Proposal 3

TACCLE 3, O5: An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers

KA2 project "TACCLE 3 – Coding" (2015-1-BE02-KA201-012307)

Authors: García-Peñalvo, F. J. (University of Salamanca, Spain), Reimann, D. (Karlsruhe Institute of Technology, Germany), Tuul, M. (Tallinn University, Estonia), Rees, A. (Pontydysgu, Welsh educational research Institute, UK) & Jormanainen, I. (School of Computing, University of Eastern Finland, Finland)

2016

ABSTRACT

Within TACCLE 3 – _Coding European Union Erasmus+ KA2 Programme project, a structured report listing and describing available literature has been done. This list is to be made availbale to teachers via the website in a user friendly format. Recomendations, instructions and directions regarding pedagogical aspects of teaching coding have been anlaysed and made available based on this literature. This document represents the TACCLE 3 O5 deliverable entitled “Literature Review”.

KEYWORDS

Computational Thinking; Coding; Literature Review; TACCLE 3

CITATION

This report may be cited as follows:

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). *An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers*. Belgium: TACCLE3 Consortium. doi:10.5281/zenodo.165123.

DISCLAIMER

Contents

Introduction

The rationale for this literature review is to take the main areas of debate surrounding the teaching of coding to primary aged children and examine the polemic and the different positions that writers and practitioners are taking on these issues. This is intended to be a literature review useful to practitioners rather than academics. In that vein, in addition to published peer referenced journal articles we have decided to make maximum use of blog posts and opinions on social media for our source material as we believe that the most informed debate on kids coding is not actually going on in academic journals!

In fact, in a meta-analysis of 27 peer-reviewed papers on the subject of teaching computational thinking to school aged children, it was found that only 'nine peer-reviewed intervention studies were based in K-12 settings', highlighting the gap in the research of developing computational thinking in school aged children. 'Even with these limited studies, most were conducted as after-school activities.' (Lye, S. Y. & Koh, J. H. L., 2014.) So even the academics know that the scope of their own work is limited.

Based on the preceding, the TACCLE 3 team members have compiled reviews of academic papers and have also trawled through blogs, news-articles and opinion pieces to find some answers to the questions teachers regularly ask us, along with a few extras that we found interesting. The information in this paper has been divided into 5 chapters. In the first chapter TACCLE 3 team members try to answer the following questions:

- Why are we teaching coding?
- Should we actually be teaching coding to young children at all?
- How should we be teaching it?
- How to best use tangible user interfaces?
- Are there gender issues to overcome?

In chapter 2 Francisco Jose García-Peñalvo provides a deeper introduction into computational thinking: what it is, what are the core concepts of it and how to introduce this approach into the classrooms. In chapter 3 Daniela Reimann introduces us 'smart textiles' as a creative environment for programming interactive objects. In chapter 4 Maire Tuul gives an overview of Makey Makey, a platform for improvising tangible user interfaces. In chapter 5 Ilkka Jormanainen takes a look into the world of robotics. All of the chapters have been written in a way that it is possible to understand the content of each chapter without reading the paper from cover to cover. You can just pick out the parts you find interesting. Enjoy reading!

References

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41,* 51-61.

1. Key debates around teaching coding to primary school children

1.1 Why are we teaching coding?

Are we creating a workforce for the future and trying to address the current skills shortages in the labour market or is it an academic subject geared to intellectual development or simply a necessary life skill?

It is very important be able to control technology in the digital world in which we live and the need to learn programming to not be at the mercy of decisions that those that are going to develop all that we will use in the future, because everything, or almost everything, that we will use or consume, is going to be digital and will be coded in some other form (Rushkoff, 2010).

Formal study of computational skills in primary and secondary schools has been recognized by many institutions and administrations. For example, England, beginning the academic year 2014-15, has formally included the study of computational thinking and computer programming as part of the curriculum of primary and secondary education, as described in the national curriculum England: Study of Computer Program (Department of Education, 2013).

The Code.org organization promotes the idea that all students should have the opportunity to learn programming. This initiative has the support of important public figures of Microsoft, Facebook and the world of technology in general.

- ‘Digital intelligence or “DQ” is the set of social, emotional and cognitive abilities that enable individuals to face the challenges and adapt to the demands of digital life. These abilities can broadly be broken down into eight interconnected areas:
- Digital identity: The ability to create and manage one’s online identity and reputation. This includes an awareness of one's online persona and management of the short-term and long-term impact of one's online presence.
- Digital use: The ability to use digital devices and media, including the mastery of control in order to achieve a healthy balance between life online and offline.
- Digital safety: The ability to manage risks online (e.g. cyberbullying, grooming, radicalization) as well as problematic content (e.g. violence and obscenity), and to avoid and limit these risks.
- Digital security: The ability to detect cyber threats (e.g. hacking, scams, malware), to understand best practices and to use suitable security tools for data protection.
- Digital emotional intelligence: The ability to be empathetic and build good relationships with others online.
- Digital communication: The ability to communicate and collaborate with others using digital technologies and media.

- Digital literacy: The ability to find, evaluate, utilize, share and create content as well as competency in computational thinking.
- Digital rights: The ability to understand and uphold personal and legal rights, including the rights to privacy, intellectual property, freedom of speech and protection from hate speech.' (World Economic Forum) (Park, 2016).

'Why is it so vital that we teach our children to code? We are already living in a world dominated by software. Your telephone calls go over software-controlled networks; your television is delivered over the internet; people don't buy maps anymore, they use the web; we all shop online. The next generation's world will be even more online and digital. Soon, your house will be controlled with software, some of your medical care will be delivered over the web and your car may even drive itself.' (Crow, 2014.)

Many people argue that the emphasis should not be on teaching coding at all, code is after all just another language. More importantly we should be teaching how to solve problems, how to use logic, and creativity, those transferable skills are more important than code on its own and can be applied not just to any programming language but to real life contexts. 'This is not primarily about equipping the next generation to work as software engineers, it is about promoting computational thinking' (Crow, 2014).

Coding is one of a range of tools which can be used to teach computational thinking. There are lots of offline, unplugged, hands-on activities which promote problem solving and logic without ever having to touch a computer.

'…computational thinking conducts a detailed analysis of problems. This helps not only in coding, but also in being able to analyze and thoroughly understand problems, identify patterns and extrapolate solutions.' (Sanz, 2015.)

Too many voices worldwide defend that computational thinking is a core element when coding is taught to children. The way of thinking, of learning to solve problems and creating algorithms and solutions both individuals and - above all - general to those problems. Software engineers have clear this concept allows them, for example, having a problem of some magnitude and knowing divide and resolving each of the parts, generalizing this solution. But the objective should not be coding without no more sense, programming should be an excuse for more. Programming is a tool that can encourage creativity to extraordinary levels. Programming should not be just writing code, programming allows us more throughout the computational thinking and creativity capabilities, promoting the sociability of the students working together to solve problems (Club de Jóvenes Programadores de la Universidad de Valladolid, Spain) (Espeso, 2015).

Learning to code and develop computational thinking are skills that make an essential contribution in the learning process of children, helping them to face many situations they will find in life, and enabling them to better collaboration between human and machine (Martín, 2016).

The area of knowledge of computational thinking is experiencing a significant expansion in the private and public education sectors both in the developed world and in the developing world. Students trained in computational thinking are significantly better prepared for the daily tasks and the professional work that awaits them in their immediate future (Basogain et al., 2016).

‘We believe in the ability of every child and young awakening and discovering his talent through creative and entrepreneurial activity at school.’ (Fundación Créate website)

Teaching coding from early ages is one of the ways to make easy the immersion of the countries in the digital economy. Many countries are betting for this initiatives, for example in China the Tarena Learning Center launched its first junior coding class in Beijing with about 50 kids; enrollment is now expected to reach as many as 4,000. ‘That Chinese kids, including some as young as six, are learning the basics of computer programming is by no means a sign that the country is on the technological cutting edge. In fact, the world's second-largest economy is lagging when it comes to such innovation.’ (NBC News) (Frayer, 2016).

During the course of this project we have already had a few long debates about what is meant by terms such as coding, programming, computing and computer science. The Computer Science Teachers Association and International Society of Technology in Education (2011) give this definition for computational thinking:

‘problem-solving process that includes formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combinations of steps and resources; and generalizing and transferring this problem solving process to a wide variety of problems’.

Taccle’s partner at the University of Salamanca have produced a detailed review of literature relating to computational thinking (see chapter 2).

References

Basogain, X., Olabe, M. A., C., O. J., Ramírez, R., & García, J. (2016). PC-01: Introduction to Computational Thinking. Educational Technology in Primary and Secondary Education. In F. J. García-Peñalvo & J. A. Mendes (Eds.), *XVIII Simposio Internacional de Informática Educativa, SIIE 2016* (pp. 191-195). Salamanca, España: Ediciones Universidad de Salamanca.

Computer Science Teachers Association and International Society of Technology in Education. (2011). *Operational definition of computational thinking for K-12 education.* [2016, October 1] http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

Crow, D. (2014). Why every child should learn to code. *The Guardian.* [2016, September 7] https://www.theguardian.com/technology/2014/feb/07/year-of-code-dan-crow-songkick

Department for Education. (2013). *National curriculum in England: computing programmes of study - key stages 1 and 2*. (DFE-00171-2013). UK: UK Government Retrieved from https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf.

Espeso, P. (2015, December 14, 2015). *Cómo iniciar a un niño en la programación desde cero.* [2016, October 2] http://www.xataka.com/otros/como-iniciar-a-un-nino-en-la-programacion-desde-cero

Frayer, J. M. (2016, September 21). *China Pushes Coding for Kids in Effort to Tackle Innovation Gap. NBC News*. [2016, October 2] http://www.nbcnews.com/news/china/china-pushes-coding-kids-effort-tackle-innovation-gap-n641966

Fundación Créate website [2016, October 19]. http://www.fundacioncreate.org/

Martín, J. (2016, May 20, 2016). Por qué debemos enseñar a los niños a programar. [2016, October 2] http://futurizable.com/debemos-ensenar-los-ninos-programar

Park, Y. (2016, 13 June 2016). 8 digital skills we must teach our children. [2016, September 7] https://www.weforum.org/agenda/2016/06/8-digital-skills-we-must-teach-our-children/

Rushkoff, D. (2010). *Program or Be Programmed: Ten Commands for a Digital Age*. USA: BookMobile.

Sanz, A. (2015, April 26). Why Teaching and Learning How to Code in Schools. EdTech Review. [2016, September 6] http://edtechreview.in/trends-insights/insights/1934-why-teaching-and-learning-how-to-code-in-schools

1.2 Too young to code?

There is ongoing popular debate about whether or not children should be exposed to computer screens let alone taught to program one. The recommended maximum screen time for children has traditionally been set at 2 hours per day but the American Academy of Pediatrics recently changed their guidelines to favour common sense whilst stressing that offline, unplugged playtime should be prioritised. Nevertheless, it is important to remember that a number of studies have shown links between children's excessive screen viewing and higher level of psychological difficulties. For example, Hamer, Stamatakis, & Mishra (2009) concluded in their research paper that 'higher levels of television and screen entertainment time and low physical activity levels interact to increase psychological distress in young children', whilst Page, Cooper, Griew, & Jago (2010) have found, that children who spend much time in front of a computer or television screen are more likely to suffer psychological difficulties, regardless of how physically active they are. Hinkley, Verbestel, Ahrens et al. (2014) concluded after analysing the effect of early childhood use of electronics and long-term well-being in over 3500 children between the ages of 2 and 6, that higher use of electronic media in early childhood is associated with poorer well-being outcomes. These results suggest that in teaching coding to young children we have to consider carefully, how to do it so, that we do not inadvertently affect the child's physical and emotional development. Fortunately, there are lots of ways to teach coding without using a computer or using it minimally. For example, robots such as Dash and Dot, Bee-Bot and CodeaPillar teach children some of the same principles of computer science but involve the active, floor-space learning that is more suited to early-childhood classrooms (Jacobson, 2016).

But should we actually be teaching coding to young children? There is a general consensus that learning languages is better done when young and since learning coding is just like learning another language, advocates of coding suggest that children can never be too young to code. 'Coding brings young children rich opportunities for language development and the 'notion of learning from mistakes', says Chip Donohue, the dean of distance learning and continuing education at the Erikson Institute in Chicago, a graduate school in child development. 'We actually don't do enough of that with young kids.' The sequencing and patterns involved in programming reinforce skills that have always been taught in the early years, but now also create 'habits of mind that are essential for the 21st century', adds Donohue. (Jacobson, 2016.) Resnick (2013) sees coding as an extension of writing. He explains: 'The ability to code allows you to "write" new types of things – interactive stories, games, animations, and simulations. And, as with traditional writing, there are powerful reasons for everyone to learn to code'. According to Resnick (2013), while learning to code in addition to learning mathematical and computational ideas, people are also learning strategies for solving problems, designing projects, and communicating ideas. He adds that these skills are useful for everyone regardless of age, background, interests or occupation.

There is also the question if coding platforms are appropriate for young children. However, developers are already producing a growing number of apps and platforms designed specifically for younger children. Back in 2012 MIT was working on a programming environment for toddlers, they described programming in terms of a 'new literacy'. Mims (2012) asks: 'If children are exposed to the alphabet from the time they can sit up, why wouldn't we introduce programming as early as possible?' It is not easy to answer, because if you introduce programming to young children taking into account children's age, their developmental needs and capabilities, it is hard to find arguments not to do it.

References

Hamer, M., Stamatakis, E., & Mishra, G. (2009). Psychological Distress, Television Viewing, and Physical Activity in Children Aged 4 to 12 Years. *Pediatrics, 123* (5), 1263-1268.

Hinkley, T., Verbestel, V., Ahrens, W., Lissner, L., Molnár, D., Moreno, L. A., Pigeot, I., Pohlabeln, H., Reisch, L. A., Russo, P., Veidebaum, T., Tornaritis, M., Williams, G., De Henauw, S., & De Bourdeaudhuij, I. (2014). Early childhood electronic media use as a predictor of poorer well-being: a prospective cohort study. *JAMA Pediatrics, 168* (5), 485-492.

Jacobson, L. (2016, April 5). Never Too Young To Code. *School Library Journal* [2016, October 3] http://www.slj.com/2016/04/technology/never-too-young-to-code/#_

Mims, C. (2012, February 26). How Young Is Too Young to Learn to Code? *MIT Technology Review.* [2016, October 3] https://www.technologyreview.com/s/427064/how-young-is-too-young-to-learn-to-code/

Page, A. S., Cooper, A. R., Griew, P., & Jago, R. (2010). Children's screen viewing is related to psychological difficulties irrespective of physical activity. *Pediatrics, 126* (5), e1011-e1017.

Resnick, M. (2013). *Learn to Code, Code to Learn*. [2016, October 4] https://cedsurge.herokuapp.com/news/2013-05-08-learn-to-code-code-to-learn

1.3 Coding Pedagogy

Is there a specific "coding pedagogy"? We argue that project based, self-initiated learning in collaborative groups, as known from constructivist and constructionist pedagogy related to arts, crafts and design-based contexts involving the imagination of the learner is key to teach coding. The latter can be realized using the different media technologies identified in the TACCLE3 project. Those allow for constructing, designing and programming real objects, such as wearable music, banana pianos, robots and other interactive objects which include behavior, modeled and controlled by the learners themselves.

From a teacher's perspective, where do you start? What are progression options? Computational thinking and a load of unplugged stuff then moving onto coding or do you just launch them straight into Scratch!

From a pedagogy point of view, it's not enough for teachers to just have lesson content and subject knowledge, there also needs to be structure and progression. Where do you start? Do you launch straight into programming a Raspberry Pi or start with offline activities like card sorts?

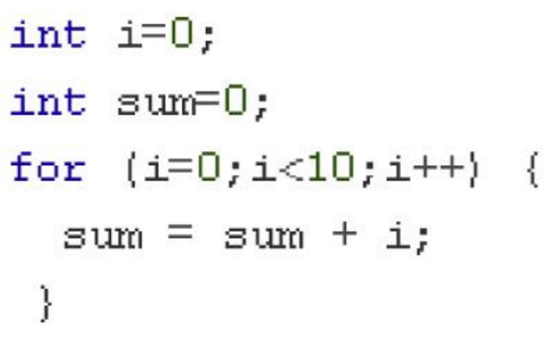
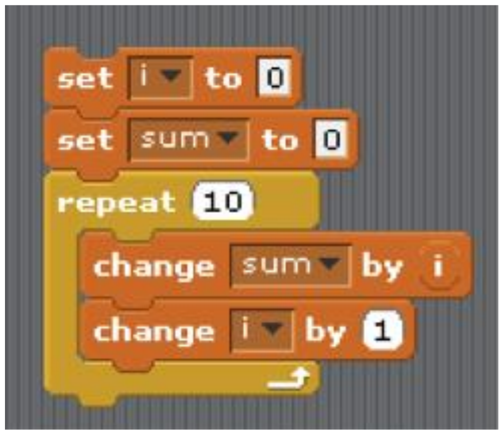
"Teachers also need to learn appropriate pedagogies for delivering a new subject, particularly in those aspects of computer science that relate to algorithms, programming and the development of computational thinking skills."

If one agrees, that the focus for teaching should be on computational thinking, then teachers can stop worrying about which language to use first and focus on how best to teach the skills required.

"What's the best way to teach little ones? Experts suggest that educators avoid the old computer lab model in which students spend a set amount of time each day or week practicing basic coding skills. Preschoolers "don't need to know how it works," says Donohue. "Teachers who are doing this well are turning this into a very active, very social opportunity for kids to use language and learn the words they will need." (Jacobson, 2016.)

Teaching code and computational thinking is not a new thing, programming was taught as far back as 1960 but as computers evolved, the skills taught changed from programming to using programmes. Rather than go back to the days of using Logo in the classroom, modern languages for school children are based on blocks of code which can be dragged and dropped to create a programme. This method of building up a series of instructions works equally well offline with simple card sorts or more sophisticated programming bricks.

'It is better to use visual programming languages rather than traditional [text-based] programming languages to facilitate the three dimensions of computational thinking in [school age] contexts because unnecessary syntax is reduced (e.g., the use of semicolon and curly brackets) and the commands are closer to spoken English. Students usually need only to drag and snap the command blocks (see Figure 1). With these features, such programming tools help reduce the cognitive load on the students and ''allow students to focus on the logic and structures involved in programming rather than worrying about the mechanics of writing programs'' (Kelleher & Pausch, 2005, p. 131). As such, these features of visual programming languages can potentially allow students to acquire the computational concepts more easily without the need to learn complex programming syntax.' (Lye & Koh, 2014.) In TACCLE 3 we argue that iconic programming tools and environments can serve as a vehicle to drive interest of particular target groups which are less confident with using language and complex text based programming languages.

