https://doi.org/10.22364/atee.2019.itre.03

INTERDISCIPLINARY COMPUTING FOR STE(A)M: A LOW FLOOR HIGH CEILING CURRICULUM

Francesco Maiorana
Kansas State University, University of Catania, Italy
Department of Pure and Applied Science
Università di Urbino Carlo Bo 1506

ABSTRACT

There is an international, 360° effort to sustain and support education involving citizens of every age, all educational systems (formal, not formal, and informal), all levels of education (from primary schools to higher education), all disciplines (from Math to Latin), and all stakeholders (from educational institutions to industries and businesses).
In the paper, after reviewing the state of the art in Computing (C), Computational Thinking (CT), Computer Science (CS) and Digital Literacy (DL), a curriculum suited for a first course in computing, rooted in international frameworks and curricula, will be discussed. The work will present a detailed discussion of the content of a computing curriculum, suited for education across Europe, and its interdisciplinary applications. The curriculum can be useful for pre-service teachers' preparation, teachers' Professional Development (PD) and high school students. It develops along three strands: C, CT, and CS; DL used as a tool to document and present the artifacts produced in the C, CT, and CS projects, and soft skills introduced by contributions from leading researchers and educators around the world. The assessment practices, learning path, pedagogical approaches, and technologies, will be presented in order to aid teachers in their pre-service studies, PD, and daily teaching practice.

Keywords: Computing; Computational Thinking; Interdisciplinary Computer Science, Teacher preparation; Subject Knowledge; Model curricula; Interactive ebook.

Computing, Computational Thinking, Computer Science, and Digital Literacy

There is an international, 360° effort to sustain and support education which involves: all citizens, starting from children at pre-school to grandparents; all education systems, from formal to informal and even non-formal; all levels of education, from primary schools to higher education and lifelong learning; all disciplines, from Math to Latin and Ancient Greek: all stakeholders, from the education system to industries and businesses.

Parallel to this effort, a worldwide movement is striving to introduce the study of computing (Luxton-Reilly et al., 2018) from the first day of school, alongside reading, writing and basic arithmetic, and sustain this study throughout the life-long learning journey. This strong effort has produced a revision of mandatory state level curricula such as the Computing Curricula in England (DFE, 2013), the Australian curriculum (ACARA, 2016), the New Zealand Technology curriculum (TKI, 2017), the Computer Science Teacher Association K-12 Framework (CSTA, 2016) and Standard (CSTA, 2017). In the USA, code.org (Code.org) has been one of the most important non-profit organizations pushing for the introduction of CS across all states.

In Europe, a key role in this process has been taken by the European Commission and the European Schoolnet in cooperation with leading educational organizations such as OECD (OECD, 2018) and ACM Europe (Caspersen, et al., 2018). In this scenario, a question naturally arises about what are the competencies and skills that 21st-century citizens have to develop in their life. Among these competencies and skills, Computational Thinking (CT) (Wing, 2016)continues to plays a key role (Bocconi, et al., 2016), despite the long debate (Tedre & Denning, 2016) going back to the 1940s (Denning, 2017). All disciplines could potentially benefit from CT in a vision advocating for a shift "from STEM to STE(A)M (where 'A' includes all other disciplines)" (Hazelkorn, et al., 2015) bringing into the educational loop all stakeholders, from educators to industries and Ministries of Education (European Schoolnet, 2016). In this, the Scientix project (Baldursson & Stone, 2015) has a leading role ensuring, among many other things, that "no teacher faces unaided the hard but most needed task of getting kids to know, like and dream about science". According to the various operational definitions of CT (Csizmadia, et al., Computing At School, 2015), (Computer Science Teachers Association, 2011), (International Society for Technology in Education, s.d.) it is possible to argue that

- CT can be interpreted as a transversal set of skills that can be used as a means to acquire and to develop broad competencies like the ones proposed in (Binkley, et al., 2012).
- "more tools in the mental toolbox seems like a worthy goal" (Denning, 2017).

In order to realize this world-wide effort it is necessary to leverage teachers, the heart of the education system, and by leveraging their pedagogical and professional experience, offer resources for filling content gaps that could be present when teachers have majored in a different field than computing. Pre-service (Blamire & Cassells, 2019), (Maiorana, et al., 2019) and in-service Professional Development (PD) (Morelli, et al., 2014), (Lucarelli, et al., 2017), (Maiorana, et al., 2017) represent another way to enhance teachers' confidence in teaching computing. Other great examples of supporting initiatives of this widespread movement are represented by informal and outreach actions offered by international movements and initiatives like CoderDojo (CoderDojo, 2013), Europe code week (Europe code week, 2014), and communities of practices like Scientix (Scientix, s.d.).

