their expertise in computational thinking develops, students would be expected to use their skills and build their understanding of applications of computational thinking via a range of examples across different subjects.

In summary the arguments outlined here present a strong case for the importance of Computer Science in the curriculum for 21st-century learning. Its importance for individual learners, in particular, leads to our recommendation that learning Computer Science is an entitlement for all school students and recent curriculum development takes this approach [14].

4. The position of Computer Science in the curriculum

Following on from the clarification above of Computer Science as an academic discipline and therefore the basis for a curriculum subject for schools, is the debate about how the curriculum should be organised. In most countries, at least at secondary level, the curriculum is subject-based. The major arguments for positioning Computer Science as a distinct subject in the K-12 curriculum were articulated at EDUSummIT 2015 as: 1) its importance as a disciplinary area as explained above; 2) the evidence that integrated approaches to curriculum delivery have failed to prepare students for higher education and 3) the importance of computational thinking, which we argued is best developed through Computer Science, including programming, and then built upon in other curriculum areas. Computational thinking involves developing ways of analysing and solving problems, designing systems and understanding human behaviour that draws upon concepts fundamental to Computer Science. Furthermore, while computational thinking is beneficial in many curriculum areas, and doesn't require programming at all, representing a solution to a problem as a program provides a way of evaluating the solution thus providing students with feedback and ways to proceed. Therefore we argued that computational thinking should be implemented through Computer Science learning including programming.

The challenges created by the limited empirical evidence for development of understanding and skills in Computer Science relate to the structure and organisation of the curriculum together with pedagogical considerations. Curricula design may be guided by epistemological considerations and other constraints [15, 16] and later informed by empirical evidence. Therefore, in spite of the limited empirical evidence on which to base curricular design, many Computer Science curricula exist and many countries have recently re-designed Computer Science curricula. An analysis of developments in five countries as well as a review and content analysis of curriculum reports [14] suggested key issues to consider when designing Computer Science curricula. First there is a consensus across various curriculum reports [2, 3, 17] about the key concepts and techniques of the discipline. However there is as yet no consensus about the importance of more general intellectual practices such as persistence in working through problems and tolerance for ambiguity as well as the importance of collaborative learning and social competence developed through group work. Furthermore an emerging consensus regarding the best starting age for Computer Science being young, about five to seven years old, came from a comparison of curriculum development in three out of the five countries examined: Poland, UK, Australia. The importance of a young age for starting to learn Computer Science was also an outcome from the panel discussion at IFIP TC3 Conference in Vilnius, 2015 [14]. The availability of programming environments and other software designed to support younger learners in learning programming was identified as one of the key factors that have supported this early development of Computer Science learning [18]. Arguments for starting Computer Science at an early age include: 1) learning programming is difficult but a consensus is emerging that learning some of the techniques, approaches and thinking involved in programming at a younger age enables more students to become successful in programming and 2) developing student self-efficacy in programming and Computer Science at an earlier age may reduce the gender gap [14, 18].

There are arguments that Computer Science might be taught from pre-school age but there is also a need for caution and further research into pedagogical issues [19]. While it is possible for pre-schoolers to engage with pre-programming systems and concepts from Computer Science, the real foundational material that is needed in Computer Science in early years are already typical of curricula. Examples include basic numeracy and literacy skills, learning to classify and sort objects, understanding sequences of events, working with patterns in numbers and other symbols, becoming familiar with physical directions such as forward/left/right, and social competency to be able to follow and give instructions or identify the needs of another user.

5. Structuring the curriculum

One of the constraints for curriculum design identified by Winch [15] is the need to introduce, early in the curriculum, all three major types of knowledge: concepts, propositions and know-how because these knowledge types are dependent on each other. One promising approach to addressing this constraint is a spiral curriculum, such as that developed in Poland [20] where at each level unified aims are addressed (see Table 2) but pedagogically the approach varies across three elements such that the first element is more important at lower levels and 2 and 3 become more important during progression:

1. problem situations, cooperative games, and puzzles that use concrete meaningful objects – discovering concepts
2. computational thinking about the objects and concepts – algorithms, solutions
3. programming

The concept of a spiral curriculum was put forward originally by Bruner [21] based on his cognitive theory in which in earlier stages of cognitive development manipulating real objects is important and later these may become more abstract representations. As the curriculum spirals upwards more complex concepts and approaches can be introduced. In line with Bruner's [21] proposals, benefits of such a spiral curriculum include: 1) reinforcement of key concepts and techniques each time the subject matter is revisited; 2) progression from simple concepts to more complex ones; 3) students can be encouraged to recap their previous knowledge and apply their knowledge to new problems and situation. This changing emphasis in a spiral curriculum can allow for a range of aspects of progression that are critical for Computer Science including: increasing difficulty of problems; enabling students to tackle more of the problem-solving process as they progress; consideration of the move from pictorial/block-based programming environments to text-based.