Traditional language (Java)	Visual programming language (Scratch)
	

```
int i=0;
int sum=0;
for (i=0;i<10;i++) {
  sum = sum + i;
 }
```

```
set i to 0
set sum to 0
repeat 10
  change sum by i
  change i by 1
```

Figure 1. Programming languages (Lye & Koh, 2014)

A survey of 357 teachers in England who are members of the Computing at School network asked teachers about the best way to teach coding and computational thinking. The results show that teachers emphasised unplugged, hands-on, contextualised activities and the importance of lots of practice. Those teachers who were confident in their ability to teach coding were "combining strategies around code exercises, using discussion (collaboration and computational thinking), and working away from the computer.", "utilising a variety of teaching strategies to support learning, rather than relying on one particular strategy." and they also reported "the need for students to reflect on what they are learning in computing and be able to articulate it." (Sentence & Csizmadia, 2015.)

References

Jacobson, L. (2016, April 5). Never Too Young To Code. *School Library Journal* [2016, October 3] http://www.slj.com/2016/04/technology/never-too-young-to-code/#_

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 83–137.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61.

Sentence, S. & Csizmadia, A. (2015). Teachers' perspectives on successful strategies for teaching Computing in school. In A. Brodnik & C. Lewin (Eds.). *IFIP TC3 Working Conference "ANew Culture of Learning: Computing and next Generations". Preliminary Proceedings*. Lithuania, Vilnius University, 201-210. [2016, October 1] http://docplayer.net/16256814-Ifip-tc3-working-conference-a-new-culture-of-learning-computing-and-next-generations.html

1.4 Do children need keyboards?

Should we keep kids away from keyboards until they have explored interfaces more in keeping with their physical development? The issue of keeping up to the qwerty-keyboard which was developed on mechanical typewriters for specific reasons linked to the mechanical typing process, and kept in digital technology though there was no technological need anymore (apart from the fact, that billions of people were trained how to type systematically on a qwerty keyboard) was stressed by Papert in the 'Mindstorms' (1982). As children learn through touching and grasping, we argue that tangible media introduced by Ishii in the framework of the Tangible Media research group at MIT [Patten, Griffith, & Ishii, 2000], offer the opportunity to learn about technology by touching and moving physical objects for interacting with a computer. We came up with tools, such as sensor technologies, micro-computers, and iconic programming we consider of high use in creative computer education processes. However, there is debate on when to introduce children to a tablet interface, which includes a rather poor haptic experience in terms of materiality (touching a smooth flat glass-like interface, linked to the digital objects behind the screen (Spitzer, 2012). Peez (2014) argues, to open up aspects of knowledge through typing with your index finger, to discard them again also by lateral wiping, zooming in to take a closer look, connects to the earliest experiences of self-efficacy 'and the elementary forms of linking the brain with the motor system.

The following list gives some good reasons for using Tangible User Interfaces (TUI) that is something which controls a computer or produces a response and can be held, manipulated or played with. For example, a MaKey MaKey, wearable arduino technology, BeeBots, and programming bricks. You can find ideas for using these in the classroom via the TACCLE 3 portal by clicking on Ideas and Resources.

- TUI requires little time to learn how to use the interface. TUI is a natural interface which requires little cognitive effort to learn, therefore children can concentrate more on the task, rather than how to use the computer or software.
- TUI offers users an alternative way of interaction and control of the computing environment. TUI can offer a variety of interactions, it allows user to solve problems with concrete physical objects and physical action when they fail using more abstract representations and complex syntax, therefore TUI can empower children with the control of the computing environment; they will feel and own the environment and will be actively engaged and not lose their interest easily.
- TUI supports Trial-And-Error activity TUI gives continuous presentation of the object of interest. It uses rapid incremental and reversible actions whose impact on the object of interest is immediately visible.
- TUI supports more than one user The advantage of using a TUI is that it is no longer restricted to a single user, children can sit down and collaborate with their friends face to face in an entirely natural way. It can provide children with a social experience. (Xu, 2005.)

But research in other areas suggests that TUIs can be confusing, in Mathematics some studies show that children struggle to transfer what they have learned to other situations. ‘Some other perspectives on manipulating physical material suggest that TUI can also lead to no or negative learning outcomes.” (Schneider, Jermann, Zufferey, & Dillenbourg, 2011.) Which highlights the need to not use the TUI for the sake of it but rather to use it to illustrate a concept or to perform a task.

In the framework of Taccle3, a literature reviews particularly related to MaKey MaKey (see chapter 4) as well as for wearable technologies created with Arduino (see chapter 3) were developed.

References

Papert, S. (1982). *Mindstorms Gedankenblitze*. Reinbek bei Hamburg.

Patten, J., Griffith, L., & Ishii, H. (2000). A tangible Interface for controlling robotic toys. *Proceedings of CHI’00*, April 1-6, 2000, The Hague, NL.

Peez, G. (2014). Mit Fingerspitzengefühl zu Erfahrung und Wissen. Kasuistische Grundlagenforschung zur sensomotorischen Bedienung von Multi-Touchscreens. In. merz medien + erziehung. *Zeitschrift für Medienpädagogik*, *2*, 67-73. [2016, September 23] http://www.georgpeez.de/texte/merz14.htm

Schneider, B., Jermann, P., Zufferey, G., & Dillenbourg, P. (2011). Benefits of a Tangible Interface for Collaborative Learning and Interaction. *IEEE Transactions on Learning Technologies*, *4*(3), 222-232.

Spitzer, M. (2012). *Digitale Demenz*. München: Droemer HC.

Xu, D. (2005). *Tangible User Interface for Children – An Overview at UCLAN Department of Computing Conference*, Preston, UK. [2016, August 17] http://www.chici.org/references/tangible_user_interface.pdf

1.5 Engaging Girls and Boys

Are there gender issues? Evidence shows that girls (relative to boys) start disengaging from IT and technology at age 7. Only 17% of Google's engineers, 15% of Facebook's, and 10% of Twitter's are women. (Zynczak, 2016.) 'The gender gap is not at all connected to innate ability; girls are not less likely than boys to be good at these subjects. In fact, The Engineer reports that in the UK girls outperform boys up to GCSE level (exams taken at age 16). But after this, the further on you go, the fewer women you find pursuing these subjects. For instance, women make up just 12% of engineering students at universities in the UK, and just 4% of those taking engineering apprenticeships.' (QS Staff Writer, 2012.) A similar situation is to be noticed in Germany.

'A statistical report by WISE (Women in Science and Engineering) (2015) shows that girls in the UK only begin losing interest in science as they grow into their teenage years.' Roughly equal numbers of boys and girls study STEM subjects at GCSE with girls marginally outperforming boys, the gap then widens significantly at A-level and again at undergraduate level with fewer girls continuing to study STEM subjects.

There is no evidence to suggest any physiological or cognitive reason for the gender gap, but the numbers suggest a gender specific technology socialisation. In gender research Buchen et al. (2006) noted that 'gender behavior is less rooted in socialization or even in the genetic personality structures, but is being renewed under specific conditions and in specific situations with regard to the understanding of technology' (cited on Schelhowe, 2007, p 131). Schelhowe emphasizes, that 'even the difficult access of girls to technology is not rooted in a fundamental technology distance or disinterest. Rather roots the ostensible interest in technology, which is found almost exclusively in (some) boys, and the distancing and the retreat to a purposive-rational use value-based access in girls rather in self-images, the fascination and passion (for technology) only for boys, or only in homogeneous girl groups, in mixed-gender context for girls are a hindrance' (Schelhowe, 2007, p 131). Even recent initiatives of promotion of girls, which are opposed to the lack of skilled workers/professionals in technology, show little girls specific support priorities with the combination of technology, art and design. (For example, the nationwide initiative of the Fraunhofer IAIS " Roberta" learning with robots, which had the goal of to support a sustainable ‘interest, especially of girls but also of boys for computer science, science and technology’).

‘The research suggests that perceived or actual differences in cognitive performance between males and females are most likely the result of social and cultural factors. For example, where girls and boys have differed on tests, researchers believe social context plays a significant role. Spelke believes that differences in career choices are due not to differing abilities but to cultural factors, such as subtle but pervasive gender expectations that kick in during high school and college.’ (American Psychological Association, 2014.)

The differences are best explained by stereotyping and social pressures. ‘The decline in STEM interest which begins in adolescence increasingly manifests itself in STEM participation in the last years of high school, when girls can express their own wants. Teenage girls score lower than boys on math SAT tests, take fewer AP tests in calculus, physics, and computer science, and are less likely to select college STEM majors. (AAUW, 2010.) We propose that these trends are connected to girls’ perception of STEM as masculine and their internalization of feminine norms. Girls are caught in a “double conformity” bind, in which they must opt out of femininity or opt out of STEM’. (Do Internalized Feminine…, s.a.)

In an attempt to redress the balance there are a growing number of initiatives aimed exclusively at girls, Google’s Made with Code project (see Why Coding is..., s.a.) promotes computer sciences to girls ‘For students today, coding is becoming an essential skill, just like reading, writing, and math. If you have a daughter, niece, or other girl that you know, encouraging her to learn to code can open up countless opportunities for her future. Whether she’s an athlete or an artist, loves animals, or wants to explore medicine, coding can help her pursue her interests now and create greater career options and job security for her future. Long story short? She can make anything with code.’

But these forms of positive discrimination are not without their problems. How do you appeal to girls without invoking traditional female stereotypes? Does pink lego really encourage girls to become engineers or does it reinforce the premise that pink is for girls whilst all of the other types of lego are for boys? The approach of the TACCLE 3 doubts this. Painting technology in pink colors is not an entrance, but contexts can be. Those learning environments which allow for a variety of physical materials, shapes and colours can be used as a vehicle to drive interest in technology. Like with physical computing, a wide range of materials and media are integrated and linked to the children’s imagination and fantasies.

‘The environment that we grow up in affects how our brain develops. Professor Rippon explains that any difference in brain circuitry comes through stereotyping. Our brain has an incredible ability to adapt, to learn, to accommodate, and to rewire (known as neuroplasticity), and to fit into our community we adopt the standards, beliefs, and prejudices of those we want to identify with – such as the role of our associated genders. This can help explain the drop in girls taking STEM subjects during their formative years between GCSEs and A level.’ (Clarke, 2015).

So girls and boys need to get the access, appropriate tools to experience technology in a different way, bringing together new contexts, new learning environments allowing the children to involve their imagination. Last not least the need role models like women in STEM to identify with as early as possible.

References

American Psychological Association (2014, August). *Think Again: Men and Women Share Cognitive Skills*. [2016, August 13] http://www.apa.org/action/resources/research-in-action/share.aspx

Clarke, T. (2015, March 10). Women and science: Time to cut the Neurotrash. *50.50 inclusive democracy*. [2016, September 5] https://www.opendemocracy.net/5050/tara-clarke/women-and-science-time-to-cut-neurotrash-0

Do Internalized Feminine Norms Depress Girls' STEM Attitudes & Participation? (s.a.) [2016, September 24] http://www.truechild.org/Images/Interior/learnthefacts/__femininity%20&%20stem.pdf

QS Staff Writer (2012, June 27). *The STEM Gender Gap in Universities*. [2016, August 13] http://www.topuniversities.com/courses/engineering/stem-gender-gap-universities

Schelhowe, H. (2007). Technologie, Imagination und Lernen: Grundlagen für Bildungsprozesse mit Digitalen Medien. Münster: Waxmann.

Zynczak, H. (2016). *What Closing The Gender Gap In Tech Would Mean Outside The Industry*. [2016, August 14] https://www.fastcompany.com/3055906/strong-female-lead/what-closing-the-gender-gap-in-tech-would-mean-outside-the-industry

Why Coding is kind of a Big Deal (s.a.). [2016, October 1] https://www.madewithcode.com/static/why-coding-is-kind-of-a-big-deal.pdf-v3.pdf

WISE (2015). *Women in Science, Technology Engineering and Mathematics: The Talent Pipeline from Classroom to Boardroom* [2016, October 16]

https://www.wisecampaign.org.uk/uploads/wise/files/WISE_UK_Statistics_2014.pdf

2. Computational thinking

2.1 Introduction

We live in software-driven world (Manovich, 2013) and current Society demands skilled professionals for ICT (Information and Communication Technologies) business sector. A very common situation in countries with a high rate of unemployment is they have unfilled positions for engineers and technicians for the industry and digital services. This has caused an increasing approach for introduce digital or information technology (IT) literacy from the early beginning of the individual development (Bers, Flannery, Kazakoff, & Sullivan, 2014; Cejka, Rogers, & Portsmore, 2006; Kazakoff & Bers, 2012) till the high school courses (Allan, Barr, Brylow, & Hambrusch, 2010), even in post-secondary institutions (Astrachan, Hambrusch, Peckham, & Settle, 2009), combining it with other key competences such as reading, writing and math skills.

New devices (Alonso de Castro, 2014; Sánchez Prieto, Olmos Migueláñez, & García-Peñalvo, 2013, 2014c), from smartphones and tablets to electronic learning toys and robots, find new audiences with increasingly young children. This causes new challenges for teachers (Sánchez Prieto, Olmos Migueláñez, & García-Peñalvo, 2014a, 2014b, 2016), for example how to define developmentally appropriate activities and content for children of different ages (Bers et al., 2014).

Whereas IT literacy is the capability to use today's technology in one's own field, the notion of IT fluency adds the capability to independently learn and use new technology as it evolves (National Research Council Committee on Information Technology Literacy, 1999) throughout one's professional lifetime. Moreover, IT fluency also includes the active use of algorithmic thinking (including programming) to solve problems, whereas IT literacy is more limited in scope.

The most frequent approach to teaching digital literacy has been to gradually encourage the learning of programming, and the term code-literacy (diSessa, 2000; Hockly, 2012; Prensky, 2008; Rushkoff, 2012; Vee, 2013) has been coined to referrer the process of teaching children programming tasks, from the simplest and most entertaining to the most complex, this way the student's progress is centred on the difficulty of the tasks and in their motivating characteristic. This means a link between the learning with the response to a stimulus instead to the child's learning and cognitive capabilities, following the traditional behaviourist theories (Zapata-Ros, 2015).

However, there exist an alternative in the constructionism approach, which was yet considered by Papert (1980) in his researches based on the Logo programming language, rooted in Piaget's (1954) constructivism that conveys the idea that the child actively builds knowledge through experience and the related "learn-by-doing" approach to education. This approach states that the competences that are shown as most effective in coding (in this document coding and computer programming are used interchangeably) are the most visible part of a way of thinking, which is useful not only in the field of cognitive activities used in the development and creation of programs and computer systems. This means, there is a specific way of thinking and organizing the ideas and representations that is propitious for the computational skills because of it promotes the analysis and the ideas interrelationship for the logic organisation and representation of the procedures. Those skills are enhanced with certain activities and certain learning environments from the early stages. It is the development of a specific thought, a computational thinking (Zapata-Ros, 2015).

Consequently, at the same time that children learn human languages, both for speaking and writing, natural languages, encompassing all matters related with the experimental sciences (physics, chemistry, biology, etc.), and humanity languages, involving social sciences and humanities, it is also necessary they learn digital languages, in which ones of the competences to be success in the digital world are included, using coding as the way to solve problems and computational thinking as working paradigm (Llorens-Largo, 2015).

With the awareness of the importance of digital skills and related information technology (eSkills), there are several proposals worldwide about the need to include coding from the curriculum of non-university levels, starting since primary education (or sooner) (Balanskat & Engelhardt, 2015; Brown et al., 2013), because of the code-literacy skills are becoming understood as a core element for STEM (Science, Technology, Engineering, & Mathematics) subjects (Gelman & Brenneman, 2004; Weintrop et al., 2016), computational thinking may play an important role in K-12 STEM education because computational modelling is an effective approach for learning challenging science and math concepts (Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009) and imaginative programming is the most crucial element of computing because it closely aligns mathematics with computing and in this way brings mathematics to life (Felleisen & Krishnamurthi, 2009).

For example, in Spain CODDII (*Conferencia de Directores y Decanos de Ingeniería Informática*) in junction with AENUI (*Asociación de Enseñantes Universitarios en Informática*) drew up the declaration for the inclusion of subjects related to science and informatics in secondary education (CODDII & AENUI, 2014) with five main recommendations: 1) all the technological subjects must be offered by all the educational centres, public and private ones; 2) the curriculum of science and informatics should be completed by all students; 3) it is absolutely necessary and urgent teacher training in this area; 4) complementarily these skills can be developed transversally into other non-technological subjects; and 5) schools are responsible for transmitting and guide students on the importance of fostering vocations in STEM (Science, Technology, Engineering and Mathematics) areas. Many European and worldwide stakeholders (educators, parents, economists, politicians and so on) think that students need some computing and coding skills because they help to understand today’s digitalised society and foster 21st century skills like critical thinking, problem-solving, collaboration, communication and creativity (Ananiadou & Claro, 2009; Balanskat & Engelhardt, 2015; Binkley et al., 2012), taking into account that Computational Thinking has been identified among the critical 21st century skills all students should develop (D. Barr, Harrison, & Conery, 2011).

However, the digital language, as sum of several languages, gathers different competences such as computational thinking, coding, eSkills, informational skills, audio-visual capabilities, etc. (Hockly, 2012).

A code-literate person means that can read and write in programming languages (Román-González, 2014), computational thinking is referred to the underlying problem-solving cognitive process that allows it. Thus, coding is a key way to enable computational thinking (Lye & Koh, 2014) and computational thinking may be applied to various kinds of problems that do not directly involve coding tasks (Wing, 2008).

Jeannette M. Wing (2006) states that computational thinking "involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science", with a very important message about this "computational thinking is a fundamental skill for everyone, not just for computer scientists".

If we have into account that computer science is the study of computation, what can be computed and how to compute it, computational thinking thus has the following characteristics (Wing, 2006):

- Computational thinking means conceptualizing, not programming. Computer science is not only coding. Thinking like a software engineering means more than being able to code. It requires thinking at multiple level of abstraction.
- Computational thinking requires fundamental skills, not rote skills. A fundamental skill is something every human being must know to function in modern society. Rote means a mechanical routine.
- Computational thinking is about how about humans, not computers, think. Computational thinking is regarding a way humans solve problems; it is not trying to get humans to think like computers. Computers are dull and boring; humans are clever and imaginative. Solving problems, with or without computers people need to have an imaginative and intelligent mind, it is also needed emotion and creativity. This is so near of the divergent thinking (Bono, 1968, 1970; Lieberman, 1965; Pólya, 1957).
- Computational thinking complements and combines mathematical and engineering thinking. Computer science inherently draws on mathematical thinking, given that, like all sciences, its formal foundations rest on mathematics. Computer science inherently draws on engineering thinking; given that we build systems that interact with the real world.
- Computational thinking relies in ideas, not in artefacts. It is not just the software or hardware artefacts we may create, it will be the computational concepts we may use to

approach and solve problems, manage our daily lives, and communicate and interact with other people.