All this effort has been supported by a strong, international 50-year research effort documented in (Luxton-Reilly, et al., 2018), (Becker & Quille, 2019), (Medeiros, et al., 2018). In this process a tension in the school system is apparent: on one side the need to offer a quality and inclusive education accessible to all students (UNESCO, 2017), (Burgstahler & Cory, 2010), (Burgstahler S., 2013) and, on the other, the necessity to increase the level of abstraction and cognitive demand in order to prepare the students for the higher cognitive skills required by the job market (Manca, 2018), (Ferrari, 2013).

The necessity of this synthesis is confirmed in many educational frameworks such as JRC (Bocconi, et al., 2016), the assessment in teaching of 21st century schools project (ATC21S) (Griffin & Care, 2014), Advanced Placement Computer Science Principles (College Board, 2017), Computer Science Teachers Association (CSTA) (CSTS, 2016) (CSTA, 2017), Organisation for Economic Co-operation and Development (OECD) (OECD, 2018), United Nations Educational, Scientific and Cultural Organization (UNESCO) (UNESCO, s.d.) that highlight a rich set of skills that students have to nurture.

In this paper the author presents a curriculum suited for a first course on computing by highlighting the design principles, and the learning trajectories. The three principal strands of the curriculum, namely Computational Thinking, digital literacy, soft and social skills are then presented. An evaluation of the proposed curriculum, a discussion summarizing main lessons learned, and conclusions and further work considerations complete the work.

Design principles

The curriculum aims to offer content and learning materials for a first course in computing suitable for all teachers and their students which, with adequate motivation, can be supported in climbing the learning pyramid from mere knowledge to creativity.

The fundamental ideas inspiring the curriculum are:

1) A low floor entry point suitable for all students and a high ceiling supporting the curiosity of all learners
2) Inquiry-based approach
3) Emphasis on design supported by many design tools
4)`Different expressive registers
5) Block based languages supporting high cognitive skills

6) Many programming languages with a common interface
7) Many advanced topics
8) Multiple learning trajectories that can be personalized to the needs of each student
9) Interdisciplinary applications
10) Multiple delivery media, e.g. book, interactive ebook, online course, etc …

In order to reach the low floor, high ceiling goal we envisage a cycle in the design process involving unplugged activities (Bell, et al., 2015), design tools such as flowgorithm (Cook, 2015), visual block languages and puzzles with an increasing level of difficulty, supporting students in their problem solving process.

The choice of using visual block languages leverages on the necessity, which has arisen from the rapid technological growth and exponential growth of the amount of available information, to sharpen the high order cognitive skills sought after by today's labor market (Manca, 2018). Visual block languages allow learners to focus on problem solving and high-order cognitive skills, avoiding the necessity to acquire syntactical details required by textual languages. Those languages become necessary when other considerations, e.g. efficiency of execution, are of primary importance.

The learning material can be used for:

1) A first high school course on computing, e.g. for K9-K10 grade band (CASTA, 2016), (CSTA, 2017)
2) Pre-service teacher training without a major in computing
3) Teacher Professional Development (PD)
4) A first undergraduate course for students majoring in fields other than computing, e.g. the humanities.

The learning trajectories

Figure 1 depicts the main concepts in the curriculum and how they are linked.

The concept maps can be navigated along many routes leaving the teachers the possibility to adapt the content to the class and each individual student.

We envisage a learning trajectory with a focus on developing CT. This will be produced by guiding students in acquiring a broad set of skills, useful not just to future computing professionals (Denning, 2017). The curriculum has the following strands:

- Computational thinking
- Digital literacy
- Soft and social skills

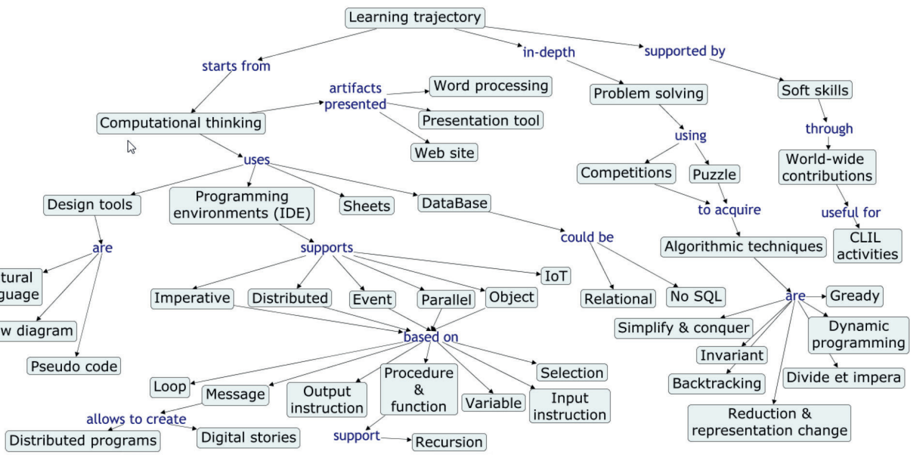

Figure 1. Curriculum concept map

The Computational Thinking strand

The CT strand uses a constructivist, student-centered approach grounded in cognitive theory/constructivism (Guzdial, 2018), and is based on the following activities:

- Reading, tracing, modifying and designing programs and algorithms expressed by means of:
  - Flow diagram (e.g. Flowgorithm)
  - Natural language
  - Pseudocode

Supported by activities requiring learners to translate from one representation to the others or to a visual block language

- Coding:
  - deluge of block languages, to experiment with core concepts in computing
  - translating the programs into a textual language
- Puzzle based learning:
  - algorithm design techniques: backtracking, divide and conquer, greedy, dynamic programming, invariant and so on.

The coding is supported by a deluge of block languages that, by sharing a common interface, allow teachers to leverage on their peculiar features to present and reason around core concepts in computing. Teachers can use the mutual support and reinforcement of the different programming and design tools, plugged and unplugged activities to offer a rich variety. For example, for parallelism unplugged activities such as the one proposed in (Bell, et al., 2015), (Tennessee Tech, s.d.) can support the plugged activities.

The author envisages in the curriculum the mutual support of plugged and unplugged activities, visual block-based and textual languages, multiple design tools to provide teachers and students with a richer set of design methodologies, tools, and expressive registers allowing each one to find the one most suited to her/his needs.

Table 1. A partial list of visual block languages used with a suggested progression and the key features of each language

	Block language	**Key features**
↓	Scratch (Resnick, et al., 2009)	Easy to use. Movement, Pen, Control, Procedures
	Scrible (Lane, Meyer, & Mullins, 2017)	Write on the stage. Create shapes
↓	NetsBlox (Broll & Ledeczi, 2017)	Message with data. Distributed programming
	Snap! (Harvey & Mönig, 2010)	Function. Recursion and functional programming. Parallel programming (e.g. map – reduce)
↓	Tunely (Trower & Gray, 2015)	Multimedia data manipulation in one dimension
	Pixly (Trower & Gray, 2015)	Multimedia data manipulation in two dimensions
↓	App Inventor (Patton, Tissenbaum, & Harunani, 2019)	Event programming. Mobile app development. NoSQL database. IoT
	Cellular (Lane, 2012),	Biological system simulation
	Blop (Federici, Gola, & Ilardi, 2014)	Block language for C/C++. Step towards textual languages
↓	BlockPy (Bart, Tibau, Tilevich, Shaffer, & Kafura, 2017)	Data manipulation. Automatic translation in Python
	Edgy (Cox, Bird, & Meyer, 2017)	Data structures. Bridge between unplugged and plugged
↓	GP (Monig, Ohshima, & Maloney, 2015)	Multimedia manipulation. Introduction to class without inheritance
	Parallel programming (Feng, Gardner, & Feng, 2017)	Blocks for parallel execution

The puzzle-based approach is a leitmotiv of the whole curriculum with puzzles proposed in all chapters and modules. Table 2 lists some of the major algorithm techniques and the puzzles used to introduce them. All the CT, and puzzle activities in the same module and across the whole curriculum, shows a progression from core to intermediate and advanced with a clear indication provided in the companion teacher's book. For students, an icon indication can guide them in choosing the preferred activities. The progression is supported by clear and sharp classification and progression provided in (Levitin & Levitin, 2011).

Table 2. Algorithmic techniques with some examples of puzzles proposed in the curriculum

Algorithmic techniques	Puzzle
Greedy	Pearson, bridge crossing and lamps; Huffman code
Decrease and conquer	A fake among eight coins, fake coin detection with a spring scale;
Divide and conquer	Tromino puzzle, 2n counters in a nxn board
Change of representation	Two jealous husbands, Stack of fake coins, Drawing a figure without lifting the pen; sequence of words
Dynamic programming	Shortest path counting; Knapsack problem; Common subsequence, Palindrome counting
Invariant	Break a chocolate bar; Colour of last marble; Knight movements; domino and tetromino tiling
Inference	Sequence of facts and conclusion;
Backtracking	Four and n queens; CriptoAlgorithms & CryptoArithmetica
Induction, proof of correctness	Knapsack problem, divide a rectangle in triangles