Table 2. Unified aims across all levels in Computing Curriculum in Poland [20]

Aim
1. understanding and analysis of problems based on logical and abstract thinking, algorithmic thinking, and information representations
2. programming and problem solving by using computers and other digital devices – designing algorithms and programs, organizing, searching and sharing information, using computer applications
3. using computers, digital devices, and computer networks – principles of functioning of computers, digital devices, and computer networks, performing calculations and executing programs
4. developing social competences – communication and cooperation, in particular in virtual environments, project based learning, taking various roles in group projects
5. observing law and security principles and regulations – respecting privacy of personal information, intellectual property, data security, netiquette, and social norms, positive and negative impact of technology on culture, social life and security

6. Pedagogical and Assessment challenges

While our discussion was focused on curriculum issues and it is beyond the scope of this paper to consider pedagogical and assessment challenges in depth, it is important that curriculum challenges are seen in the broader context of education. Specifically the curriculum is frequently depicted as in a triangular push-pull relationship with pedagogy and assessment in that both of these act as forces constraining or promoting curriculum change. Assessment was mentioned by many participants in the discussions as a constraint, particularly for more creative aspects of Computer Science that are typically harder to assess by traditional methods.

Implementing a curriculum focused on Computer Science was agreed to be particularly challenging in countries where the subject is being reintroduced and where there are not enough specialist-trained and experienced teachers and hence there is a lack of pedagogical expertise and a major professional development challenge. It is beyond the scope of this paper to discuss in depth the professional development needed to tackle these pedagogical challenges. Furthermore we acknowledge that factors leading to effective teacher professional development are difficult to identify and define and professional development is largely context dependent [see 22 for a review]. But in summary, our recommendations were to focus on teachers developing pedagogical content knowledge [23] which is much less well understood for Computer Science compared with other subject areas. Furthermore, we agreed on the importance of using technology resources to enhance or transform learning where appropriate and the importance of understanding the relationship between knowledge of technology use and pedagogical content knowledge [24]. An ongoing debate around the curriculum for Computer Science is the extent to which this curriculum must encompass enabling students to make good use of technologies. Learning Computer Science is not dependent on using the latest technologies and indeed the value of “unplugged activities” [25] for learning various difficult concepts

in Computer Science is well recognised. Nevertheless good use of new technologies can transform learning and we believe that pedagogy for Computer Science should reflect appropriate use of technologies for achieving this transformation [24] and thus set an example for other curriculum areas.

The nature of Computer Science as both scientific and practical presents significant challenges for its implementation. The curriculum in Poland, for example, emphasises the importance of problem solving at all levels. In our discussions we achieved a consensus regarding the value of engaging students in tackling real-world problems in order to stimulate their intellectual curiosity and motivation [24]. However, we should not underestimate the challenges of implementing a curriculum that incorporates problem solving and computational thinking as key practical elements of Computer Science. Understanding of the importance of problem solving in the curriculum has had a long and difficult history [see for example Schoenfeld's reflections over many years 26]. Schoenfeld's work encompassed theory and practice of problem-solving generally but focused specifically on its implementation in Mathematics education, where problem-solving is recognised by many to be crucial. In spite of a strong focus on the importance of problem-solving, implementation in Mathematics curricula has been limited (ibid.) owing to: pedagogical challenges in teaching and managing open-ended problem-solving; difficulties assessing problem-solving skills especially in authentic contexts and identifying suitable problems for students to tackle.

7. Discussion and conclusions

Key questions that emerged during our debate [4] [5] were:

1. What is the range of skills and understanding that should be developed in Computer Science?
2. Are such skills and understanding necessary for everyone? Should it be and remain compulsory?
3. At what age should Computer Science education commence?
4. How many computing languages or frameworks should a student be exposed to in the span of schooling from K-12?
5. How varied should these languages be? Should a variety of paradigms be explored?
6. How closely should the curriculum match computers available to schools and students?
7. What consideration in curriculum design should be given to emerging technologies such as quantum networks and optical computing?
8. What pedagogical approaches are likely to be appropriate, and how do they vary with age and other factors?

We have made some progress in answering Questions 1, 2, 3 and 8 but others remain to be examined. One of the strengths of the discussions that we have had to date are that they were not restricted to those engaged in Computer Science education but involved others who are committed to understanding the importance of digital technologies for learning but are more focused on digital literacy and the use of technologies for learning in other subjects. This has enabled us to gain greater insight into how Computer Science as a subject may be perceived and how we need to

continue to make the case for Computer Science and consider its relationship to the rest of the curriculum. There remain, for example, many who regard Computer Science as a specialist subject that is accessible only to older or more capable students. Currently the curriculum in Israel exemplifies this approach. Furthermore, in the broader education community, the driver/mechanic analogy, a long-standing argument against Computer Science as a curriculum subject for all, is still often used. This argument suggests that cars are analogous to computers and that the majority of pupils need only to be able to use them rather than understand their working. In order to counter such arguments, we need to be aware of the rationales for Computer Science, as outlined in this paper, and to continue to research and develop pedagogical approaches and professional development that enables the promise of Computer Science as a curriculum subject to be realised.