- Computational thinking is for everyone, everywhere. Computational thinking will be a reality when it is so integral to human endeavours it disappears as an explicit philosophy.

Within a cognitive approach computational thinking is related to three abilities-factors (Ambrosio, Xavier, & Georges, 2014) from the Cattell-Horn-Carroll (CHC) model of intelligence (McGrew, 2009):

- Fluid reasoning, defined as: "the use of deliberate and controlled mental operations to solve novel problems that cannot be performed automatically" (McGrew, 2009).
- Visual processing, defined as "the ability to generate, store, retrieve, and transform visual images and sensations" (McGrew, 2009).
- Short-term memory, defined as "the ability to apprehend and maintain awareness of a limited number of elements of information in the immediate situation (events that occurred in the last minute or so)" (McGrew, 2009).

Despite of this increasing interest about introducing coding, code-literacy or computational thinking in the pre-university studies, there still exist a lack of consensus on a formal definition of these terms (V. Barr & Stephenson, 2011; Gouws, Bradshaw, & Wentworth, 2013; Grover & Pea, 2013) and very different positions about how they may be integrated in the official curricula of the countries (Lye & Koh, 2014).

2.2 Definition of computational thinking

The ACM computer science definition specifically for K-12 educators argues computer science is neither programming nor computer literacy, it is the study of computers and algorithmic processes including their principles, their hardware and software design, their applications, and their impact on society, including: programming, hardware, design, networks, graphics, databases and information retrieval, computer security, software design, programming languages and paradigms, logic, translation between levels of abstraction, artificial intelligence, the limits of computations (what computers cannot do), applications in information technology and information systems, and social issues (Internet security, privacy, intellectual property, etc.) (Tucker et al., 2006).

In framing the conceptual and educational importance of computational thinking, it is important to present clearly that both concepts, computational thinking and computer science are different.

As we stated above, the term computational thinking was made popular by Jeannette M. Wing (2006), with her definition "computational thinking involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science". Wing (2011b) revisited the topic and provided this new definition

“Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent”. Regarding computational thinking, Isbell et al. (2009) proposed a focus on providing services, interfaces, and behaviours that involves a more central role for modelling as a means of formulating relationships and identifying relevant agencies that are sources of change. Moreover, Riley and Hunt (2014) asserted that the best way to characterize computational thinking is as the way that computer scientists think, the manner in which they reason.

Aho (2012) simplified this concept defining it as the thought processes involved in formulating problems so “their solutions can be represented as computational steps and algorithms”.

The Royal Society (2012) offered the following definition “Computational thinking is the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes”.

Mannila et al. (2014) stated that computational thinking is a term covering a set of concepts and thinking processes from computer science that help in formulating problems and their solutions in different disciplines.

García-Peñalvo (2016b) defined computational thinking as the application of high level of abstraction and an algorithmic approach to solve any kind of problems.

Barr and Stephenson (2011) provided an operational definition of computational thinking as a “problem-solving process that includes (but is not limited to) the following characteristics: formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analysing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analysing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; generalizing and transferring this problem solving process to a wide variety of problems”.

The above definition is also in the report by The International Society for Technology in Education (ISTE) and Computer Science Teacher Association (CSTA) (CSTA & ISTE, 2011).

The general definitions of computational thinking may not be suited for programming, thus from a curricular perspective there are frameworks for developing computational thinking in the classroom. From EEUU, Brennan and Resnick (2012) describe a computational thinking framework that involves three key dimensions: computational concepts (sequences, loops, events, parallelism, conditionals, operators, and data); computational practices (experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularizing); and computational perspectives (expressing, connecting, and questioning). From UK, the organization Computing At School (CAS) states that computational thinking involves six different concepts: logic, algorithms, decomposition, patterns, abstraction, and evaluation; and five approaches to working in the classroom: tinkering, creating, debugging, persevering, and collaborating (CAS Barefoot, 2014).

However, as Hemmendinger (2010) stated, the ultimate computational thinking should not be to teach everyone to think like a computer scientific nor to convert every child in a software engineer, but rather to teach them to apply these common elements to solve problems and discover new questions to explore within and across all disciplines. Close to this approach Sysło and Kwiatkowska (2013) also underlined that computational thinking is a set of thinking skills that may not result in computer programming, it should focus on the principles of computing rather than on computer programming skills.

For closing this section, Figure 1 shows a word-cloud graph based on the words that appear in well-known definitions of computational thinking.

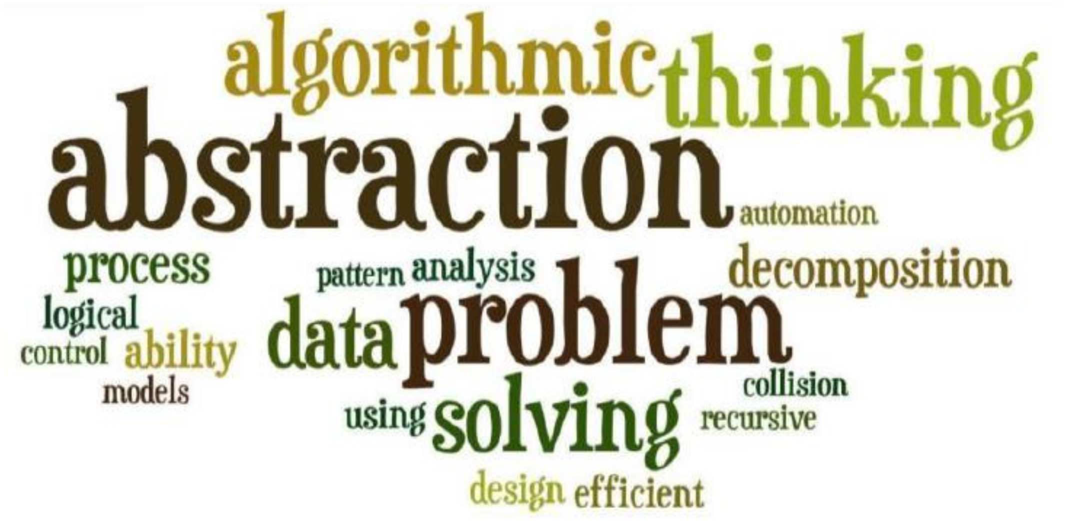


Figure 1. Most commonly used words in the definitions of computational thinking. Source (Kalelioglu, Gulbahar, & Kukul, 2016)

2.3 Computational thinking core concepts

Computational thinking in mainly an active problem solving methodology in the students use a set of concepts, such as abstraction or iteration among others, to process and analyse data, and to create real or virtual artefacts. Thus the goal of introducing ICT in pre-university curriculum is not the students become merely tool user but tool builders.

The power of computational thinking is that it may be applied to every other type of reasoning and enables all kind of things to get done in different subjects or knowledge areas (V. Barr & Stephenson, 2011).

Different core computational thinking set of components are proposed to define specific computational thinking frameworks.

Barr and Stephenson (2011) present a structured model that emerged focused on identifying core computational thinking concepts and capabilities. The core concepts are data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization and simulation. The capabilities are computer science, math, science, social studies and language arts.

Brennan and Resnick (2012) propose a computational thinking where the components are classified into three dimensions:

1. Computational concepts, which are the concepts that students employ when they code: sequences, loops, events, parallelism, conditionals, operators, and data.
2. Computational practices, which are problem solving practices that occur in the process of coding: experimenting and iterating, testing and debugging, reusing and mixing, and abstracting and modularisating.
3. Computational perspectives, which are the students' understandings of themselves, their relationships with others, and the digital world around them: expressing, connecting and questioning.

Gouws et al. (2013) design a computational thinking framework to serve as foundation for creating computational thinking resources. This framework is a two-dimensional grid. One dimension gathers the skill sets that make up computational thinking: processes and transformations, models and transformation, patterns and algorithms, inference and logic, and evaluations and improvements. The other dimension means the different levels at which these skills may be practiced: recognise, understand, apply, and assimilate.

Zapata-Ros (2015) tries to connect computational thinking with the learning theories conceptualizations and thinking models, proposing the following computational thinking components: bottom-up analysis, top-down analysis, heuristics, divergent thinking, creativity, problem solving, abstract thinking, recursion, iteration, Successive approximation methods (trial and error), collaborative methods, patterns, synectics and metacognition.

2.4 Computational thinking practices

As we stated above, computational thinking in an approach to problem solving using computer science techniques, described by Jeannette Wing (2008) as the mental tools that allow us to make the best use of our mental tools.

Due to computational thinking has different interpretations, there are different approaches for introducing this approach into the classrooms, including those that consider computational thinking provides transparent advantages focusing on semantics rather the syntax of a specific language; those that prefer some kind of programming environment, based on blocks such as scratch or based on most traditional coding languages; those that control robots; or those that build physical kits to control things.

Taccle 3 Coding project (García-Peñalvo, 2016a; TACCLE 3 Consortium, 2016) provides practical ideas that teachers can use immediately together with suggestions on how these can be adapted for introducing computing or coding in their classrooms. Many countries are introducing computing as a core curriculum subject. Some have already done so; many others are intending to. Inevitably the detail of the curricula will be different in each country but there is a substantial overlap - most all of the curricula available so far include programming (Aho, 2012), control technology (Atmatzidou & Demetriadis, 2016) and computational/logical thinking (Wing, 2011a), so Taccle3 has started with these. The following sections are centred on presenting the most outstanding computational thinking practices.

2.4.1 Computational thinking for developing mental models

The main idea behind this approach is computational thinking is not an alternative to learn coding; it is a way to reinforcing concepts and supplementing coding or programming education. The objective is that students may develop stronger mental models that ultimately make them better software engineers. Besides, the challenge is to use computational thinking approach to be useful and effective in a broader range of disciplines.

The CS Unplugged project (http://csunplugged.org/) by CS Education Research Group at the University of Canterbury in New Zealand is a good example of pedagogical and creativity activities oriented to introduce computational thinking principles without using a computer (Bell, Witten, & Fellows, 2016).

Games have been used extensively for achieving this goal. Game design is a popular way to teach programming to students who have little or no prior programming experience (Peppler & Kafai, 2007; Repenning, 2006). Gouws et al. (2013) use Light-Bot, an educational game whose objective is to program a small robot to light up all the blue blocks on a board. Authors applies their computational thinking framework to assess the skills and the levels at which these skills were practiced.

Basawapatna et al. define Computational Thinking Patterns, which are abstracted programming patterns that are learned by students when they create games and can readily be used by students to model scientific phenomena. Examples of games and their associated computational thinking patterns are: Frogger (Generation, Absorption, Collision, Transportation); Sokoban (Push, Pull); Centipede (Generation, Absorption, Collision, Push, Pull); Space Invaders (Generation, Absorption, Collision); or Sims (Diffusion, Hill Climbing) (Basawapatna, Koh, & Repenning, 2010; Basawapatna, Koh, Repenning, Webb, & Marshall, 2011).

2.4.2 Computational thinking through programming tools

It is better to use visual programming languages, as the motivational context with embedded computer science content (Bennett, Koh, & Repenning, 2011), rather than traditional programming languages to facilitate the three dimensions of computational thinking (concepts, practices and perspectives) specially in K-12 contexts (Lye & Koh, 2014).

With these block-based commands languages students usually need only to drag and snap the blocks, reducing the cognitive load on the students and allowing them to focus on the logic and structures involved in programming rather than the mechanics of writing programs (Kelleher & Pausch, 2005).

Visual languages present the following advantages to introduce computational thinking through programming (Lye & Koh, 2014):

- They make easy to enact the computational practices because the outcomes of their programming can be visualized as animated objects.
- They become technology-as-partner in the learning process (Howland, Jonassen, & Marra, 2011) and help students to extend the computational practices towards enhancing their general problem-solving ability (Lin & Liu, 2012).
- They engage students in the building of multi-media digital products, thereby enabling programming activities to be used as a means for students to express their ideas.
- They develop students' digital literacy for creating, sharing and remixing digital resources (Ng, 2012), thus they afford for such kinds of digital literacy experiences (Mills, 2010).
- They transform students, they will be no longer passive consumers of the technology (Resnick et al., 2009).

Scratch (Resnick et al., 2009) is a very popular programming language to learn coding, languages (Burke, 2012; Lee, 2010) or maths.

Logo is other programming language used to introduce coding (Lin & Liu, 2012), for example to help students with hearing disorders to learn English words (Miller, 2009) or to learn maths (Fessakis, Gouli, & Mavroudi, 2013).

Python is also used to introduce coding concepts with older students. Aiken et al. (2013) used VPython (http://vpython.org/) a high-level programming environment for a 9th-grade conceptual physics course in USA. Computational assignments followed in-class experiments and problem-solving sessions. After instruction, roughly a third of students were able to successfully complete an individual assessment in which they constructed a model of a new physical system. In this sense, SECANT project (http://secant.cs.purdue.edu/) (Ahamed et al., 2010) at Purdue University uses Python and VPython focusing on computational methods and visualizations. This project collaborated with two high school physics teachers to incorporate selected material of the

Matter&Interaction Curriculum with computational thinking principles into high school physics courses.

App Inventor 2 (http://appinventor.mit.edu/) (Walter & Sherman, 2015) is a cloud-based visual blocks language (Wolber, Abelson, & Friedman, 2015), which allows build Android apps using a web browser, without to code. The App Inventor service is available at ai2.appinventor.mit.edu. App Inventor has been used extensively in both primary and secondary schools. A good example of it is the Computational Thinking through Mobile Computing project (Turbak, Pokress, & Sherman, 2014), which is devoted to teach the big ideas of computer science to undergraduate students using App Inventor. This project is a collaboration between MIT, Wellesley College, Trinity College, University of Massachusetts Lowell, and University of San Francisco (https://nsfmobilect.wordpress.com/).

The user interface of App Inventor is based on the idea of low-floor, high-ceiling development environments (Papert, 1980), and consists of two parts: a Designer for selecting the components of the app, and a Blocks Editor for setting the behaviour of the app. App Inventor's building blocks are common user interface elements (buttons, labels, list pickers, images, etc.) coupled with the mobile device's features (texting, GPS, NFC, Bluetooth, etc.) (Pokress & Domínguez Veiga, 2013).

2.4.3 Computational thinking as key element for teaching and curriculum reform

The idea behind this approach is the claim of using a computational thinking approach in order to train future teachers' computational thinking ability.

Yu (2014) proposes that basic computer courses at the university level should embody many computational thinking methods, such as computer hardware components, various algorithms (sorting, recursion, etc.) in order to train students getting calculating thinking ability.

It is interesting the strength the connection between the computational thinking and the computational values support such as open publication of educational materials and resources, the policy on open publication of academic research, and the open online instruction based on non-proprietary software platforms (Ableson, 2012).

In 2010, nine Chinese universities included in the country's "985 project" explicitly pointed out that "prominent computational thinking ability must be the fundamental skill for innovative talents in any discipline" (Long, Zhang, & Li, 2013). College students' computational thinking is usually cultivated via a series of courses, some are directed to a certain discipline, some are disciplinary, and others are general to all disciplines. The result was the design of a categorized and multi-layer course system in accordance with (1) the teaching contents, the extensiveness of utilization and the level of difficulty of methods of computational thinking; (2) the frequency of its features and the difficulty of apprehension; and (3) students' ability of apprehension.

2.4.4 *Computational thinking assessment*

Developing assessments of student learning is an urgent area of need for the relatively young computer science education community as it advances toward the ranks of more mature disciplines (Buffum et al., 2015).

Frequent use of computational constructs is favoured, but their effects on final artefacts and the relation between final artefacts and pre-defined learning goals are rarely considered (Basu, Kinnebrew, & Biswas, 2014).

Interesting contributions regarding the measure and the assessment of computational thinking are the Fairy Assessment (Werner, Denner, Campe, & Kawamoto, 2012), which tries to measure the understanding and use of different computational concepts that students utilize to solve problem.

Koh et al. (2010) identify several computational thinking patterns that young students abstract and develop during the creation of video-games in a controlled environment; they create an automated tool that analyses the games programmed and represents graphically how far each game has involved the different patterns when compared with a model.

Basu et al. (2014) propose a more systematic assessment of Computational Thinking based science learning, using CTSiM – a Computational Thinking based science learning environment. In this environment, students first construct a conceptual model and then design a corresponding computational model for a given science phenomenon. Authors designed and independent system of pre-post assessments for science and Computational Thinking and developed vector-distance, effectiveness, and consistency measures to characterize student models. Using these assessments, they show that students gained significantly on both science and Computational Thinking content in Kinematics and Ecology subjects.

Dr. Scratch (http://drscratch.org/) (Moreno-León & Robles, 2015a, 2015b; Moreno-León, Robles, & Román-González, 2015) is a web application that analyses automatically Scratch projects and gives feedback to improve programming skills and to develop computational skills. The research group behind Dr. Scratch has developed a computational thinking test (Román-González, 2015a, 2015b) and has demonstrate its convergent validity with respect to other traditional software quality and complexity metrics (Moreno-León, Robles, & Román-González, 2016).

Quiz Maker, for the creation of quizzes, and Quizly are a couple of tools for assessing and automatically grading exercises done through App Inventor (Maiorana, Giordano, & Morelli, 2015). These tools mean an assessment platform able to manage students and classes, administer and propose formative, summative and informal tests and to be able to track user progress as well as the question solving process.

Regarding App Inventor, Mark Sherman and Fred Martin (2015) proposed a rubric for analysing the mobile computational thinking as represented in App Inventor work products.

2.4.5 Teacher Training Programs in Computational Thinking

Bringing Computational Thinking into pre-university is also rife with challenges which must be addressed. The most important amongst these is preparing in-service and future teachers to tackle the challenge of teaching Computational Thinking.

Some workshops focus on K-12 students, such as (Franklin et al., 2015), but provide research and advice for best practices like curriculum, content delivery, interfacing with schools, and even classroom layout.

CS4HS has funded face-to-face workshops oriented to directly train teachers, like (Blum & Cortina, 2007; Bort & Brylow, 2013; Liu, Lin, Hasson, & Barnett, 2011), that have shown to not only increase participants understanding of Computational Thinking and Computer Science, but also how to integrate it as a part of their curriculum. CS4HS has also began introducing MOOCs (Massive Open Online Course) targeted for training teachers (Spradling et al., 2015).

Bean et al. (2015) develop a program for future teachers piloted at Kansas State University based on Scratch.

References

Ableson, H. (2012). From Computational Thinking to Computational Values. *Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12* (pp. 239). New York, NY, USA: ACM.

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Early, J. P. (2010). Computational thinking for the sciences: A three day workshop for high school science teachers. *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 42-46). Milwaukee, Wisconsin, USA: ACM.