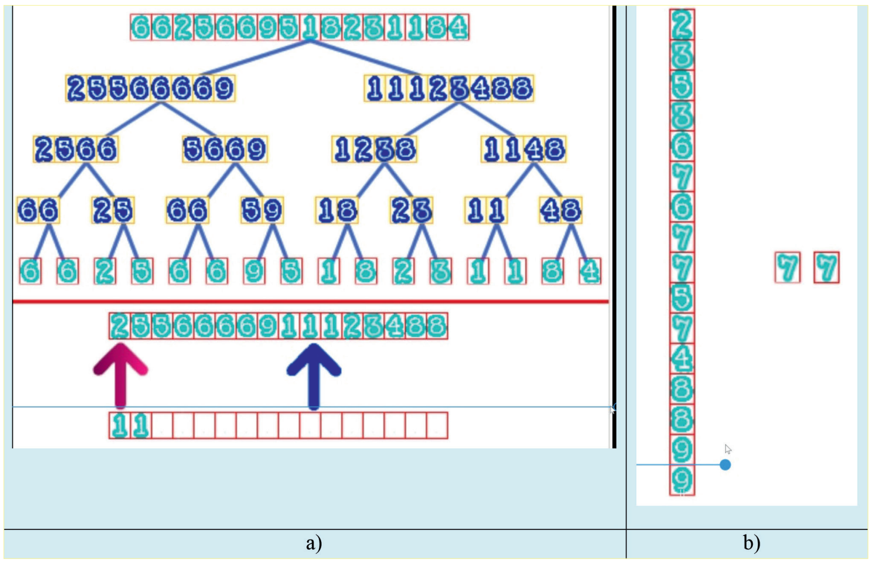

Figure 2. Sorting algorithm animations: a) Merge sort; b) Bubble sort

An inquiry-based approach is used to give the students a central role. Figure 2 shows a snapshot of two animations of merge-sort and bubble-sort. Students are requested to watch the animation and, before any educational intervention, are guided by a set of questions in discovering the algorithms behind this sorting processes. The set of questions goes into deeper detail in successive runs, e.g. midterm and final.

A similar approach is useful for algorithmic techniques such as backtracking. Figure 3 shows an example of a graph created with Edgy and its topological order.

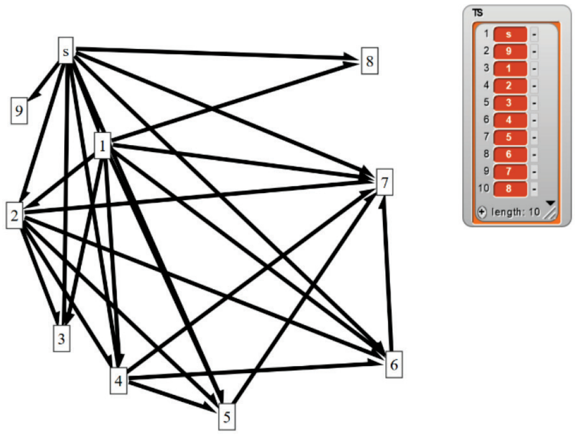

Figure 3. A graph and its topological order obtained with Edgy

The Digital Literacy strand

The digital literacy strand covers the following topics:

– Conduct bibliographic research.
– Being able to search, select, summarize, visualize and reference quality information. Particular emphasis is given to a rigorous process with clear and objective indications for every step: from selecting the search engine to selecting the best key phrases, for judging the source of information, verifying it and so on.
– Office automation. The major suites for office automation are presented, both proprietary, such as Microsoft, and open source such as LibreOffice and OpenOffice. Emphasis has been given to online and cloud-based tools as a way to hone collaboration and group work skills. To present the suites, an explorative approach is suggested, asking the students to find ways to accomplish tasks, either by exploration of the interface or by searching through the technical documentation. This is the best way to cope with

different interfaces changing over device, over software and over time. The explorative approach is always preferred and the correct solution, e.g. the sequence of steps to accomplish the task is given at the end of the activities, frequently only in the companion teacher's guide. Interface design principles are given by comparing the different interfaces available in the different devices (desktops, tablets and smartphones) and by analysing commonalities both intra applications inside the same family of software tools and inter office suites.

– Particular emphasis is given to searching, retrieving, analysing, visualizing and storing data. The importance of open and linked data is used as the key starting idea. Data are searched and retrieved and then analysed and visualized using Excel, Libre and Google sheets.
– Finally, storing data in databases (both relational and NOSQL) is considered. Activities for designing and querying a relational database and ways to visualize the data via an ad hoc designed interface are presented and suggested. The difference with a NOSQL database are explored and practical mobile applications are designed and developed by means of App Inventor and available NOSQL database components.