References

1. Webb, M.E., Micheuz, P., Brinda, T., Overland, E., Malyn-Smith, J., Angeli, C., Kalas, I., Fluck, A., Syslo, M., Chtouki, Y.: From Curriculum Visions To Computer Science And Computational Thinking In The Curriculum In Practice. World Conference on Computers in Education (WCCE), Dublin, Ireland (2017)
2. Joint Informatics Europe & ACM Europe Working Group on Informatics Education: Informatics education : Europe cannot afford to miss the boat: Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education. (2013)
3. The Royal Society: Shut down or restart? The way forward for computing in UK schools. The Royal Society (2012)
4. Wilson, C., Sudol, L.A., Stephenson, C., Stehlik, M.: Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age. Association for Computing Machinery (ACM), Computer Science Teachers Association (CSTA) (2010)
5. Fluck, A., Webb, M.E., Cox, M., Angeli, C., Malyn-Smith, J., Voogt, J., Zagami, J.: Arguing for Computer Science in the School Curriculum. Education Technology and Society 19, 38-46 (2016)
6. Denning, P.J.: The Profession of IT: Computing is a Natural Science. Communications of the ACM 50, 13-18 (2007)
7. Computer Science Teachers Association (CSTA) "CSTA K-12 Computer Science Standards." https://www.csteachers.org/page/standards (2017).
8. Sahami, M., Roach, S., Cuadros-Vargas, E., LeBlanc, R.: ACM/IEEE-CS computer science curriculum 2013: reviewing the ironman report. Proceeding of the 44th ACM technical symposium on Computer science education, pp. 13-14. ACM, Denver, Colorado, USA (2013)
9. Webb, M.E., Fluck, A., Cox, M., Angeli-Valanides, C., Malyn-Smith, J., Voogt, J., Zagami, J.: Thematic Working Group 9: Curriculum - Advancing Understanding of the Roles of Computer Science/Informatics in the Curriculum. In: Lai, K.-W. (ed.) EDUsummIT 2015 Summary Report: Technology Advance Quality Learning for All, pp. 60-69, Bangkok, Thailand (2015)
10. Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kegl, B., Rousseau, D.: The Higgs boson machine learning challenge. Journal of Machine Learning Research:

Workshop on High-energy Physics and Machine Learning and conference proceedings 42, 19-55 (2015)
11. Salomon, G., Perkins, D.: Transfer of cognitive skills from programming: When and how? Journal of Educational Computing Research 3, 149-169 (1987)
12. Grover, S., Pea, R.: Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher 42, 38-43 (2013)
13. Lye, S.Y., Koh, J.H.L.: Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior 41, 51-61 (2014)
14. Webb, M.E., Davis, N., Bell, T., Katz, Y.J., Nicholas, R., Chambers, D.P., Sysło, M.M.: Computer Science in K-12 school curricula of the 2lst Century: Why, what and when? Education and Information Technologies (2015)
15. Winch, C.: Curriculum Design and Epistemic Ascent. Journal of Philosophy of Education 47, 128-146 (2013)
16. Young, M.: Overcoming the crisis in curriculum theory: a knowledgebased approach. Journal of Curriculum Studies 45, 101-118 (2013)
17. Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., Boucher Owens, B., Stephenson, C., Verno, A.: CSTA K–12 Computer Science Standards. ACM/CSTA (2011)
18. Duncan, C., Bell, T., Tanimoto, S.: Should your 8-year-old learn coding? In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WIPSCE 2015), pp. 60-69. New York, ACM, 2670774 (Year)
19. Manches, A., Plowman, L.: Computing education in children's early years: A call for debate. British Journal of Educational Technology 48, 191-201 (2017)
20. Sysło, M.M., Kwiatkowska, A.B.: Introducing a new Computer Science curriculum for all school levels in Poland. In: ISSEP 2015, pp. 141-154. LNCS 9378, Springer Verlag, (Year)
21. Bruner, J.S.: The process of education. Harvard University Press (1960)
22. Cordingley, P., Higgins, S., Greany, T., Buckler, N., Coles-Jordan, D., Crisp, B., Saunders, L., Coe, R.: Developing great teaching : lessons from the international reviews into effective professional development. Project Report, DU (2015)
23. Shulman, L.: Those who understand: Knowledge growth in teaching. Educational Researcher 15, 4-14 (1986)
24. Angeli, C., Voogt, J., Malyn-Smith, J., Webb, M.E., Fluck, A., Cox, M., Zagami, J.: A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. Education Technology and Society 19, 47-57 (2016)
25. Bell, T., Newton, H.: Unplugging Computer Science. In: Kadijevich, D.M., Angeli, C., Schulte, C. (eds.) Improving computer science education, pp. 66-81. Routledge, New York, London (2013)
26. Schoenfeld, A.H.: Reflections on problem solving theory and practice. The Mathematics Enthusiast 10, 9-34 (2013)