Aho, A. V. (2012). Computation and Computational Thinking. *Computer Journal, 55*(7), 832-835. doi:10.1093/comjnl/bxs074

Aiken, J. M., Caballero, M. D., Douglas, S. S., Burk, J. B., Scanlon, E. M., Thoms, B. D., & Schatz, M. F. (2013). Understanding Student Computational Thinking with Computational Modeling. In P. V. Engelhardt, A. D. Churukian, & N. S. Rebello (Eds.), *2012 Physics Education Research Conference* (Vol. 1513, pp. 46-49).

Allan, V., Barr, V., Brylow, D., & Hambrusch, S. (2010). Computational thinking in high school courses *Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE'10* (pp. 390-391). New York, NY, USA: ACM.

Alonso de Castro, M. G. (2014). Educational projects based on mobile learning. *Education in the Knowledge Society, 15*(1), 10-19.

Ambrosio, A. P., Xavier, C., & Georges, F. (2014). Digital ink for cognitive assessment of computational thinking *Proceedings of the 2014 IEEE Frontiers in Education Conference (FIE) (Madrid, Spain, 22-25 Oct. 2014)* (pp. 1-7). EEUU: IEEE.

Ananiadou, K., & Claro, M. (2009). 21st Century skills and competences for new millennium learners in OECD Countries. *OECD Education Working Papers, 41*.

Astrachan, O., Hambrusch, S., Peckham, J., & Settle, A. (2009). The present and future of computational thinking. *SIGCSE Bulletin Inroads, 41*(1), 549-550. doi:10.1145/1539024.1509053

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661-670. doi: 10.1016/j.robot.2015.10.008

Balanskat, A., & Engelhardt, K. (2015). *Computing our future. Computer programming and coding Priorities, school curricula and initiatives across Europe*. [2016, August 5] http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf/746e36b1-e1a6-4bf1-8105-ea27c0d2bbe0

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology, 38*(6), 20-23.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and What is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54. doi:10.1145/1929887.1929905

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using scalable game design to teach computer science from middle school to graduate school. *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education, ITiCSE '10* (pp. 224-228). New York, USA: ACM.

Basawapatna, A. R., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 245-250). New York, USA: ACM.

Basu, S., Kinnebrew, J. S., & Biswas, G. (2014). Assessing Student Performance in a Computational-Thinking Based Science Learning Environment. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Intelligent Tutoring Systems, Its 2014* (Vol. 8474, pp. 476-481).

Bean, N., Weese, J., Feldhausen, R., & Bell, R. S. (2015). Starting from Scratch Developing a Pre-Service Teacher Training Program in Computational Thinking. *2015 IEEE Frontiers in Education Conference (FIE), October 21-24, 2015, Camino Real El Paso, El Paso, TX, USA* (pp. 1307-1314). EEUU: IEEE.

Bell, T., Witten, I. H., & Fellows, M. (2016). *CS Unplugged. An enrichment and extension programme for primary-aged students. Version 3.2.2*. New Zealand: University of Canterbury. CS Education Research Group.

Bennett, V., Koh, K., & Repenning, A. (2011). Computing learning acquisition? *Proceedings of the 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 18-22 Sept. 2011, Pittsburgh, PA, USA* (pp. 243-244). EEUU: IEEE.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education, 72*, 145-157. doi: 10.1016/j.compedu.2013.10.020

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). Defining twenty-first century skills. In P. Griffin, B. McGaw, & E. Care (Eds.), *Assessment and teaching of 21st century skills* (pp. 17–66). Netherlands: Springer.

Blum, L., & Cortina, T. J. (2007). CS4HS: an outreach program for high school CS teachers. *SIGCSE Bull., 39*(1), 19-23. doi:10.1145/1227504.1227320

Bono, E. D. (1968). *New think: The use of lateral thinking in the generation of new ideas*. New York, NY: Basic Books.

Bono, E. D. (1970). *Lateral Thinking. A Textbook of Creativity*. London: Ward Lock Educational.

Bort, H., & Brylow, D. (2013). CS4Impact: measuring computational thinking concepts present in CS4HS participant lesson plans *Proceeding of the 44th ACM technical symposium on Computer Science Education, SIGCSE '13, March 6-9, 2013, Denver, Colorado, USA* (pp. 427-432). New York, NY, USA: ACM.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Annual American Educational Research Association meeting, Vancouver, BC, Canada.

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing computer science back into schools: Lessons from the UK *Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13* (pp. 269-274). New York, USA: ACM.

Buffum, P. S., Lobene, E. V., Frankosky, M. H., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2015). A Practical Guide to Developing and Validating Computer Science Knowledge Assessments with Application to Middle School *Proceedings of the 46th ACM Technical Symposium on Computer Science Education. SIGCSE '15 (Kansas City, Missouri, USA, March 4th-7th)* (pp. 622-627). New York, NY, USA: ACM.

Burke, Q. (2012). The markings of a new pencil: Introducing programming-aswriting in the middle school classroom. *Journal of Media Literacy Education, 4*(2), 121–135.

CAS Barefoot. (2014). Computational Thinking. [2016, July 18] http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/

Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education, 22*(4), 711–722.

CODDII, & AENUI. (2014). Por la inclusión de asignaturas específicas de ciencia y tecnología informática en los estudios básicos de la enseñanza secundaria y bachillerato. *ReVisión, 7*(2), 5-7.

CSTA, & ISTE. (2011). *Computational Thinking in K–12 Education leadership toolkit*. [2016, August 7] https://csta.acm.org/Curriculum/sub/CurrFiles/471.11CTLeadershiptToolkit-SP-vF.pdf

diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge: MIT Press.

Felleisen, M., & Krishnamurthi, S. (2009). Why computer science doesn't matter. *Communications of the ACM, 52*(7), 37-40. doi:10.1145/1538788.1538803

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97. doi:http://dx.doi.org/10.1016/j.compedu.2012.11.016

Franklin, D., Hill, C., Dwyer, H., Iveland, A., Killian, A., & Harlow, D. (2015). Getting Started in Teaching and Researching Computer Science in the Elementary Classroom *Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE '15, March 4-7, 2015, Kansas City, Missouri* (pp. 552-557). New York, NY, USA: ACM.

García-Peñalvo, F. J. (2016a). *Proyecto TACCLE3 – Coding.* Paper presented at the XVIII Simposio Internacional de Informática Educativa, SIIE 2016, Salamanca, España.

García-Peñalvo, F. J. (2016b). What Computational Thinking Is. *Journal of Information Technology Research, 9*(3).

Gelman, R., & Brenneman, K. (2004). Science learning pathways for young children. *Early Childhood Research Quarterly, 19*(1), 150–158.

Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities: An evaluation of the educational game light-bot *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '13* (pp. 10-15). New York, NY, USA: ACM.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. doi:10.3102/0013189X12463051

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors *Proceedings of the 40th ACM technical symposium on Computer Science Education, SIGCSE '09, March 4-7, 2009, Chattanooga, TN USA* (pp. 183-187). New York, NY, USA: ACM.

Hemmendinger, D. (2010). A plea for modesty. *ACM Inroads, 1*(2), 4-7. doi:10.1145/1805724.1805725

Hockly, N. (2012). Digital literacies. *ELT Journal, 66*(1), 108-112. doi:10.1093/elt/ccr077

Howland, J., Jonassen, D. H., & Marra, R. M. (2011). *Meaningful learning with technology* (4th ed.). Upper Saddle River, NJ, USA: Pearson.

Isbell, C., Stein, A., Cutler, R., Forber, J., Fraser, L., Impagliazzo, J., . . . Y., X. (2009). (Re)Defining computing curricula by (re)defining computing. *ACM SIGCSE Bulletin, 41*(4), 195-207. doi:10.1145/1709424.1709462

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing, 4*(3), 583-596.

Kazakoff, E., & Bers, M. (2012). Programming in a robotics context in the kindergarten classroom: the impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia, 21*(4), 371–391.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 83-137. doi:10.1145/1089733.1089734

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning *2010 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2010 (Leganés, Madrid, Spain, 21-25 Sept. 2010)* (pp. 59 - 66). USA: IEEE.

Lee, Y.-J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia, 19*(3), 307–326.

Lieberman, J. N. (1965). Playfulness and Divergent Thinking: An Investigation of their Relationship at the Kindergarten Level. *The Journal of Genetic Psychology: Research and Theory on Human Development, 107*(2), 219-224. doi:10.1080/00221325.1965.10533661

Lin, J. M., & Liu, S. F. (2012). An investigation into parent-child collaboration in learning computer programming. *Educational Technology and Society, 15*(1), 162-173.

Liu, J., Lin, C.-H., Hasson, E. P., & Barnett, Z. D. (2011). Introducing computer science to K-12 through a summer computing workshop for teachers *Proceedings of the 42nd ACM technical symposium on Computer Science Education, SIGCSE '11, March 9-12, 2011, Dallas, TX, USA* (pp. 389-394). New York, NY, USA: ACM.

Llorens-Largo, F. (2015). Dicen por ahí… que la nueva alfabetización pasa por la programación. *ReVisión, 8*(2), 11-14.

Long, X., Zhang, J., & Li, Z. (2013). The Design of Teaching Structure based on Competency Training of Computational Thinking *Proceedings of the International Conference on Education Technology and Information System (ICETIS 2013)* (pp. 421-425). Amsterdam, The Netherlands: Atlantis Press.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. doi: 10.1016/j.chb.2014.09.012

Maiorana, F., Giordano, D., & Morelli, R. (2015). Quizly: A live coding assessment platform for App Inventor *Proceedings of the 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (pp. 25-30). USA: IEEE.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference* (pp. 1-29). New York, USA: ACM.

Manovich, L. (2013). *Software Takes Command*. New York, USA: Bloomsbury.

McGrew, K. S. (2009). CHC theory and the human cognitive abilities project: Standing on the shoulders of the giants of psychometric intelligence research. *Intelligence, 37*(1), 1-10. doi:http://dx.doi.org/10.1016/j.intell.2008.08.004

Miller, P. (2009). Learning with a missing sense: What can we learn from the interaction of a deaf child with a turtle? *American Annals of the Deaf, 154*(1), 71-82.

Mills, K. A. (2010). A review of the ‘‘digital turn’’ in the new literacy studies. *Review of Educational Research, 80*(2), 246–271. doi:10.3102/0034654310364401

Moreno-León, J., & Robles, G. (2015a). *Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills*. Paper presented at the Scratch Conference, Amsterdam, The Netherlands. http://jemole.me/replication/2015scratch/InferCT.pdf

Moreno-León, J., & Robles, G. (2015b). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects *Proceedings of the Workshop in Primary and Secondary Computing Education, WiPSCE '15 (London, United Kingdom, November 9-11, 2015)* (pp. 132-133). New York, NY, USA: ACM.

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED, Revista de Educación a distancia, 46*.

Moreno-León, J., Robles, G., & Román-González, M. (2016). Comparing Computational Thinking Development Assessment Scores with Software Complexity Metrics *Proceedings of the IEEE Global Engineering Education Conference, EDUCON 2016, Abu Dahbi, United Arab Emirates, 11-13 April, 2016*. EEUU: IEEE.

National Research Council Committee on Information Technology Literacy. (1999). *Being Fluent with Information Technology*. Washington, DC: National Academy Press.

Ng, W. (2012). Can we teach digital natives digital literacy? *Computers & Education*, *59*(3), 1065–1078. doi: 10.1016/j.compedu.2012.04.016

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY, USA: Basic Books.

Peppler, K. A., & Kafai, Y. B. (2007). Collaboration, computation, and creativity: media arts practices in urban youth culture *Proceedings of the 8th international conference on Computer supported collaborative learning* (pp. 590-592). New Brunswick, New Jersey, USA: International Society of the Learning Sciences.

Piaget, J. (1954). *The construction of reality in the child*. New York, USA: Basic Books.

Pokress, S. C., & Domínguez Veiga, J. J. (2013). MIT App Inventor. Enabling personal mobile computing *Proceedings of Programming for Mobile and Touch, PRoMoTo 2013*.

Pólya, G. (1957). *How to Solve It. A new aspect of mathematical method* (2nd ed.). Priceton, New Jersey: Princeton University Press.

Prensky, M. (2008). Programming Is the New Literacy. Retrieved from http://www.edutopia.org/literacy-computer-programming

Repenning, A. (2006). Excuse me, I need better AI!: employing collaborative diffusion to make game AI child's play *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (pp. 169-178). Boston, Massachusetts: ACM.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: Programming for all. *Commuication of the ACM, 52*(11), 60-67. doi:10.1145/1592761.1592779

Riley, D. D., & Hunt, K. A. (2014). *Computational thinking for the modern problem Solver*. Boca Raton, FL, USA: CRC Press.

Román-González, M. (2014). Aprender a programar 'apps' como enriquecimiento curricular en alumnado de alta capacidad. *Bordón. Revista de Pedagogía, 66*(4), 135-155. doi:http://dx.doi.org/10.13042/Bordon.2014.66401

Román-González, M. (2015a). Computational Thinking Test: Design guidelines and content validation *Proceedings of EDULEARN15 Conference, 6th-8th July 2015, Barcelona, Spain* (pp. 2436-2444).

Román-González, M. (2015b). Test de Pensamiento Computacional: principios de diseño, validación de contenido y análisis de ítems. In M. Á. Murga Menoyo (Ed.), *Perspectivas y avances de la investigación* (pp. 279-302). España: UNED.

Royal Society. (2012). *Shut down or restart: The way forward for computing in UK schools*. [2016, May 24] https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Rushkoff, D. (2012). Code Literacy: A 21st-Century Requirement. [2016, April 3] http://www.edutopia.org/blog/code-literacy-21st-century-requirement-douglas-rushkoff

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2013). Mobile Learning: Tendencies and Lines of Research. In F. J. García-Peñalvo (Ed.), *Proceedings of the First International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'13) (Salamanca, Spain, November 14-15, 2013)* (pp. 473-480). New York, NY, USA: ACM.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2014a). ICTs Integration in Education: Mobile Learning and the Technology Acceptance Model (TAM). In F. J. García-Peñalvo (Ed.), *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'14)* (pp. 683-687). New York, USA: ACM.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2014b). Mobile Learning Adoption from Informal into Formal: An Extended TAM Model to Measure Mobile Acceptance among Teachers. In F. J. García-Peñalvo (Ed.), *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'14)* (pp. 595-602). New York, USA: ACM.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2014c). Understanding mobile learning: devices, pedagogical implications and research lines. *Education in the Knowledge Society, 15*(1), 20-42.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2016). Informal Tools in Formal Contexts: Development of a Model to Assess the Acceptance of Mobile Technologies among Teachers. *Computers in Human Behavior, 55A*, 519-528. doi:http://dx.doi.org/10.1016/j.chb.2015.07.002

Sherman, M., & Martin, F. (2015). The assessment of mobile computational thinking. *Journal of Computing Sciences in Colleges, 30*(6), 53-59.

Spradling, C., Linville, D., Rogers, M., J. Clark, Computing, & Sciences in Colleges, v., no. 5, pp. 115-125, 2015. (2015). Are MOOCs an appropriate pedagogy for training K-12 teachers' computer science concepts? *Computing Sciences in Colleges, 30*(5), 115-125.

Sysło, M. M., & Kwiatkowska, A. B. (2013). Informatics for All High School Students: A Computational Thinking Approach. In I. Diethelm & R. T. Mittermeir (Eds.), *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages. 6th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2013,* *Oldenburg, Germany, February 26–March 2, 2013. Proceedings* (pp. 43–56). Heidelberg: Springer.

TACCLE 3 Consortium. (2016). TACCLE 3: Coding Erasmus + Project website. Retrieved from http://www.taccle3.eu/

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2006). *A model curriculum for K-12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee* (2nd ed.). New York, NY, USA: ACM.

Turbak, F., Pokress, S. C., & Sherman, M. (2014). Mobile computational thinking with APP inventor 2. *Journal of Computing Sciences in Colleges, 29*(6), 15-17.

Vee, A. (2013). Understanding Computer Programming as a Literacy. *LiCS. Literacy in Composition Studies, 1*(2), 42-64.

Walter, D., & Sherman, M. (2015). *Learning MIT App Inventor: A Hands-On Guide to Building Your Own Android Apps*. Upper Saddle River, NJ, USA: Addison-Wesley.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127-147. doi:10.1007/s10956-015-9581-5

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: measuring computational thinking in middle school *Proceedings of the 43rd ACM technical symposium on Computer Science Education, SIGCSE '12* (pp. 215-220). New York, NY, USA: ACM.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35. doi:10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences, 366*(1881), 3717-3725. doi:10.1098/rsta.2008.0118

Wing, J. M. (2011a). Computational Thinking. In G. Costagliola, A. Ko, A. Cypher, J. Nichols, C. Scaffidi, C. Kelleher, & B. Myers (Eds.), *2011 Ieee Symposium on Visual Languages and Human-Centric Computing* (pp. 3-3).

Wing, J. M. (2011b). Research Notebook: Computational Thinking--What and Why? *The Link. The magazine of the Carnegie Mellon University School of Computer Science*.

Wolber, D., Abelson, H., & Friedman, M. (2015). Democratizing Computing with App Inventor. *GetMobile: Mobile Computing and Communications, 18*(4), 53-58. doi:10.1145/2721914.2721935

Yu, D. (2014). The Research on Teaching Reform of the "University Computer Basic" Course In Police Active Colleges *Proceedings of the 2nd International Conference on Education*

Technology and Information System (ICETIS 2014) (pp. 583-586). Amsterdam: Atlantis Press.

Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital. *RED, Revista de Educación a distancia, 46*.

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41,* 51-61.

3. Smart Textile with Lilypad Arduino technology experiences

3.1. Introduction

In the following, a literature review related to a cross-disciplinary approach bringing together, computer science, arts, design and crafting processes is introduced. Also the developments and debates towards coding in education and its perception (especially in Germany) are presented. Smart textile opens up creative processes and the shaping of technology. It enables the pupils to integrate the haptic dimension and include a variety of the human senses for learning. An example of integrating physical computing and arts- and design-based learning is presented and discussed: Smart textile, self-made wearables become meaningful interactive objects in project- and team based learning.