The soft and social skills strand

The importance of soft skills as well as social skills is recognized worldwide. For this reason, these topics are discussed through contributions from leading experts to open a window onto the world for students, giving them the possibility to compare the experiences from different countries and cultures. Among the topics covered, to cite just a few, it is possible to recall:

- Professional ethics
- Informal education
- Humanitarian Free and Open Source Software (HFOOS) – Free and Open Source Software (FOSS) (Hislop, Jackson, & Ellis, 2015), (Morelli, et al., 2009)
- Computer Science and its impact on society
- Inclusive education
- Mens sana in corpore sano (Healthy brain in healthy bod). Importance of sport
- Sustainable development
- Technologies and well-being

Contributions come from leading experts from: Australia; Canada; Europe: England, Ireland, Italy, Lithuania, Spain, Switzerland; New Zealand

and USA working in universities, international organizations, international institutions, enterprises.

This contribution can be used as Content Language Integrated Learning (CLIL) activities for students learning English as a second language.

Evaluation of results and discussion

The content derives from several experiences described and qualitatively and quantitatively evaluated in different studies (Giordano & Maiorana, 2014), (Giordano & Maiorana, 2015), (Maiorana, 2019). The positive effects of a first version of the curriculum have been evaluated by means of student progress on assessment evaluation and student survey (Giordano & Maiorana, 2015). Starting from the 2013 academic year, the curriculum was iteratively designed, developed, deployed, evaluated and improved. Each year the curriculum was field-tested in at least one class with an average of 25 students. Students, majoring in CS, where in either the first or second year (K9 or K10) of an Italian high school. The average female population was 15%. An average of 15% of students with disadvantaged socioeconomic status can be estimated. K9 students approached the course without mandatory prerequisites. For K10 students, a mandatory knowledge of basic problem-solving techniques and major programming constructs in an imperative language including procedures and functions were required. On average in each class, there were two students with learning disabilities (dyslexia or dysgraphia) and one student with special education needs. Curriculum effectiveness was qualitatively evaluated through student surveys and pre-test post-test assessment. When possible, comparisons with other classes in the same school taught by different professors were performed. The main conclusion that can be drawn from the evaluation process is that overall 14/16 years old students at the beginning of the course tend to underestimate blocks languages, considering them too simple, useful for younger people, not teenagers. As the progression of the topic becomes tougher and challenges the students, their appreciation of block languages increases since these languages allow the students to easily reason on the problems, construct artifacts and test them without worrying about too many details (Giordano & Maiorana, 2014).

Teacher feedback was obtained from five anonymous teacher reviews regarding the curriculum. The reviewers were located in Italy and the reviews were collected from mid 2017 to mid 2018. Other feedback was obtained from direct observations, informal unstructured teacher interviews inside a pre-service and professional teacher development course run in 2015. The teacher development course was attended by 40 teachers. Thanks to a Google CS4HS grant, the project run a teacher workshop where by means of surveys, and meeting with teachers the author obtained feedback about learning resources, teachers' needs, and expectations, and features desired for a curriculum. Analysing the teachers' feedback, it is possible to summarize the following key ideas:

1) On first impression, the quality of the proposed material and the diversity of the materials seem to disorientate some of them. For this purpose, indications of different progressions and a teacher guide offer a way to get acquainted with the curriculum. This guide can be used just as an ice-breaker; the experience and teachers' knowledge of their students will allow them to navigate the curriculum and find the best activity suited for the next steps in the zone of "Proximal development" for each individual student.
2) The ample diversity of communication channels and expressive registers, tools and technologies coupled with clearly stated progression and levels of difficulties allows for an inclusive and equitable approach. This approach is strengthened by an attention to learners with special abilities (UNESCO, 2017) in content delivery (edX, 2019).
3) The teaching approach sustained by inquiry-based pedagogies (Hazelkorn E., et al., 2015), Peer Instruction (Porter, et al., 2016), (Peer Instruction, 2019) and Process-Oriented Guided Inquiry learning (Education ambivalence, 2010), (Computer Science POGIL, s.d.) has the advantage of giving students an active role. By flipping the classroom (Bishop, Verleger, & others, 2013), (Karabulut-Ilgu, Jaramillo Cherrez, & Jahren, 2018) teacher-led and peer-led classroom time can be focused on problem-solving activities. Solving puzzles, engaging in projects (Blumenfeld, et al., 1991) and realizing artifacts to solve real world problems (Wolber, 2011), alone, in pairs and in groups allows learners to hone their collaboration and communication skills (Griffin & Care, 2014).
4) The interdisciplinary approach seems to be a promising way to expose students to computing, especially in school streams (e.g. classical studies) where computing is not a mandatory topic. In this case, where there is a lack of teachers with a specific certification in computing, approaching computing with applications in the teachers' and student's comfort zones represents a low floor entry point.
5) Use of formative assessment (Giordano D., et al., 2015), (Oates, Coe, Peyton Jones, Scratcherd, & Woodhead, 2016) supported by the above-mentioned pedagogies greatly supports students' activities and teachers' instructional process.