3.1.1. The German context

In Germany, the integration of coding into the school curriculum is not realized yet, especially not at primary school level. The disciplines like technology didactics and computer science education developed separately from media education and media didactics. This also applies to esthetic education, wherein computational thinking and modeling was neither a learning goal to aspire to, nor a part of the curriculum. Computer science-based contents have hardly ever been integrated into digital media curricula. Programming competence which goes beyond the familiarity with a machine that computes, and the support of software application skills, was not identified a learning goal to aspire to. Programming skills, known as "code literacy", include the understanding of how computers and software programs work. This can be achieved by opening the black box and make is more transparent. Meanwhile programming, it has been accepted as a key skill for participation in the digital age. The algorithms underlying digital technologies partly are understood to be the “basis of a new world language" (Kreye, A., 2014). Such a perception of programming was not always the case. Computational modeling outside the computer science classes was not considered a relevant area of learning in the context of media education. Digital media and their media-specific characteristics, as programmed and programmable artefacts which are interactive, were not addressed. From 2001 to 2003, a German model project (ArtDeCom) brought together art, design and computer science in general education, including primary school level. The introduction of tangible media and programming environments for kids using specific iconic interface was realized (Reimann et al 2003; Reimann, 2006). The project included the identification of digital technologies to be shaped and programmed by the pupils themselves, getting away from screen based learning infront of single workplaces in computer classrooms. The approach was funded and through the German BLK-programme of “Cultural Education in the Media age”, follow-up projects (MediaArtLab@School, University of Flensburg) and wider initiatives (e.g. Kids in media and motion/KiMM, University of Lübeck) addressed and explored the computer technology as medium for creative processes. Education was explored across the borders of the curricula.

In the last years, more approaches were developed in the context of the Do-it-yourself practices and *Maker*-movement. A trend towards transdisciplinary and informal learning was mirrored in media labs in after school programs and publications such as handbooks addressed 'easy programming and digital creation' like the 'Making activities with children and young people' (Schön et al., 2016, p. 81). Those concepts went beyond technocratic conceptions of technology education integrating meaningful creative environments and artistic contexts. The potential of the arts, and design processes were discovered and realized in more arts based learning concepts and explored with different target groups of young people e.g. in general education (Reimann, 2006), in vocational education (2011) as well as in vocational preparation (pre-training) (Reimann & Bekk, 2014, 2015).

Even from the computer science education research, there are some projects and initiatives which introduce physical computing and more creative design- and crafts-based approaches to learn about computer science, e.g 'My interactive garden' (Przybylla & Romeike, 2013, p. 87-91). It is based on the Arduino technology and consists of interactive objects (such as 'magic flowers') pupils develop and put together in the framework of the interactive garden:

'Today computer science is more present than ever in society, because of interactive and embedded computing systems. Nevertheless, many students take computer science as a school subject with abstract and unrealistic contents. To counteract this and make computer science accessible to a wider population of pupils, the teaching concept "My Interactive Garden" was developed, which includes a design, programming and learning environment'. (Przybylla & Romeike, 2013, p. 87). The initiators of interactive garden aim to support creative education scenarios within constructionistic methods to offer less computer affine pupils an attractive access to computer science. The didactic considerations are to support a concrete practical experience for pupils at school. In German media pedagogy, the issue of "easy programming and digitally designing" (Schön et al., 2016) was perceived important rather late, e.g. through the widespread of "Scratch", the open source software for kids developed at MIT Lifelong Kindergarten (Resnick, s.a.).

3.1.3 Smart textile as a creative environment for programming interactive objects

Why should school teachers be interested in Smart textile, especially if they are neither admitted to textile nor to technology? This question will be answered in the following. Interactive textiles, which are also referred to as 'smart textiles' or 'wearables' represent a current generation of clothes and accessories with embedded microcomputers. They offer various possibilities of creatively dealing with so-called 'intelligent media', and intelligent in this case refers to their ability to perceive their environment by means of programmed microsensors. Using e.g. conductive yarn (as cable), sensors, motors, LED lights as well as sewable circuit boards (Arduino LilyPad introduced by Buechley in 2006, smart textiles create a link between sensual-haptic materials, precise computer control, and creative concepts. Smart textile as physical computing offers a space, wherein the acting with technology becomes a physical and aesthetic experience. New interfaces – sewed, woven or stitched – can be experienced between body, clothing, and the environment. It can be stitched together with conductive thread to create interactive garments and accessories. In conjunction with the open-source Arduino technology, they are currently open up opportunities for cross disciplinary teaching, of art, design, computer science and music, for example to address learning in the context of storytelling wearables (Tan, 2005), wearable music (Rosales, 2012) or sounding artifacts (Trappe, 2012). The Arduino LilyPad technology consists of hard components as well as a programming interface which can be connected to an icon based interface to be used by younger children at primary school level. The LilyPad can "sense information about the environment using inputs like light and temperature sensors and can act on the environment with outputs like LED lights, vibrator motors, and speakers." (see Web site http://lilypadarduino.org/)

'The **'eduwear starter kit'** (developed in context of the European project 'eduwear' by dimeb the digital media in education research group at the University of Bremen in 2008). It *c*ombines crafting, electronics, and programming. kids and adults of all ages interested in interactive toys, smart accessories, or light-up fashions can develop their own project according to their imagination. Pupils play with the components and learn to sew, program, and design circuits along the way. It's a technology for all ages and a new way to introduce pupils to technology by means of more arts design-based learning, including those kids who are less interested in engineering, e.g. the target group of girls or pupils with little interest in technology. The combining of art and design driven processes with learning and technology, is expected to trigger identification with a project and a deeper interest and understanding of technology. Since, however, the handling of the software and hardware used in the project is documented only insufficiently in German, it was decided to write down in a structured way of the experiences gained. Although the resultant tutorial does not claim to discuss all software and hardware issues, relevant problems need to be explained in detail. The tutorial was developed on the basis of the EduWear manual compiled by the "Digital media in Education (dimeb)" research group of the University of Bremen. (see Web site www.dimeb.de)

The **lesson plans** developed for the *TACCLE3 coding* project in Germany, includes the extended starter tutorial, which introduces the teacher both to the handling of the LilyPad Arduino hardware and to the application of the AMICI user interface and can be used as instructions for teaching processes related to interactive clothing. The introduction of tangible media and programming environments for kids with an iconic interface, such as *amici*, was realized.

The LilyPad Arduino hardware is usable for children of different age groups, able to use needles and yarn, or getting enough time to practice stitching and sewing**.** The core of the system consists of microcontroller LilyPad (see Web site http://lilypadarduino.org/), which was developed on the basis of microcontroller Arduino. Arduino exists in different versions. Whereas some of these are simple improvements of the original Arduino, others e.g., the LilyPad for textile uses, have been designed especially for particular uses. Inventor of the LilyPad Leah Buechley (software developer at MIT) had the idea to use this microcontroller in the area of textiles to create synergy effects.

Based on the above, the project EduWear (see Web site http://dimeb.informatik.uni-bremen.de/eduwear) was launched with the objective of acquainting children with the related technology. The starter kit of eduwear has been used at the Karlsruhe Institute of Technology (KIT) within the framework of the BMBF-project “MediaArt@Edu” (Acronym) (see Web site http://www.ibap.kit.edu/mediaartedu/). Within that project, the components and skills for working with the system were tested and evaluated in different institutions. An additional tutorial (Reimann et al., 2015) was developed to introduce smart textile hands on workshops in education, to be adapted to different target groups.

3.2 Literature review and findings

The Smart textile Lilypad-Arduino-technology was invented by Leah Buechley of the MIT High-Low-Tech research group to embed micro computers into textile. She introduced the e-textile in the higher education curriculum. Her work was perceived in Germany as well, g.e. by Schelhowe who developed the eduwear starter kit described above. In ‘making things wearable’ Bohne (2012) introduced a hands on starter kit for beginners to smart textile and do it yourself wearable targeting on people interested in hands on practice (makers). Wearable computing refers to computer systems that are integrated into clothing or - such as a heart rate monitor - be worn on the body. In 'Making things wearable', tailor made do it yourself smart clothing, modern textile processing tools and electronic components are used to produce wearable computing clothes themselves. The book provides the electronic knowledge of this top-modern DIY variant, showing through concrete workshop step by step with numerous color photographs and illustrations, how a LilyPad (the wearable computing sister of Arduino), LEDs and sensors are integrated in trendy clothes. Programming sensor and actuator-based interactive systems, including a playful, textile-integrated approach to be explored by children from primary school level on. How to use smart textile Arduino LilyPad Technology. It includes an instruction to do it yourself smart textile with Arduino LilyPad.

Buechleys work was strongly influenced by the MIT tradition of constructionist technology education and the basic ideas of Seymour Papert, which he developed in collaboration with Piaget in the 80s, when computer science met constructivist pedagogy, learning sciences and creativity. In his book ‘Mindstorms’ Seymour Papert (1982) introduced the issue of children becoming constructors and programmers of computers using their powerful ideas and imagination, as well as more visual approaches to technology and programming. It became a time-independent standard work, wherein his constructionist approach to technology education developed in close collaboration with the work of Piaget is presented. Children were taught to program computers, rather than the machine instructing children. He discusses how learning to learn of children is supported in any subject through programming computers; how an understanding of technology is supported by controlling and shaping it, making children control the computer, rather than using computers to control children. “Mindstorms has two central themes: that children can learn to use computers in a masterful way and that learning to use computers can change the way they learn everything else. Even outside the classroom, Papert had a vision that the computer could be used just as casually and as personally for a diversity of purposes throughout a person's entire life. Seymour Papert makes the point that in classrooms saturated with technology there is actually more socialization and that the technology often contributes to greater interaction among students and among students and instructors. The Lego Mindstorms programmable construction set system is named after the book. Also Papert’s ‘The children’s machine’ (1993) is a milestone, introducing the rethinking of the institution of the school in the age of the computer, in terms of the learning culture, the engineering culture and curriculum in the context of more *constructionistic* technology education, developed from the constructivist idea of the child developing his/her own learning by doing and from primary level on: “The optimistic note of this book comes from recognizing the potential synergy of two trends in the world. One of these trends is technological. The same technological revolution that has been responsible for the acute need for better learning also offers the means to take effective action. Information technologies, from television to computers and all their combinations, open unprecedented opportunities for action in improving the quality of the learning environment, by which I mean the whole set of conditions that contribute to shaping learning in work, in school, and in play. The other trend is epistemological, a revolution in thinking about knowledge. The central thesis of this book is that the powerful contribution of the new technologies in the enhancement of learning is the creation of personal media capable of supporting a wide range of intellectual styles. Women and members of minority cultures have seen most articulate in protesting the imposition of a single, uniform way of learning. Most have scarcely begun to use the new media to express and develop their particular voices. But it is children who have most visibly demonstrated the energizing effect of media that match their intellectual preferences. They have the most to gain and they have the most to give. Across the world children have entered a passionate and enduring love affair with the computer. What they do with computers is as varied as their activities. The greatest amount of time is devoted to playing games, with the result that names like Nintendo have become household words. They use computers to write, to draw, to communicate, to obtain information. Some use computers as a means to establish social ties, while others use them to isolate themselves. In many cases their zeal has such force that it brings the word addiction to the minds of concerned parents. (Papert, 1993.)

In ‘robots for kids’ Druin, and Hendler (2000) explore new technologies for learning. The introduce the issue of emotional meaningfulness to robots as ‘evocative objects’ (Turkle, 2011) to tell stories meaningful to the kids. By integrating children into the learning and teaching at primary school level process as partners at eye level, the become co-design partners, reporting their perception of software interfaces back to the developers.

“Coding, once considered an arcane craft practiced by solitary techies, is now recognized by educators and theorists as a crucial skill, even a new literacy, for all children” as stressed ‘connected code’ by Kafai and Burke (2014). They explore why children need to learn programming. "They argue that it is not simply enough for students to learn to code, but rather for all pupils to become computational participants in today's increasingly digital society. From this perspective, learning to program is to computational participation as writing is to literacy. Computational participation goes beyond programming to include collaboration in a maker society, just as literacy goes beyond the fundamental act of writing."" (Kafai & Burke, 2014)[1]. Kafai and Burke describe contemporary examples of computational participation: students who code not for the sake of coding but to create games, stories, and animations to share; the emergence of youth programming communities; the practices and ethical challenges of remixing (rather than starting from scratch); and the move beyond stationary screens to programmable toys, tools, and textiles."[2] . Kafai, a former PhD student of Papert is known for her work on computer game design as an environment for learning (1994), wherein she realized a constructionist approach to technology education, using games designed by 5 graders to introduce the issue of fractions to younger pupils, applying the idea of Papert to make children control the computer technology, rather than use fixed computer application and software to control the pupils. Kafai et al. (2007) also undertook research in the field of e-textile, comparing starter kits on the market. In the scientific paper entitled 'Fröbel's Forgotten Gift", textile construction kits are perceived as pathways into play, design and computation. "The paper explores a recent renaissance in commercially available textile construction kits for children. Through a survey of such kits, they argue that revisiting embroidery in this digital age is a powerful leverage to introduce computation into material culture. The Arduino LilyPad approach is highlighted being a shapable set of technology, bringing together crafting, design and technology, supporting individual projects, beyond a specific esthetic like the 'pink painted' application for girls.

To perceive the digital medium as an educational medium and as a subject matter to be investigated in education is not self-evident. Schelhowe (2008) stressed the point, that computer education in the context of computer science, ICT education or the "computer driving license" is mainly focusing on teaching something about the computer, in order to use the technology for a specific purpose. The latter is mirrored in the tool paradigm. However, education understood as the development of the personality of a subject and to position oneself in the world, is a complex process, moving beyond technocratic issues technology is linked to. As computer technology strongly coins and influenced the current media culture, a broader concept of education is to be developed.

In 'new creativity paradigms' Peppler (2014) introduces arts-based and interest driven learning, through the creative use of digital technology. Amongst others, smart textile and LilyPad Arduino-technology was explored by her for education. She explores different tools and media answering the question on "how young people creatively Use digital technologies?" In the book, the current tools, practices of media and digital arts are presented and discussed, including the

[1] (cited on: https://mitpress.mit.edu/connected-code)
[2] (cited on: https://mitpress.mit.edu/connected-code)

Do-It-Yourself movement, and the importance of ‘making -communities’ which can support interest-driven arts learning, inviting and sustaining participation in the arts for learning processes. Challenges and recommendations are presented: "This book explores research indicating that youth are learning new ways to engage in the arts on their own time and according to their own interests. Digital technologies, such as production tools and social media, allow youth to create and share their art. Kylie Peppler urges educators and policymakers to take advantage of arts learning opportunities and imagine a school setting where young people are driven by their own interests, using tablets, computers, and other devices to produce visual arts, music composition, dance, and design. This book gives educators an understanding of what is happening with current digital technologies and the opportunities that exist to connect to youth practice, and raises questions about why we don’t use these opportunities more frequently." (cited on**:** https://www.peterlang.com/view/product/30711?tab=reviews&format=HC) [16]

As mentioned earlier, the Arduino LilyPad technology was developed and introduced in education by Leah Buechley, MIT research group ‘High-Low Tech’. She was the first to introduce Smart Textile electronic crafting as an area for curriculum in higher school classroom. Also she developed technology including conducible ink in a pen, to enable children to sketch with electronics:

“So now that we developed these tools and found these materials that let us do these things, we started to realize that, essentially, anything that we can do with paper, anything that we can do with a piece of paper and a pen we can now do with electronics” (Buechley, s.a.).[3]

3.3 Recommendations for teachers

The school subject of computer science (as well as of STEM subjects in general) which grew in the tradition of engineering is often perceived to be too abstract, complicated and not attractive, lacking any ‘glam factor’. There was a similar perception related to programming to be noticed. However, the opportunities opening up through the introduction of physical computing and more design- and art education-based learning concepts are brought together in the Smart Textile/Lilypad Arduino technology applied in the framework of project-based learning. Pupils invent, develop, draw, connect, construct, program, control, debug and reprogram meaningful physical artefacts using their imagination, iconic interfaces in team based arrangements.

An introduction to Smart Textile with LilyPad-Arduino technology for teachers can be found online. The tutorial gives an overview over the components as well as a step-by-step introduction on how to set up and control the interactive systems. The free open source software ‘amici’ is available online as well: http://dimeb.informatik.uni-bremen.de/eduwear/about-2/

A more detailed tutorial for teachers interested in teaching coding can be accessed via the TACCLE3 Website www.taccle3.eu/

[3] (quoted on: https://www.ted.com/talks/leah_buechley_how_to_sketch_with_electronics/transcript?language=en#t-206180)

It is related to the classroom activities developed for TACCLE3 and available in German and english.

References

Bohne, R. (2012). *Making Things Wearable.* Intelligente Kleidung selber schneidern. O'Reilly Verlag.

Buechley, L. (s.a.). *Script of the TED talk: Hot to sketch with electronics.* [2016, September 21] https://www.ted.com/talks/leah_buechley_how_to_sketch_with_electronics/transcript?language=en#t-14430 Video ot the talk: https://www.ted.com/talks/leah_buechley_how_to_sketch_with_electronics?language=en

Druin, A. & Hendler, J. (2000). *Robots for kids: exploring new technologies for learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kafai, J. B. & Burke, Q. (2014). *Connected code. Why children need to learn programming*. Cambridge, Ma: The MIT press.

Kafai, J. B., Peppler, K. A., Burke, Q., Moore, M., & Glosson, D. (2007). Fröbel's Forgotten Gift. [2016, May 15] https://www.gse.upenn.edu/c4ls/sites/gse.upenn.edu.c4ls/files/pdfs/Paper_65_final.pdf

Kreye, A. (2014). *Die neue Weltsprache. Algorithmen bestimmen die Welt um uns herum. Höchste Zeit, dass wir uns mit ihnen beschäftigen.* in. Süddeutsche Zeitung No. 163, Friday, 18 July 2014, p.11

Papert, S. (1982). *Gedankenblitze (mindstorms).* Reinbek near Hamburg.

Peppler, K. (2014). *New creativity paradigms*. Cambridge, Ma: MIT.

Przybylla, M. & Romeike, R. (2013). Physical computing mit "My interactive garden". In N. Breier, P. Stechert, T. Wilke (Hrsg.). *INFOS 2013,15. GI-Fachtagung „Informatik und Schule", Praxisband, 3*. Department of Computer Science, Kiel University, 87-91. [2016, July 5] https://www.numerik.uni-kiel.de/~discopt/kcss/kcss_2013_03_v1.0_print.pdf

Reimann, D. (2006). *Ästhetisch-informatische Medienbildung mit Kindern und Jugendlichen*. Oberhausen.

Reimann, D. & Bekk, S. (2014). Künstlerisch geleitete Medienbildung mit Portfolios: Potenziale für Jugendliche in berufsvorbereitenden Bildungsmaßnahmen - Herausforderungen beim Übergang Schule – Beruf und das Konzept der berufsbiografischen Gestaltungskompetenz. In T. Hug, P. Missomelius, & W. Sützl (Eds.). *Medienimpulse, Beiträge zur Medienpädagogik.* Online-Zeitschrift, Themenschwerpunkt "Medienpädagogische Potenziale digitaler Medienkunst", Ausgabe 2/2014.

Reimann, D. & Bekk, S. (2015): Artistic approaches to media technology in interdisciplinary education – Innovative portfolios in digital media art projects. in: LARRY O'FARRELL, SHIFRA SCHONMANN, ERNST WAGNER (EdS., 2015), "The Wisdom of the Many - Key Issues in Arts Education, International Yearbook for Research in Arts Education, 2015, Münster: WAXMANN

Reimann, D., Bekk, S, Wüst, A., Schneider, C., Uller, C., & Walter, S. (2015). Handreichung „Smart Textile Interaktive Kleidung selbst gestalten"– ein Praxisleitfaden für die Arbeit mit Portfolios und Mentoring, mit Tutorial zu Arduino LilyPad/Software amici und Unterrichtsszenario. Entwickelt und publiziert im Rahmen des BMBF-Forschungsprojekts "MediaArt@Edu", KIT Karlsruhe. [2016, May 21] http://www.ibap.kit.edu/mediaartedu/downloads/Handreichung_MediaArt.pdf

Reimann, D. et al. (2003). Exploring the Computer as a Shapeable Medium by Designing Artefacts for Mixed Reality Environments in Interdisciplinary Education Processes, in: Proceedings of the ED-MEDIA, World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003, Honolulu, 2003, p. 915-923.

Rosales, A. (2012). Wearable music. Creating sound effects and music by playing, project presented at Ars Electronica 2012, u19 create your world village, festival catalogue.

Resnick, M. (s.a.). Webisite at MIT, Lifelong Kindergarten: http://web.media.mit.edu/~mres/;

https://llk.media.mit.edu/; https://www.media.mit.edu/research/groups/lifelong-kindergarten

Schelhowe, H. (2008). Digitale Medien als kulturelle Medien: Medien zum Be-Greifen wesentlicher Konzepte der Gegenwart. In J. Fromme, W. Sesink (Hrsg.). *Pädagogische Medientheorie*. Medienbildung und Gesellschaft Band 6, Wiesbaden.

Schön, S, Ebner, M., & Narr, K. (2016) (eds.). Making-Aktivitäten mit Kindern und Jugendlichen. Handbuch zum kreativen digitalen Gestalten, Norderstedt: books on demand, online PDF [2016, July 31] http://www.bimsev.de/n/userfiles/downloads/making_handbuch_online_final.pdf

Tan, X. L. (2005). *Storytelling wearables, an alternative biography*. www.xiaolitan.com/thesis/thesis.html [19.2.2010] and http://we-make-money-not-art.com/xiao_li_tans_st/ [12.04.2016]

Trappe, C. (2012). Creative access to technology: Building Sounding Artifact with children. In *Proceedings of IDC 2012*, Bremen, short paper.

Turkle, S. (2011). *Evocative objects - things we think with*. Cambridge Ma: MIT

4. Makey Makey practices

4.1 Introduction

Makey Makey was developed by MIT researchers Jay Silver and Eric Rosenbaum together with SparkFun Electronics. (Rogers, Paay, Brereton, Vaisutis, Marsden & Vetere, 2014). But what it is the Makey Makey? As Beginner's Mind Collective and Shaw (2012) have written, it is a platform for improvising tangible user interfaces. They clarify that "Makey Makey is a device that implements the Human Interface Device (HID) protocol, which allows it to send keyboard and mouse events to a computer without installing drivers or other software. The user connects everyday objects and natural materials to Makey Makey, in order to create a tangible user interface that controls any software running on a computer." According to Silver (2014) "Makey Makey catalyzes people to think of the world (the digital conductive or insulative on/off connectedness of objects) as a user interface to a modern computer".

As such, it is a device that promises to turn anything into a key or mouse button. The principle is clipping two objects to the Makey Makey board, which then makes a circuit, and Makey Makey sends the computer a keyboard message. It is easy to handle, compatible with all existing computer program that normally receives input from the keyboard or mouse, and it does not require the user to program or to assemble electronics. Due to its simplicity, people without technical skills can still use it – it is possible to construct a working interface with a physical input controlling a digital output in under a minute. (Beginner's Mind Collective & Shaw, 2012.)

This open-source hardware micro-controller built off of the Arduino platform. Each kit consists of Makey Makey board and two bundles of wires. The wires in the first bundle end in alligator clips, which attach to the top of the board, while the wires in the second bundle end in normal points that plug into the pins on the underside of the board. Sansing (2015, 12) explains, how it works: 'You connect the Makey Makey to your computer with a USB cord. Then you clip your alligator-pin wires to spots on the board labeled with input names—up, down, right, left, space, and click—that correspond to keys on your keyboard or buttons on your mouse. Then you clip at least one more wire to the "earth" bar at the bottom of the Makey Makey, and you're good to go. So long as you connect one of your input wires to your earth wire through any kind of conductive material, the Makey Makey will tell your computer to "press" that button. Connect the "space" wire to the "earth" wire, and your computer will "press" the space key.' In depth instructions of Makey Makey can be found from Sansing's (2015) article "Test-Driving the MaKey MaKey. The Arduino kit makes a game of learning. Banana pianos are just a start." Also, if you are interested, then you can read Silver`s paper (2014) which contains a good overview about Makey Makey process of inventing and there is also the story about inventing Rawdio Invention Kit.

4.2 Literature review and findings

Although the using Makey Makey is relatively easy, we found only a couple of articles which have studied its effect on learning. The first article by Lin and Chang (2014) gave an overview about using Makey Makey to increase training motivation of physically handicapped (Cerebral palsy) children. In the second article Rogers, Paay, Brereton, Vaisutis, Marsden and Vetere (2014) gave an overview of a study that focused on investigating how retirees learn, construct and play with the Makey Makey inventor's toolkit, and what this hands-on experience suggests to them about future technology uses and experiences. Following will be a brief overview of these studies.

In their study, Lin and Chang (2014) employed Flash- and Scratch-based multimedia by using a Makey Makey-based task system to increase the motivation level of children with cerebral palsy to perform physical activities. This study followed single-case design using ABAB models in which A indicated the baseline and B indicated the intervention. The experiment lasted one and a half months, the participants in this experiment were two kids: a 5-year and 9-month-old girl with severe physical disabilities with cerebral palsy convulsions, and a 3-year and 8-month-old boy with moderate multiple disabilities with both legs weak. Experiment was carried out as follows:

'Makey Makey was connected to a notebook computer on which the interactive Flash- and Scratch-base multimedia were installed with Microsoft Window 7. The Makey Makey connected with one participant and conductive materials. At the beginning, there is a prepared multimedia show on the screen for 5 s, then on the pause situation. When the participant touches the conductive material again, the multimedia plays for 5 s, after which the multimedia pauses. When the participant touches the conductive material, it creates a conductive loop, which then starts the play function. Because the participants are two children with physical disabilities, and they respond more slowly than normal children, the operating process time was set to 1 min. The intervention materials design by Flash and Scratch technology to set up the multimedia pauses automatically every 5 s, and so the multimedia could not play anything unless the participant touches the conductive material. The goal was to identify how many times the participant applied his/her physical strength to touch any part of the conductive material as a physical activity.' (Lin & Chang, 2014, 1964-1965.)

Experiment had 4 cycles: regular training cycle was changed to training with use of Makey Makey. During different sessions, the amount of correct movements per minute were counted. It was found, that both children made more correct movements with the use of Makey Makey technology than during normal training sessions. Authors concluded, that Makey Makey offers many opportunities to encourage development and increase motivation in children with special needs. They suggest that teachers, therapists and parents could use the ideas from the experiment and go even further to make regular exercises more interesting.

The fact that anyone, who is even remotely interested in Makey Makey, can manage using it is clearly illustrated in a study by Rogers et al. (2014) in which the participants were retirees aged over 60. Researchers ran a series of workshops where groups of two to three retirees used the Makey Makey inventor's toolkit. Workshops were usually two to three hours long and consisted of two parts: at first there was training, in which the groups were instructed to learn, assemble, create and play with the MaKey MaKey toolkit. This was followed by having an in-depth discussion over tea and cake. The researchers asked the participants to think about the future and suggest ideas for new technologies. This study showed, that retirees handled using Makey Makey very well, were interested in it and maid many suggestions for using it outside workshops. Although the retirees main focus was on how other people (for example their children or grandchildren) could use these tools, they enjoyed every activity included in the workshop and it was evident that mutual discovery and activities helped build their team spirit. As mentioned previously, the study by Rogers et al. (2014) supported the understanding that Makey Makey is suitable for use for a very large audience – it provides the joy of discovery to both, the young and old, independently from their technological prowess.

4.3 Recommendations for teachers

Despite the fact that it can take time to figure out how to use a Makey Makey in class activities, the rewards are definitely worth it. Studying will be more fun and interesting, so the school turns into a place where students and teachers alike want to come back to. If you want to know more about Makey Makey visit the webpage http://www.makeymakey.com, there are many good ideas, lesson plans and videos for you.

References

Beginner's Mind Collective & Shaw, D. (2012*). Makey Makey: Improvising Tangible and Nature-Based User Interfaces.* In TEI '12 Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, 367-370.

Hancock, C., Hichar, C., Holl-Jensen, C., Kraus, K., Mozafari, C., & Skutlin, K. (2013). Bibliocircuitry and the Design of the Alien Everyday. *Textual Cultures, 8*(1), 72-100.

https://scholarworks.iu.edu/journals/index.php/textual/article/view/5051/4649

Lin, C. Y. & Chang, Y. M. (2014). Increase in physical activities in kindergarten children with cerebral palsy by employing MaKey–MaKey-based task systems. *Research in Developmental Disabilities*, 35(9), 1963-1969.

Sansing, C. (2012). Test-Driving the MaKey MaKey. The Arduino kit makes a game of learning. Banana pianos are just a start. *School Library Journal*, april 2015, 12-13.

http://www.slj.com/2015/04/opinion/test-drive/slj-reviews-the-makey-makey-test-drive/

Silver, J. S. (2014). *Lens × Block. World as Construction Kit.* [PhD thesis]. Massachusetts Institute of Technology.

http://web.media.mit.edu/~silver/Lens-Block-Jay-Silver-PhD-Thesis.pdf

Rogers, Y., Paay, J., Brereton, M., Vaisutis, K., Marsden, G., & Vetere, F. (2014). Never Too Old: Engaging Retired People Inventing the Future with MaKey MaKey. *CHI 2014,* April 26 - May 01, Toronto, Canada.

5. Robotic practices

5.1 Introduction

New school curricula across Europe have a strong emphasis on computing skills of pupils with components that traditionally have been considered advanced – and often abstract and difficult to comprehend – subjects, such as computational and algorithmic thinking skills and programming. While this development is no doubt needed in the modern information society, the methods to deliver these skills to masses of pupils might be difficult.

One of the promising ways to deliver teaching in these contexts is use of *educational robotics*. Tangible devices make program execution explicit and concrete, and they often are considered to increase students' motivation towards the subject. In this chapter, we will review experiences found from literature about usage of educational robotics in different teaching contexts. We will also explore some of the common findings and problems when teaching with robotics.

5.2 Literature review

Educational expectations from technology have shifted from management and usability of learning materials towards motivational issues, like how to engage an individual student with a topic. Recently, the low-cost and highly accessible educational robot kits have gained popularity in tangible learning environments (Sklar et al., 2004). Technical subjects, such as electronic engineering (Newman et al., 2002) or programming, have been taught with educational robotics. Educational robotics has been used also to increase motivation of children in technology related school activities (del Solar and Avilés, 2004) or to promote STEM-based outreach programs (Cristoforis et al., 2013). Williams (2003) argues that Lego Mindstorms are qualitatively effective as educational technology for computer engineering in general, not only for robotics education.

However, successful use of educational robotics requires new kinds of classroom settings and teachers have to change their teaching methods according to the needs of the new environment. Otherwise, the use of educational robotics may lead to negative results, as reported by Fagin and Merkle (2002). Correll et al. (2013) propose solutions to the recognised issues related to deployment of large-scale robotics classes, such as scalability of a learning environment, or complexity of robot constructions that may prevent learning of the subject.

As indicated above, educational robotics has long been applied in teaching and learning especially science, technology, engineering, and mathematics (STEM). Reports found from literature, such as Petre and Price (2004), emphasize motivational factors and social context that educational robotics can bring to school classes and affect not only to technically oriented children. According to Petre and Price (2004), the primary motivational factor driving robotics projects is the desire to build a better robot. The study was conducted in the context of RoboCup Junior and RoboFesta robotics competitions, and the second common motivational reason was the reward that would be available from the competition. In general, competitions and tournaments as a learning context for educational robotics have been very popular (see for example Johnson, 2003; Sklar & Eguchi, 2004; Martins et al., 2015). Cooperative and collaborative learning are important factors of a successful educational robotics project (Denis & Hubert, 2001).

Role of educational robotics has been seen as a tool to foster not only STEM skills but also skills such as collaboration, imagination, and self-expression (Alimisis, 2013). These aims bind elements of robotics to the development of school curricula that define the essential 21st century learning skills. It has been recognized for a long time that there is a need to find new and novel ways to integrate robotics into school curricula (Johnson, 2003), but this development has taken place just during last few years together with introduction of revised curricula across the world where computational thinking and programming are vital elements. Nature of robotics projects fits very well in problem-based learning to teach non-subject specific 21st century skills as proposed by Arlegui et al. (2013). On the other hand, educational robotics fits very well to support especially science and technology courses, and also in these subject-specific contexts students' active learning attitude and project-based working methods are essential (Karahoca et al., 2011).

Together this development, focus in educational robotics is shifting from competitions towards more interdisciplinary projects where traditional STEM thinking is supplied with arts (STEAM) to attract and foster motivation of wider student groups than before, for example the Arts & Bots program (Cross et al, 2015) that was originally developed as extracurricular activity, and transformed since that into in-school intervention. Cross et al. (2015) conclude that the Arts & Bots program increase students' technical knowledge and skills, motive students and make them confident with technology, and engage a broad participant group across gender and previous technological exposure. These factors in a form or another can be found across the literature virtually in any report about educational robotics experiments or projects. For example, Kaloti-Hallak et al. (2015) present a study that shows students' high motivation towards robotics projects in STEM education.

Research in educational robotics settings is as varied as use contexts and scenarios. A recent review study by Toh et al. (2016) identified a large selection of research methods in educational robotics settings but they concluded also that experimental methods are lacking and appropriate quantitative analysis is needed. This development would also help to justify and strengthen position of educational robotics as a versatile yet strong tool in many levels and fields of education.

5.3 Key findings

Computing education is a relatively young field that designs, implements and evaluates education—curricula, pedagogy, methods, materials, tools—in the five interrelated fields of Computer Science, Computer Engineering, Software Engineering, Information Systems and

Information Technology. A key question of the computing education research (CER) community is the pedagogy that should be adopted when teaching computing skills for all students (Falkner et al., 2014). One of the rising pedagogical issues is contextualization of teaching computing. The computing educators are more and more demanded to bring real-life context into their teaching (Rick et al., 2012).

Based on the literature reviewed, we argue that educational robotics is a very fit tool for contextualizing computing education (computational thinking, programming skills). Nature of educational robotics drives students to work actively and a robotics learning environment has an emphasis on project-based learning. Furthermore, educational robotics creates a natural framework for multidisciplinary projects where technology and ICT meet art and humanities. Successful utilization of educational robotics in a classroom, however, requires attitude to change teaching methods and learning environment in general. The environment should allow exploratory learning and support students taking different paths in their problem solving processes. Furthermore, teachers or instructors should have necessary skillsets to overcome technical and conceptual problems related to hardware in use. Also, even if the computational thinking is language or platform independent skill, teachers should familiarize themselves with the programming environment and other tools they're going to expose in a classroom. This obviously requires active participation of teachers on acquiring necessary supplementary education. The most interesting robotics projects are results of creative-minded projects that go beyond normal subject boundaries combining several of the 21st century skills.

References

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. *Themes in Science & Technology Education*, *6*(1), 63-71.

Arlegui, J., Pina, A., & Moro, M. (2013). A PBL approach using virtual and real robots (with BYOB and LEGO NXT) to teaching learning key competences and standard curricula in primary level. *Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality (TEEM '13)*, 323-328.

Correll, N., Wing, R., & Coleman, D. (2013). A One-Year Introductory Robotics Curriculum for Computer Science Upperclassmen. *IEEE Transactions on Education 56*, 54–60.

de Cristoforis, P., Pedre, S., Nitsche, M., Fischer, T., Pessacg, F., & Di Pietro, C. A Behavior-Based Approach for Educational Robotics Activities. *IEEE Transactions on Education 56*, 61–66.

Cross, J. L., Hamner, E., Bartley, C., & Nourbakhsh, I. (2015). Arts & Bots: Application and Outcomes of a Secondary School Robotics Program. *Frontiers in Education Conference (FIE)*, 1-9.

Denis, B. & Hubert, S. (2001). Collaborative learning in an educational robotics environment. *Computers in Human Behavior*, *17*(5–6), 465-480.

Fagin, B. S. & Merkle, L. (2002). Quantitative analysis of the effects of robots on introductory Computer Science education. *Journal on Educational Resources in Computing 2*, 1–17.

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: challenge and opportunity. In J. Whalley & D. D'Souza (Eds.). *Proceedings of the Sixteenth Australasian Computing Education Conference*. Volume 148 (ACE '14*)*, Australian Computer Society, Inc., Darlinghurst, Australia, 3-12.

Johnson, J. (2003). Children, Robotics, and Education. *7th Symposium of Artificial Life and Robotics,* 16-21.

Kaloti-Hallak, F., Armoni, M., & Ben-Ari, M. (2015). Students' Attitudes and Motivation During Robotics Activities. *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE)*, 102-110.

Karahoca, D., Karahoca, A., & Uzunboylub, H. (2011). Robotics teaching in primary school education by project based learning for supporting science and technology courses. *Procedia Computer Science 3*, 1425-1431.

Martins, F. N., Gomes, I. S., & Santos, C. R. F. (2015). Junior Soccer Simulation: Providing all Primary and Secondary Students Access to Educational Robotics. *12th Latin American Robotics Symposium and 3rd Brazilian Symposium on Robotics (LARS-SBR),* 61-66.

Newman, K., Hamblen, J., & Hall, T. (2002). An introductory digital design course using a low-cost autonomous robot. *IEEE Transactions on Education 45*, 289–296.

Petre, M. & Price, B. (2004). Using Robotics to Motivate 'Back Door' Learning. *Education and Information Technologies 9* (2), 147-158.

Rick, D., Morisse, M., & Schirmer, I. (2012). Bringing contexts into the classroom: a design-based approach. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education,* 105-115.

Sklar, E. & Eguchi, A. (2004). RoboCupJunior – Four Years Later. *Proceedings of the Eighth International RoboCup Symposium.*

Sklar, E., Parsons, S., & Stone, P. (2004). Using RoboCup in university-level computer science education. *Journal on Educational Resources in Computing 4*, 1–21.

del Solar, J. R. & Avile´s, R. (2004). Robotics courses for children as a motivation tool: the Chilean experience. *IEEE Transactions on Education 47*, 474–480.