Undergraduate students with a major in the Humanities (Maiorana F., Computational Thinking and Humanities, 2018), most of them exposed for the first time to computing, reported, after overcoming foreseeable difficulties, joy and fulfillment in developing real work applications related to their subject of study and future profession and appreciated the design methodologies, the block language (Patton, Tissenbaum, & Harunani, 2019) and the possibilities to create mobile apps and sites showcasing their project portfolio.

Conclusion and further work

This work has presented the content, assessment, pedagogies, technologies and equity of a curriculum suited for a first course in computing, e.g. K9-K10 students, pre and in-service teachers, and undergraduate students. The curriculum is enriched by video, animation, assessment questions, and a companion website. The curriculum has been evaluated and improved during a multiyear and multidisciplinary teaching experience in high schools, undergraduate courses and informal education. A synthesis of the feedback received from students, teachers and reviewer and main lessons learned has been reported.

As a further study, the author plans to fine-tune the curriculum evaluation and improve it by leveraging different inputs, e.g. an international teacher surveys (Falkner, 2019) publish it and fully deploy and publish the companion web site (Maiorana F., Compucogito, 2019). The curriculum will be enlarged by designing, developing, deploying and evaluating learning resources suitable for a second and successive computing course. These courses will leverage on multiple design tools and on the use of visual block languages, as design and scaffolding tools. These tools will be coupled with textual language to develop interdisciplinary projects and real-world applications of interest for domains different from computing.

References

ACARA (2016) Australian Curriculum, Assessment and Reporting Authority. (2016, December 16). *F-10 Curriculum: Technologies*. Retrieved from https://www.australiancurriculum.edu.au/umbraco/Surface/Download/Pdf?subject=Digital%20 Technologies&type=F10.

Baldursson, R., & Stone, M. (2015). Scientix 2 results: How Scientix adds value to STEM education. *European Schoolnet, Brussels: http://files. eun. org/scientix/scientixworks/publications/Scientix_Results_Nov2015_Publication_ ONLINE. pdf*.

Bart, A., Tibau, J., Tilevich, E., Shaffer, C., & Kafura, D. (2017). Blockpy: An open access data-science environment for introductory programmers. *Computer, 50*(5), 18–26.

Becker, B., & Quille, K. (2019). 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, (p. 338–344).

Bell, T., Witten, I., & Fellows, M. (2015). CS Unplugged: An enrichment and extension programme for primary-aged students.

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). Defining twenty-first century skills. In M. Binkley, O. Erstad, J. Herman, S. Raizen, M. Ripley, M. Miller-Ricci, & M. Rumble, *Assessment and teaching of 21st century skills* (p. 17–66). Springer.

Bishop, J., Verleger, M., & others. (2013). The flipped classroom: A survey of the research. *ASEE national conference proceedings, Atlanta, GA*, *30*, p. 1–18.

Blamire, R., & Cassells, D. (2019). *INNOVATION IN INITIAL TEACHER EDUCATION. Emerging trends, issues and recommendations*. http://itelab.eun.org/documents/452109/470959/ITELab_deliverable_v3.pdf/1edcaca7-3a82-4e19-b5bb-f5988406744d.

Blumenfeld, P., Soloway, E., Marx, R., Krajcik, J., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational psychologist, 26*(3–4), 369–398.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., & others. (2016). *Developing computational thinking in compulsory education-Implications for policy and practice.* Joint Research Centre (Seville site).

Broll, B., & Ledeczi, A. (2017). Distributed Programming with NetsBlox is a Snap! *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education,* (p. 640).

Burgstahler, S. (2013). Universal design in higher education: Promising practices. *Seattle: U of Washington.*

Burgstahler, S., & Cory, R. (2010). *Universal design in higher education: From principles to practice.* Harvard Education Press.

Care, E., Griffin, P., & McGaw, B. (2012). *Assessment and teaching of 21st century skills.* Springer.

Caspersen, M., Gal-Ezer, J., McGettrick, A., & Nardelli, E. (2018). Informatics for All The strategy.

Chaiklin, S. (2003). The zone of proximal development in Vygotsky's analysis of learning and instruction. Vygotsky's educational theory in cultural context*, 1*, 39–64.

Code.org. (s.d.). https://code.org/.

CoderDojo. (2013). https://coderdojo.com/.

College Board. (2017). *AP Computer Science principles. Including the curriculum framework.*

Computer Science POGIL. (s.d.). http://cspogil.org/Home.

Computer Science Teachers Association. (2011). *Operational Definition of Computational Thinking*. http://www.csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf.

Cook, D. (2015). Flowgorithm: Principles for Teaching Introductory Programming Using Flowcharts. *Proc. American Society of Engineering Education Pacific Southwest Conf.(ASEE/PSW)*, (p. 158–167).

Cox, R., Bird, S., & Meyer, B. (2017). Teaching computer science in the victorian certificate of education: A pilot study. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, (p. 135–140).

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, H., Ng, T., Selby, C., & Woollard, J. (2015). *Computing At School.* Computational Thinking: A guide for Teachers: https://community.computingatschool.org.uk/resources/2324/single.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking – A guide for teachers.

CSTA. (2017). *K-12 Computer Science Standards.* Retrived from https://www.csteachers.org/page/standards.

CSTA (2016) Computer Science Teachers Association K–12 Computer Science Framework. (2016).Retrived from http://www.k12cs.org.

Denning, P. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39.

DFE (2013) U.K. Department of Education.. *National curriculum in England: computing programmes of study*. Retrieved from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.

Education ambivalence. (2010, 6 2). *Nature, 465*, 525.

edX. (2019). *Accessibility Best Practices Guidance for Content Providers*. https://edx.readthedocs.io/projects/edx-partner-course-staff/en/latest/accessibility/index.html.

Europe code week. (2014). https://codeweek.eu/.

European Schoolnet. (2016). ICT in STEM Education – Inpacts and Challenges: Setting the scene. A STEM Alliance Litterature Review. Brussels.

Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M., Quille, K. An International Benchmark Study of K-12 Computer Science Education in Schools. Working Group, Extended Abstract 24th Annual Conference on Innovation and Technology in Computer Science Education, Aberdeen, UK, July 2019, in press.

Federici, S., Gola, E., & Ilardi, E. (2014). Build Your Own Block Programming Language. *Scratch@ MIT 2014 Conference*, (p. 26).

Feng, A., Gardner, M., & Feng, W.-c. (2017). Parallel programming with pictures is a Snap! *Journal of Parallel and Distributed Computing, 105*, 150–162.

Ferrari, A. (2013). DIGCOMP: A framework for developing and understanding digital competence in Europe. Publications Office of the European Union Luxembourg.

for Economic Co-operation, O., & (OECD), D. (2018). The future of education and skills: Education 2030. Directorate for Education and Skills, OECD Paris, France.

Giordano, D., & Maiorana, F. (2014). Use of cutting edge educational tools for an initial programming course. *Global Engineering Education Conference (EDUCON)*, (p. 556–563).

Giordano, D., & Maiorana, F. (2015). Teaching Algorithms: Visual Language vs Flowchart vs Textual Language. *Global Engineering Education Conference (EDUCON)*, 499–504.

Giordano, D., Maiorana, F., Csizmadia, A., Marsden, S., Riedesel, C., & Mishra, S. (2015). New horizons in the assessment of computer science at school and beyond: Leveraging on the ViVA platform. *ITiCSE-WGP 2015 – Proceedings of the 2015 ITiCSE Conference on Working Group Reports.*

Griffin, P., & Care, E. (2014). *Assessment and teaching of 21st century skills: Methods and approach.* Springer.

Guzdial, M. (2018). *Constructivism vs. Constructivism vs. Constructionism.* https://computinged.wordpress.com/2018/03/19/constructivism-vs-constructivism-vs-constructionism/.

Harvey, B., & Mönig, J. (2010). Bringing "no ceiling" to scratch: Can one language serve kids and computer scientists. *Proc. Constructionism*, 1–10.

Hazelkorn, E., Ryan, C., Beernaert, Y., Constantinou, C., Deca, L., Grangeat, M., … others. (2015). Science Education for Responsible Citizenship: Report to the European Commission of the Expert Group on Science Education. Luxembourg: Publications Office of the European Union.

Hazelkorn, E., Ryan, C., Beernaert, Y., Constantinou, C., Deca, L., Grangeat, M., … Welzel-Breuer, M. (2015). Science education for responsible citizenship. *Report to the European Commission of the expert group on science education.*

Hislop, G., Jackson, S., & Ellis, H. (2015). FOSS Artifacts for Evaluating Students on Team Projects. *Proceedings of the 16th Annual Conference on Information Technology Education*, (p. 57).

International Society for Technology in Education. (s.d.). *ISTE Standards for Students*. http://www.iste.org/standards/standards/for-students-2016.

Karabulut-Ilgu, A., Jaramillo Cherrez, N., & Jahren, C. (2018). A systematic review of research on the flipped learning method in engineering education. *British Journal of Educational Technology, 49*(3), 398–411.

Kožuh, I., Krajnc, R., Hadjileontiadis, L., & Debevc, M. (2018). Assessment of problem solving ability in novice programmers. *PloS one, 13*(9), e0201919.