Toh, L. P. E., Causo, A., Tzuo, P. W., Chen, I. M., & Yeo, S. H. (2016). A Review on the Use of Robots in Education and Young Children. *Educational Technology & Society 19*(2), 148–163.

Williams, A. (2003). The qualitative impact of using LEGO MINDSTORMS robots to teach computer engineering. *IEEE Transactions on Education*, *46*, 206.

Conclusion

We live in a time of technology, which means that technology surrounds us daily and most of us can’t live a day without using a computer, mobile phone, tablet or some other “smart” device. It means that digital proficiency and computational thinking become vital for our daily lives. The need for professionals, who in addition to the knowledge how to use a “smart” device know the inner workings of it as well and can develop it further, is growing. That is why we need to update our goals and methods in education, so that the next generation of children wouldn’t be passive users, but would become active developers and creators of future.

Jonassen, Howland, Marra & Crismond (2008) have written that ’If schools are to foster meaningful learning, then the ways that we use technologies in schools must change from technology-as-teacher to technology-as-partner in the learning process.’ They argue that ’technologies should be used as engagers and facilitators of thinking’.

In this literature review we tried to show, that developing computational thinking is not a ‘rocket-science’ and that there are many possibilities to teach computational thinking for children from their early age. We hope that we managed to dispel the myths like ‘technical engineering and computer science are suitable only for boys’ and ‘it is not possible to teach computational thinking without computers or special equipment’. We also hope that we inspired teachers to thoroughly examine Makey Makey, smart textiles, educational robotics and the other tools and environments that help them to bring teaching in their classroom into line with the requirements of the 21st century.

If you need more information about supporting primary teachers to teach coding, visit the Web site http://www.taccle3.eu/.

References

Jonassen, D., Howland, J., Marra, R. M., & Crismond, D. (2008). Meaningful learning with technology (3rd ed.). Upper Saddle River, New Jersey: Pearson Education. [2015, mai 03]. http://www.education.com/pdf/how-does-technology-facilitate-learning/

References

Ableson, H. (2012). From Computational Thinking to Computational Values. *Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12* (pp. 239). New York, NY, USA: ACM.

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Early, J. P. (2010). Computational thinking for the sciences: A three day workshop for high school science teachers. *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 42-46). Milwaukee, Wisconsin, USA: ACM.

Aho, A. V. (2012). Computation and Computational Thinking. *Computer Journal, 55*(7), 832-835. doi:10.1093/comjnl/bxs074

Aiken, J. M., Caballero, M. D., Douglas, S. S., Burk, J. B., Scanlon, E. M., Thoms, B. D., & Schatz, M. F. (2013). Understanding Student Computational Thinking with Computational Modeling. In P. V. Engelhardt, A. D. Churukian, & N. S. Rebello (Eds.), *2012 Physics Education Research Conference* (Vol. 1513, pp. 46-49).

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. *Themes in Science & Technology Education, 6*(1), 63-71.

Allan, V., Barr, V., Brylow, D., & Hambrusch, S. (2010). Computational thinking in high school courses *Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE'10* (pp. 390-391). New York, NY, USA: ACM.

Alonso de Castro, M. G. (2014). Educational projects based on mobile learning. *Education in the Knowledge Society, 15*(1), 10-19.

Ambrosio, A. P., Xavier, C., & Georges, F. (2014). Digital ink for cognitive assessment of computational thinking *Proceedings of the 2014 IEEE Frontiers in Education Conference (FIE) (Madrid, Spain, 22-25 Oct. 2014)* (pp. 1-7). EEUU: IEEE.

American Psychological Association (2014, August). *Think Again: Men and Women Share Cognitive Skills.* [2016, August 13] http://www.apa.org/action/resources/research-in-action/share.aspx

Ananiadou, K., & Claro, M. (2009). 21st Century skills and competences for new millennium learners in OECD Countries. *OECD Education Working Papers, 41*.

Arlegui, J., Pina, A., & Moro, M. (2013). A PBL approach using virtual and real robots (with BYOB and LEGO NXT) to teaching learning key competences and standard curricula in primary level. *Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality (TEEM '13)*, 323-328.

Astrachan, O., Hambrusch, S., Peckham, J., & Settle, A. (2009). The present and future of computational thinking. *SIGCSE Bulletin Inroads, 41*(1), 549-550. doi:10.1145/1539024.1509053

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661-670. doi: 10.1016/j.robot.2015.10.008

Balanskat, A., & Engelhardt, K. (2015). *Computing our future. Computer programming and coding Priorities, school curricula and initiatives across Europe*. [2016, August 5] http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf/746e36b1-e1a6-4bf1-8105-ea27c0d2bbe0

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology, 38*(6), 20-23.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and What is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54. doi:10.1145/1929887.1929905

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using scalable game design to teach computer science from middle school to graduate school. *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education, ITiCSE '10* (pp. 224-228). New York, USA: ACM.

Basawapatna, A. R., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 245-250). New York, USA: ACM.

Basogain, X., Olabe, M. A., C., O. J., Ramírez, R., & García, J. (2016). PC-01: Introduction to Computational Thinking. Educational Technology in Primary and Secondary Education. In F. J. García-Peñalvo & J. A. Mendes (Eds.), *XVIII Simposio Internacional de Informática Educativa, SIIE 2016* (pp. 191-195). Salamanca, España: Ediciones Universidad de Salamanca.

Basu, S., Kinnebrew, J. S., & Biswas, G. (2014). Assessing Student Performance in a Computational-Thinking Based Science Learning Environment. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Intelligent Tutoring Systems, Its 2014* (Vol. 8474, pp. 476-481).

Bean, N., Weese, J., Feldhausen, R., & Bell, R. S. (2015). Starting from Scratch Developing a Pre-Service Teacher Training Program in Computational Thinking. *2015 IEEE Frontiers in Education Conference (FIE), October 21-24, 2015, Camino Real El Paso, El Paso, TX, USA* (pp. 1307-1314). EEUU: IEEE.

Beginner's Mind Collective & Shaw, D. (2012). *Makey Makey: Improvising Tangible and Nature-Based User Interfaces*. In TEI '12 Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, 367-370.

Bell, T., Witten, I. H., & Fellows, M. (2016). *CS Unplugged. An enrichment and extension programme for primary-aged students. Version 3.2.2*. New Zealand: University of Canterbury. CS Education Research Group.

Bennett, V., Koh, K., & Repenning, A. (2011). Computing learning acquisition? *Proceedings of the 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 18-22 Sept. 2011, Pittsburgh, PA, USA* (pp. 243-244). EEUU: IEEE.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education, 72*, 145-157. doi: 10.1016/j.compedu.2013.10.020

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). Defining twenty-first century skills. In P. Griffin, B. McGaw, & E. Care (Eds.), *Assessment and teaching of 21st century skills* (pp. 17–66). Netherlands: Springer.

Blum, L., & Cortina, T. J. (2007). CS4HS: an outreach program for high school CS teachers. *SIGCSE Bull., 39*(1), 19-23. doi:10.1145/1227504.1227320

Bohne, R. (2012). *Making Things Wearable*. Intelligente Kleidung selber schneidern. O'Reilly Verlag.

Bono, E. D. (1968). *New think: The use of lateral thinking in the generation of new ideas*. New York, NY: Basic Books.

Bono, E. D. (1970). *Lateral Thinking. A Textbook of Creativity*. London: Ward Lock Educational.

Bort, H., & Brylow, D. (2013). CS4Impact: measuring computational thinking concepts present in CS4HS participant lesson plans *Proceeding of the 44th ACM technical symposium on Computer Science Education, SIGCSE '13, March 6-9, 2013, Denver, Colorado, USA* (pp. 427-432). New York, NY, USA: ACM.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Annual American Educational Research Association meeting, Vancouver, BC, Canada.

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing computer science back into schools: Lessons from the UK *Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13* (pp. 269-274). New York, USA: ACM.

Buechley, L. (s.a.). *Script of the TED talk: Hot to sketch with electronics.* [2016, September 21] https://www.ted.com/talks/leah_buechley_how_to_sketch_with_electronics/transcript?lang uage=en#t-14430 Video ot the talk: https://www.ted.com/talks/leah_buechley_how_to_sketch_with_electronics?language=en

Buffum, P. S., Lobene, E. V., Frankosky, M. H., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2015). A Practical Guide to Developing and Validating Computer Science Knowledge Assessments with Application to Middle School *Proceedings of the 46th ACM Technical Symposium on Computer Science Education. SIGCSE '15 (Kansas City, Missouri, USA, March 4th-7th)* (pp. 622-627). New York, NY, USA: ACM.

Burke, Q. (2012). The markings of a new pencil: Introducing programming-aswriting in the middle school classroom. *Journal of Media Literacy Education*, *4*(2), 121–135.

CAS Barefoot. (2014). Computational Thinking. [2016, July 18] http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/

Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education*, *22*(4), 711–722.

Clarke, T. (2015, March 10). Women and science: Time to cut the Neurotrash. *50.50 inclusive democracy*. [2016, September 5] https://www.opendemocracy.net/5050/tara-clarke/women-and-science-time-to-cut-neurotrash-0

CODDII, & AENUI. (2014). Por la inclusión de asignaturas específicas de ciencia y tecnología informática en los estudios básicos de la enseñanza secundaria y bachillerato. *ReVisión*, *7*(2), 5-7.

Computer Science Teachers Association and International Society of Technology in Education. (2011). *Operational definition of computational thinking for K-12 education*. [2016, October 1] http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

Correll, N., Wing, R., & Coleman, D. (2013). A One-Year Introductory Robotics Curriculum for Computer Science Upperclassmen. *IEEE Transactions on Education 56*, 54–60.

Cross, J. L., Hamner, E., Bartley, C., & Nourbakhsh, I. (2015). Arts & Bots: Application and Outcomes of a Secondary School Robotics Program. *Frontiers in Education Conference (FIE)*, 1-9.

Crow, D. (2014). Why every child should learn to code. *The Guardian*. [2016, September 7] https://www.theguardian.com/technology/2014/feb/07/year-of-code-dan-crow-songkick

CSTA, & ISTE. (2011). *Computational Thinking in K–12 Education leadership toolkit*. [2016, August 7] https://csta.acm.org/Curriculum/sub/CurrFiles/471.11CTLeadershiptToolkit-SP-vF.pdf de Cristoforis, P., Pedre, S., Nitsche, M., Fischer, T., Pessacg, F., & Di Pietro, C. A Behavior-Based Approach for Educational Robotics Activities. *IEEE Transactions on Education 56*, 61–66.

del Solar, J. R. & Avile´s, R. (2004). Robotics courses for children as a motivation tool: the Chilean experience. *IEEE Transactions on Education 47*, 474–480.

Denis, B. & Hubert, S. (2001). Collaborative learning in an educational robotics environment. *Computers in Human Behavior*, *17*(5–6), 465-480.

Department for Education. (2013). *National curriculum in England: computing programmes of study - key stages 1 and 2*. (DFE-00171-2013). UK: UK Government Retrieved from https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf.

diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge: MIT Press.

Do Internalized Feminine Norms Depress Girls' STEM Attitudes & Participation? (s.a.) [2016, September 24] http://www.truechild.org/Images/Interior/learnthefacts/__femininity%20&%20stem.pdf

Druin, A. & Hendler, J. (2000). *Robots for kids: exploring new technologies for learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Espeso, P. (2015, December 14, 2015). *Cómo iniciar a un niño en la programación desde cero.* [2016, October 2] http://www.xataka.com/otros/como-iniciar-a-un-nino-en-la-programacion-desde-cero

Fagin, B. S. & Merkle, L. (2002). Quantitative analysis of the effects of robots on introductory Computer Science education. *Journal on Educational Resources in Computing 2*, 1–17.

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: challenge and opportunity. In J. Whalley & D. D'Souza (Eds.). *Proceedings of the Sixteenth Australasian Computing Education Conference*. Volume 148 (ACE '14*)*, Australian Computer Society, Inc., Darlinghurst, Australia, 3-12.

Felleisen, M., & Krishnamurthi, S. (2009). Why computer science doesn't matter. *Communications of the ACM, 52*(7), 37-40. doi:10.1145/1538788.1538803

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97. doi:http://dx.doi.org/10.1016/j.compedu.2012.11.016

Franklin, D., Hill, C., Dwyer, H., Iveland, A., Killian, A., & Harlow, D. (2015). Getting Started in Teaching and Researching Computer Science in the Elementary Classroom *Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE '15, March 4-7, 2015, Kansas City, Missouri* (pp. 552-557). New York, NY, USA: ACM.

Frayer, J. M. (2016, September 21). *China Pushes Coding for Kids in Effort to Tackle Innovation Gap. NBC News*. [2016, October 2] http://www.nbcnews.com/news/china/china-pushes-coding-kids-effort-tackle-innovation-gap-n641966

Fundación Créate website [2016, October 19]. http://www.fundacioncreate.org/

García-Peñalvo, F. J. (2016a). Proyecto TACCLE3 – Coding. In F. J. García-Peñalvo & J. A. Mendes (Eds.), *XVIII Simposio Internacional de Informática Educativa, SIIE 2016* (pp. 187-189). Salamanca, España: Ediciones Universidad de Salamanca.

García-Peñalvo, F. J. (2016b). What Computational Thinking Is. *Journal of Information Technology Research, 9*(3), v-viii.

Gelman, R., & Brenneman, K. (2004). Science learning pathways for young children. *Early Childhood Research Quarterly, 19*(1), 150–158.

Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities: An evaluation of the educational game light-bot *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '13* (pp. 10-15). New York, NY, USA: ACM.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. doi:10.3102/0013189X12463051

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors *Proceedings of the 40th ACM technical symposium on Computer Science Education, SIGCSE '09, March 4-7, 2009, Chattanooga, TN USA* (pp. 183-187). New York, NY, USA: ACM.

Hamer, M., Stamatakis, E., & Mishra, G. (2009). Psychological Distress, Television Viewing, and Physical Activity in Children Aged 4 to 12 Years. *Pediatrics, 123* (5), 1263-1268.

Hancock, C., Hichar, C., Holl-Jensen, C., Kraus, K., Mozafari, C., & Skutlin, K. (2013). Bibliocircuitry and the Design of the Alien Everyday. *Textual Cultures, 8*(1), 72-100. https://scholarworks.iu.edu/journals/index.php/textual/article/view/5051/4649

Hemmendinger, D. (2010). A plea for modesty. *ACM Inroads, 1*(2), 4-7. doi:10.1145/1805724.1805725

Hinkley, T., Verbestel, V., Ahrens, W., Lissner, L., Molnár, D., Moreno, L. A., Pigeot, I., Pohlabeln, H., Reisch, L. A., Russo, P., Veidebaum, T., Tornaritis, M., Williams, G., De Henauw, S., & De Bourdeaudhuij, I. (2014). Early childhood electronic media use as a predictor of poorer well-being: a prospective cohort study. *JAMA Pediatrics, 168* (5), 485-492.

Hockly, N. (2012). Digital literacies. *ELT Journal, 66*(1), 108-112. doi:10.1093/elt/ccr077

Howland, J., Jonassen, D. H., & Marra, R. M. (2011). *Meaningful learning with technology* (4th ed.). Upper Saddle River, NJ, USA: Pearson.

https://llk.media.mit.edu/; https://www.media.mit.edu/research/groups/lifelong-kindergarten

Isbell, C., Stein, A., Cutler, R., Forber, J., Fraser, L., Impagliazzo, J., . . . Y., X. (2009). (Re)Defining computing curricula by (re)defining computing. *ACM SIGCSE Bulletin, 41*(4), 195-207. doi:10.1145/1709424.1709462

Jacobson, L. (2016, April 5). Never Too Young To Code. *School Library Journal* [2016, October 3] http://www.slj.com/2016/04/technology/never-too-young-to-code/#_

Johnson, J. (2003). Children, Robotics, and Education. *7th Symposium of Artificial Life and Robotics*, 16-21.

Jonassen, D., Howland, J., Marra, R. M., & Crismond, D. (2008). Meaningful learning with technology (3rd ed.). Upper Saddle River, New Jersey: Pearson Education. [2015, mai 03]. http://www.education.com/pdf/how-does-technology-facilitate-learning/

Kafai, J. B. & Burke, Q. (2014). *Connected code. Why children need to learn programming*. Cambridge, Ma: The MIT press.

Kafai, J. B., Peppler, K. A., Burke, Q., Moore, M., & Glosson, D. (2007). Fröbel's Forgotten Gift. [2016, May 15] https://www.gse.upenn.edu/c4ls/sites/gse.upenn.edu.c4ls/files/pdfs/Paper_65_final.pdf

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing*, *4*(3), 583-596.

Kaloti-Hallak, F., Armoni, M., & Ben-Ari, M. (2015). Students' Attitudes and Motivation During Robotics Activities. *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE)*, 102-110.

Karahoca, D., Karahoca, A., & Uzunboylub, H. (2011). Robotics teaching in primary school education by project based learning for supporting science and technology courses. *Procedia Computer Science 3*, 1425-1431.

Kazakoff, E., & Bers, M. (2012). Programming in a robotics context in the kindergarten classroom: the impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia*, *21*(4), 371–391.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 83-137. doi:10.1145/1089733.1089734

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning *2010 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2010 (Leganés, Madrid, Spain, 21-25 Sept. 2010)* (pp. 59 - 66). USA: IEEE.

Kreye, A. (2014). *Die neue Weltsprache. Algorithmen bestimmen die Welt um uns herum. Höchste Zeit, dass wir uns mit ihnen beschäftigen.* in. Süddeutsche Zeitung No. 163, Friday, 18 July 2014, p.11

Lee, Y.-J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia, 19*(3), 307–326.

Lieberman, J. N. (1965). Playfulness and Divergent Thinking: An Investigation of their Relationship at the Kindergarten Level. *The Journal of Genetic Psychology: Research and Theory on Human Development, 107*(2), 219-224. doi:10.1080/00221325.1965.10533661

Lin, C. Y. & Chang, Y. M. (2014). Increase in physical activities in kindergarten children with cerebral palsy by employing MaKey–MaKey-based task systems. *Research in Developmental Disabilities,* 35(9), 1963-1969.

Lin, J. M., & Liu, S. F. (2012). An investigation into parent-child collaboration in learning computer programming. *Educational Technology and Society, 15*(1), 162-173.

Liu, J., Lin, C.-H., Hasson, E. P., & Barnett, Z. D. (2011). Introducing computer science to K-12 through a summer computing workshop for teachers *Proceedings of the 42nd ACM technical symposium on Computer Science Education, SIGCSE '11, March 9-12, 2011, Dallas, TX, USA* (pp. 389-394). New York, NY, USA: ACM.

Llorens-Largo, F. (2015). Dicen por ahí… que la nueva alfabetización pasa por la programación. *ReVisión, 8*(2), 11-14.