Lane, A., Meyer, B., & Mullins, J. (2017). *Generative Art with Scribble*. https://www.alexandriarepository.org/syllabus/scribble/.

Lane, A., Meyer, B., Mullins, J., & Albrecht, D. (2012). Simulation with Cellular. A Project Based Introduction to Programming: http://www.flipt.org/.

Levitin, A., & Levitin, M. (2011). *Algorithmic puzzles.* OUP USA.

Lucarelli, C., Rosato, J., & Beckworth, C. (2017). "Virtual Visits": Promising practices and lessons learned in the use of video teaching samples for online professional development. *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, (p. 978–983).

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B., Giannakos, M., Kumar, A., … Szabo, C. (2018). A Review of Introductory Programming Research 2003–2017. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (p. 342–343). New York, NY, USA: ACM.

Maiorana, F. (2019). "Low floor high ceiling" Computer Science. *Didamatica*, (p. 219–228).

Maiorana, F., Richards, G., Lucarelli, C., Miles, B., & Ericson, B. (2019). Interdisciplinary Computer Science pre-service Teacher Preparation. *24th Annual Conference on Innovation and Technology in Computer Science Education.* Aberdeen, UK: ACM.

Maiorana, F. (2019 b) CompuCogito. compucogito.net.

Maiorana, F. (2018). Computational Thinking and Humanities, pp. 87–96. Didamatica 2018.

Maiorana, F., Berry, M., Nelson, M., Lucarelli, C., Phillips, M., Mishra, S., & Benassi, A. (2017). International perspectives on CS teacher formation and professional development. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE, Part F1286.*

Manca, F. (2018). *Skills for Job.* OECD.

Medeiros, R., …, G.-I., & 2018, u. (s.d.). A systematic literature review on teaching and learning introductory programming in higher education. *ieeexplore.ieee.org.*

Monig, J., Ohshima, Y., & Maloney, J. (2015). Blocks at your fingertips: Blurring the line between blocks and text in GP. *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, (p. 51–53).

Morelli, R., Tucker, A., Danner, N., De Lanerolle, T., Ellis, H., Izmirli, Ö., ... Parker, G. (2009). Revitalizing computing education through free and open source software for humanity. *Commun. ACM, 52*(8), 67–75.

Morelli, R., Wolber, D., Rosato, J., …, C.-P., & 2014, u. (s.d.). Mobile computer science principles: a professional development sampler for teachers. *dl.acm.org.*

Oates, T., Coe, R., Peyton Jones, S., Scratcherd, T., & Woodhead, S. (2016). *Quantum: tests worth teaching to.*

OECD. (2018). *The future of education and skills. Education 2030*. http://www.oecd.org/education/2030/oecd-education-2030-position-paper.pdf.

Patton, E., Tissenbaum, M., & Harunani, F. (2019). MIT App Inventor: Objectives, Design, and Development. In E. Patton, M. Tissenbaum, & F. Harunani, *Computational Thinking Education* (p. 31–49). Springer.

Peer Instruction. (2019). https://www.peerinstruction4cs.org/.

Porter, L., Bouvier, D., Cutts, Q., Grissom, S., Lee, C., McCartney, R., ... Simon, B. (2016). A multi-institutional study of peer instruction in introductory computing. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, (p. 358–363).

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... others. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60–67.

Scientix. (s.d.). http://www.scientix.eu/.

Tedre, M., & Denning, P. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, (p. 120–129).

Tennessee Tech. (s.d.). *iPDC modules*. https://www.csc.tntech.edu/pdcincs/index.php/ipdc-modules/.

TKI (2017) Ministry of Education. (2017). *Technology in the New Zealand Curriculum*. Retrived from http://technology.tki.org.nz/Technology-in-the-NZC.

Trower, J., & Gray, J. (2015). Blockly language creation and applications: Visual programming for media computation and bluetooth robotics control. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, (p. 5).

Tsan, J., Rodr\'\iguez, F., Boyer, K., & Lynch, C. (2017). Let's work together: Improving block-based environments by supporting synchronous collaboration. *2017 IEEE Blocks and Beyond Workshop (B&B)*, (p. 53–56).

UNESCO. (2017). *Moving forward the 2030 agenda for sustainable development.*

UNESCO. (s.d.). *Competency Framework*. https://unesdoc.unesco.org/ark:/48223/pf0000245056.

UNESCO, A. (2017). A guide for ensuring inclusion and equity in education. UNESCO Paris.

Wing, J. (2016). Computational thinking, 10 years later. *Microsoft Research Blog, March, 23*, 2016.

Wolber, D. (2011). App inventor and real-world motivation. *SIGCSE, 11*, p. 601–606.