Long, X., Zhang, J., & Li, Z. (2013). The Design of Teaching Structure based on Competency Training of Computational Thinking *Proceedings of the International Conference on Education Technology and Information System (ICETIS 2013)* (pp. 421-425). Amsterdam, The Netherlands: Atlantis Press.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41,* 51-61. doi: 10.1016/j.chb.2014.09.012

Maiorana, F., Giordano, D., & Morelli, R. (2015). Quizly: A live coding assessment platform for App Inventor *Proceedings of the 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (pp. 25-30). USA: IEEE.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference* (pp. 1-29). New York, USA: ACM.

Manovich, L. (2013). *Software Takes Command*. New York, USA: Bloomsbury.

Martín, J. (2016, May 20, 2016). Por qué debemos enseñar a los niños a programar. [2016, October 2] http://futurizable.com/debemos-ensenar-los-ninos-programar

Martins, F. N., Gomes, I. S., & Santos, C. R. F. (2015). Junior Soccer Simulation: Providing all Primary and Secondary Students Access to Educational Robotics. *12th Latin American Robotics Symposium and 3rd Brazilian Symposium on Robotics (LARS-SBR)*, 61-66.

McGrew, K. S. (2009). CHC theory and the human cognitive abilities project: Standing on the shoulders of the giants of psychometric intelligence research. *Intelligence, 37*(1), 1-10. doi:http://dx.doi.org/10.1016/j.intell.2008.08.004

Miller, P. (2009). Learning with a missing sense: What can we learn from the interaction of a deaf child with a turtle? *American Annals of the Deaf, 154*(1), 71-82.

Mills, K. A. (2010). A review of the ‘‘digital turn’’ in the new literacy studies. *Review of Educational Research, 80*(2), 246–271. doi:10.3102/0034654310364401

Mims, C. (2012, February 26). How Young Is Too Young to Learn to Code? *MIT Technology Review*. [2016, October 3] https://www.technologyreview.com/s/427064/how-young-is-too-young-to-learn-to-code/

Moreno-León, J., & Robles, G. (2015a). *Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills*. Paper presented at the Scratch Conference, Amsterdam, The Netherlands. http://jemole.me/replication/2015scratch/InferCT.pdf

Moreno-León, J., & Robles, G. (2015b). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects *Proceedings of the Workshop in Primary and Secondary Computing Education, WiPSCE '15 (London, United Kingdom, November 9-11, 2015)* (pp. 132-133). New York, NY, USA: ACM.

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED, Revista de Educación a distancia, 46*.

Moreno-León, J., Robles, G., & Román-González, M. (2016). Comparing Computational Thinking Development Assessment Scores with Software Complexity Metrics *Proceedings of the IEEE Global Engineering Education Conference, EDUCON 2016, Abu Dahbi, United Arab Emirates, 11-13 April, 2016*. EEUU: IEEE.

National Research Council Committee on Information Technology Literacy. (1999). *Being Fluent with Information Technology*. Washington, DC: National Academy Press.

Newman, K., Hamblen, J., & Hall, T. (2002). An introductory digital design course using a low-cost autonomous robot. *IEEE Transactions on Education 45*, 289–296.

Ng, W. (2012). Can we teach digital natives digital literacy? *Computers & Education, 59*(3), 1065–1078. doi: 10.1016/j.compedu.2012.04.016

Page, A. S., Cooper, A. R., Griew, P., & Jago, R. (2010). Children's screen viewing is related to psychological difficulties irrespective of physical activity. *Pediatrics, 126* (5), e1011-e1017.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY, USA: Basic Books.

Papert, S. (1982). *Gedankenblitze (mindstorms).* Reinbek near Hamburg.

Papert, S. (1982). *Mindstorms Gedankenblitze*. Reinbek bei Hamburg.

Park, Y. (2016, 13 June 2016). 8 digital skills we must teach our children. [2016, September 7] https://www.weforum.org/agenda/2016/06/8-digital-skills-we-must-teach-our-children/

Patten, J., Griffith, L., & Ishii, H. (2000). A tangible Interface for controlling robotic toys. *Proceedings of CHI'00*, April 1-6, 2000, The Hague, NL.

Peez, G. (2014). Mit Fingerspitzengefühl zu Erfahrung und Wissen. Kasuistische Grundlagenforschung zur sensomotorischen Bedienung von Multi-Touchscreens. In. merz medien + erziehung. *Zeitschrift für Medienpädagogik*, *2*, 67-73. [2016, September 23] http://www.georgpeez.de/texte/merz14.htm

Peppler, K. (2014). *New creativity paradigms*. Cambridge, Ma: MIT.

Peppler, K. A., & Kafai, Y. B. (2007). Collaboration, computation, and creativity: media arts practices in urban youth culture *Proceedings of the 8th international conference on Computer supported collaborative learning* (pp. 590-592). New Brunswick, New Jersey, USA: International Society of the Learning Sciences.

Petre, M. & Price, B. (2004). Using Robotics to Motivate 'Back Door' Learning. *Education and Information Technologies 9* (2), 147-158.

Piaget, J. (1954). *The construction of reality in the child*. New York, USA: Basic Books.

Pokress, S. C., & Domínguez Veiga, J. J. (2013). MIT App Inventor. Enabling personal mobile computing *Proceedings of Programming for Mobile and Touch, PRoMoTo 2013*.

Pólya, G. (1957). *How to Solve It. A new aspect of mathematical method* (2nd ed.). Priceton, New Jersey: Princeton University Press.

Prensky, M. (2008). Programming Is the New Literacy. Retrieved from http://www.edutopia.org/literacy-computer-programming

Przybylla, M. & Romeike, R. (2013). Physical computing mit "My interactive garden". In N. Breier, P. Stechert, T. Wilke (Hrsg.). *INFOS 2013,15. GI-Fachtagung „Informatik und Schule", Praxisband*, *3*. Department of Computer Science, Kiel University, 87-91. [2016, July 5] https://www.numerik.uni-kiel.de/~discopt/kcss/kcss_2013_03_v1.0_print.pdf

QS Staff Writer (2012, June 27). *The STEM Gender Gap in Universities*. [2016, August 13] http://www.topuniversities.com/courses/engineering/stem-gender-gap-universities

Reimann, D. (2006). *Ästhetisch-informatische Medienbildung mit Kindern und Jugendlichen*. Oberhausen.

Reimann, D. & Bekk, S. (2014). Künstlerisch geleitete Medienbildung mit Portfolios: Potenziale für Jugendliche in berufsvorbereitenden Bildungsmaßnahmen - Herausforderungen beim Übergang Schule – Beruf und das Konzept der berufsbiografischen Gestaltungskompetenz. In T. Hug, P. Missomelius, & W. Sützl (Eds.). *Medienimpulse, Beiträge zur Medienpädagogik.* Online-Zeitschrift, Themenschwerpunkt "Medienpädagogische Potenziale digitaler Medienkunst", Ausgabe 2/2014.

Reimann, D. & Bekk, S. (2015): Artistic approaches to media technology in interdisciplinary education – Innovative portfolios in digital media art projects. in: LARRY O'FARRELL,

Reimann, D. et al. (2003). Exploring the Computer as a Shapeable Medium by Designing Artefacts for Mixed Reality Environments in Interdisciplinary Education Processes, in: Proceedings of the ED-MEDIA, World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003, Honolulu, 2003, p. 915-923.

Reimann, D., Bekk, S, Wüst, A., Schneider, C., Uller, C., & Walter, S. (2015). Handreichung „Smart Textile Interaktive Kleidung selbst gestalten“– ein Praxisleitfaden für die Arbeit mit Portfolios und Mentoring, mit Tutorial zu Arduino LilyPad/Software amici und Unterrichtsszenario. Entwickelt und publiziert im Rahmen des BMBF-Forschungsprojekts "MediaArt@Edu", KIT Karlsruhe. [2016, May 21] http://www.ibap.kit.edu/mediaartedu/downloads/Handreichung_MediaArt.pdf

Repenning, A. (2006). Excuse me, I need better AI!: employing collaborative diffusion to make game AI child's play *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (pp. 169-178). Boston, Massachusetts: ACM.

Resnick, M. (2013). *Learn to Code, Code to Learn*. [2016, October 4] https://cedsurge.herokuapp.com/news/2013-05-08-learn-to-code-code-to-learn

Resnick, M. (s.a.). Webisite at MIT, Lifelong Kindergarten: http://web.media.mit.edu/~mres/;

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: Programming for all. *Commuication of the ACM, 52*(11), 60-67. doi:10.1145/1592761.1592779

Rick, D., Morisse, M., & Schirmer, I. (2012). Bringing contexts into the classroom: a design-based approach. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, 105-115.

Riley, D. D., & Hunt, K. A. (2014). *Computational thinking for the modern problem Solver*. Boca Raton, FL, USA: CRC Press.

Rogers, Y., Paay, J., Brereton, M., Vaisutis, K., Marsden, G., & Vetere, F. (2014). Never Too Old: Engaging Retired People Inventing the Future with MaKey MaKey. *CHI 2014,* April 26 - May 01, Toronto, Canada.

Román-González, M. (2014). Aprender a programar ‘apps’ como enriquecimiento curricular en alumnado de alta capacidad. *Bordón. Revista de Pedagogía, 66*(4), 135-155. doi:http://dx.doi.org/10.13042/Bordon.2014.66401

Román-González, M. (2015a). Computational Thinking Test: Design guidelines and content validation *Proceedings of EDULEARN15 Conference, 6th-8th July 2015, Barcelona, Spain* (pp. 2436-2444).

Román-González, M. (2015b). Test de Pensamiento Computacional: principios de diseño, validación de contenido y análisis de ítems. In M. Á. Murga Menoyo (Ed.), *Perspectivas y avances de la investigación* (pp. 279-302). España: UNED.

Rosales, A. (2012). Wearable music. Creating sound effects and music by playing, project presented at Ars Electronica 2012, u19 create your world village, festival catalogue.

Royal Society. (2012). *Shut down or restart: The way forward for computing in UK schools*. [2016, May 24] https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Rushkoff, D. (2010). *Program or Be Programmed: Ten Commands for a Digital Age*. USA: BookMobile.

Rushkoff, D. (2012). Code Literacy: A 21st-Century Requirement. [2016, April 3] http://www.edutopia.org/blog/code-literacy-21st-century-requirement-douglas-rushkoff

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2013). Mobile Learning: Tendencies and Lines of Research. In F. J. García-Peñalvo (Ed.), *Proceedings of the First International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'13) (Salamanca, Spain, November 14-15, 2013)* (pp. 473-480). New York, NY, USA: ACM.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2014a). ICTs Integration in Education: Mobile Learning and the Technology Acceptance Model (TAM). In F. J. García-Peñalvo (Ed.), *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'14)* (pp. 683-687). New York, USA: ACM.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2014b). Mobile Learning Adoption from Informal into Formal: An Extended TAM Model to Measure Mobile Acceptance among Teachers. In F. J. García-Peñalvo (Ed.), *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'14)* (pp. 595-602). New York, USA: ACM.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2014c). Understanding mobile learning: devices, pedagogical implications and research lines. *Education in the Knowledge Society*, *15*(1), 20-42.

Sánchez Prieto, J. C., Olmos Migueláñez, S., & García-Peñalvo, F. J. (2016). Informal Tools in Formal Contexts: Development of a Model to Assess the Acceptance of Mobile Technologies among Teachers. *Computers in Human Behavior, 55A*, 519-528. doi:http://dx.doi.org/10.1016/j.chb.2015.07.002

Sansing, C. (2012). Test-Driving the MaKey MaKey. The Arduino kit makes a game of learning. Banana pianos are just a start. *School Library Journal*, april 2015, 12-13. http://www.slj.com/2015/04/opinion/test-drive/slj-reviews-the-makey-makey-test-drive/

Sanz, A. (2015, April 26). Why Teaching and Learning How to Code in Schools. EdTech Review. [2016, September 6] http://edtechreview.in/trends-insights/insights/1934-why-teaching-and-learning-how-to-code-in-schools

Schelhowe, H. (2007). Technologie, Imagination und Lernen: Grundlagen für Bildungsprozesse mit Digitalen Medien. Münster: Waxmann.

Schelhowe, H. (2008). Digitale Medien als kulturelle Medien: Medien zum Be-Greifen wesentlicher Konzepte der Gegenwart. In J. Fromme, W. Sesink (Hrsg.). *Pädagogische Medientheorie*. Medienbildung und Gesellschaft Band 6, Wiesbaden.

Schneider, B., Jermann, P., Zufferey, G., & Dillenbourg, P. (2011). Benefits of a Tangible Interface for Collaborative Learning and Interaction. *IEEE Transactions on Learning Technologies*, *4*(3), 222-232.

Schön, S, Ebner, M., & Narr, K. (2016) (eds.). Making-Aktivitäten mit Kindern und Jugendlichen. Handbuch zum kreativen digitalen Gestalten, Norderstedt: books on demand, online PDF [2016, July 31] http://www.bimsev.de/n/userfiles/downloads/making_handbuch_online_final.pdf

Schonmann, S. & Wagner, E. (Eds., 2015), *The Wisdom of the Many - Key Issues in Arts Education*, International Yearbook for Research in Arts Education, 2015, Münster: WAXMANN

Sentence, S. & Csizmadia, A. (2015). Teachers' perspectives on successful strategies for teaching Computing in school. In A. Brodnik & C. Lewin (Eds.). *IFIP TC3 Working Conference "ANew Culture of Learning: Computing and next Generations". Preliminary Proceedings*. Lithuania, Vilnius University, 201-210. [2016, October 1] http://docplayer.net/16256814-Ifip-tc3-working-conference-a-new-culture-of-learning-computing-and-next-generations.html

Sherman, M., & Martin, F. (2015). The assessment of mobile computational thinking. *Journal of Computing Sciences in Colleges*, *30*(6), 53-59.

Silver, J. S. (2014). *Lens × Block. World as Construction Kit.* [PhD thesis]. Massachusetts Institute of Technology. http://web.media.mit.edu/~silver/Lens-Block-Jay-Silver-PhD-Thesis.pdf

Sklar, E. & Eguchi, A. (2004). RoboCupJunior – Four Years Later. *Proceedings of the Eighth International RoboCup Symposium.*

Sklar, E., Parsons, S., & Stone, P. (2004). Using RoboCup in university-level computer science education. *Journal on Educational Resources in Computing 4*, 1–21.

Spitzer, M. (2012). *Digitale Demenz*. München: Droemer HC.

Spradling, C., Linville, D., Rogers, M., J. Clark, Computing, & Sciences in Colleges, v., no. 5, pp. 115-125, 2015. (2015). Are MOOCs an appropriate pedagogy for training K-12 teachers' computer science concepts? *Computing Sciences in Colleges, 30*(5), 115-125.

Sysło, M. M., & Kwiatkowska, A. B. (2013). Informatics for All High School Students: A Computational Thinking Approach. In I. Diethelm & R. T. Mittermeir (Eds.), *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages. 6th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2013, Oldenburg, Germany, February 26–March 2, 2013. Proceedings* (pp. 43–56). Heidelberg: Springer.

TACCLE 3 Consortium. (2016). TACCLE 3: Coding Erasmus + Project website. Retrieved from http://www.taccle3.eu/

Tan, X. L. (2005). *Storytelling wearables, an alternative biography*. www.xiaolitan.com/thesis/thesis.html [19.2.2010] and http://we-make-money-not-art.com/xiao_li_tans_st/ [12.04.2016]

Toh, L. P. E., Causo, A., Tzuo, P. W., Chen, I. M., & Yeo, S. H. (2016). A Review on the Use of Robots in Education and Young Children. *Educational Technology & Society 19*(2), 148–163.

Trappe, C. (2012). Creative access to technology: Building Sounding Artifact with children. In *Proceedings of IDC 2012*, Bremen, short paper.

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2006). *A model curriculum for K-12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee* (2nd ed.). New York, NY, USA: ACM.

Turbak, F., Pokress, S. C., & Sherman, M. (2014). Mobile computational thinking with APP inventor 2. *Journal of Computing Sciences in Colleges, 29*(6), 15-17.

Turkle, S. (2011). *Evocative objects - things we think with*. Cambridge Ma: MIT

Vee, A. (2013). Understanding Computer Programming as a Literacy. *LiCS. Literacy in Composition Studies, 1*(2), 42-64.

Walter, D., & Sherman, M. (2015). *Learning MIT App Inventor: A Hands-On Guide to Building Your Own Android Apps*. Upper Saddle River, NJ, USA: Addison-Wesley.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127-147. doi:10.1007/s10956-015-9581-5

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: measuring computational thinking in middle school *Proceedings of the 43rd ACM technical symposium on Computer Science Education, SIGCSE '12* (pp. 215-220). New York, NY, USA: ACM.

Why Coding is kind of a Big Deal (s.a.). [2016, October 1] https://www.madewithcode.com/static/why-coding-is-kind-of-a-big-deal.pdf-v3.pdf

Williams, A. (2003). The qualitative impact of using LEGO MINDSTORMS robots to teach computer engineering. *IEEE Transactions on Education, 46*, 206.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35. doi:10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences, 366*(1881), 3717-3725. doi:10.1098/rsta.2008.0118

Wing, J. M. (2011a). Computational Thinking. In G. Costagliola, A. Ko, A. Cypher, J. Nichols, C. Scaffidi, C. Kelleher, & B. Myers (Eds.), *2011 Ieee Symposium on Visual Languages and Human-Centric Computing* (pp. 3-3).

Wing, J. M. (2011b). Research Notebook: Computational Thinking--What and Why? *The Link. The magazine of the Carnegie Mellon University School of Computer Science*.

WISE (2015). *Women in Science, Technology Engineering and Mathematics: The Talent Pipeline from Classroom to Boardroom* [2016, October 16] https://www.wisecampaign.org.uk/uploads/wise/files/WISE_UK_Statistics_2014.pdf

Wolber, D., Abelson, H., & Friedman, M. (2015). Democratizing Computing with App Inventor. *GetMobile: Mobile Computing and Communications, 18*(4), 53-58. doi:10.1145/2721914.2721935

Xu, D. (2005). *Tangible User Interface for Children – An Overview at UCLAN Department of Computing Conference*, Preston, UK. [2016, August 17] http://www.chici.org/references/tangible_user_interface.pdf

Yu, D. (2014). The Research on Teaching Reform of the "University Computer Basic" Course In Police Active Colleges *Proceedings of the 2nd International Conference on Education Technology and Information System (ICETIS 2014)* (pp. 583-586). Amsterdam: Atlantis Press.

Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital. *RED, Revista de Educación a distancia, 46*.

Zynczak, H. (2016). *What Closing The Gender Gap In Tech Would Mean Outside The Industry*. [2016, August 14] https://www.fastcompany.com/3055906/strong-female-lead/what-closing-the-gender-gap-in-tech-would-mean-outside-the-industry