DOI: 10.34641/ctestem.2022.456

A Robotic-based Approach for CT Development: Challenges of Teaching Programming Concepts to Children and the Potential of Informal Learning

Rafael ZEREGA[1*], Ali HAMIDI[2*], Sepideh TAVAJOH[3*], Marcelo MILRAD[4*]
[1,2,3,4] Faculty of Technology, Linnaeus University, Sweden
rafael.zerega@lnu.se, ali.hamidi@lnu.se, st222yd@student.lnu.se, marcelo.milrad@lnu.se

ABSTRACT
In many countries worldwide, Computational thinking (CT) is now considered as a fundamental skill for dealing with the challenges of the 21st century society. One of the most common ways of imparting CT knowledge in K-12 education is by teaching programming and coding, as it requires applying a set of concepts and practices that are essential for thinking computationally. However, learning to program can be challenging and it may take time to develop these skills in the context of school activities. Thus, complementing formal K-12 education with after-school or other types of informal learning activities aimed at fostering CT concepts and practices among young students can be an alternative approach to develop these skills. During the summer of 2021, we carried out a series of workshops in the context of a summer camp taking place at a public library, organized by a local municipality in southern Sweden. These workshops (with a total teaching duration of 20 hours in one week) consisted of activities where children aged 11-14 had to assemble wheeled robots and then program them using a visual language to make them execute different types of tasks and challenges. The outcomes of our study show that roughly one third of the participants managed to program the robots with code that made use of CT core concepts, such as conditionals, loops, and logical operators, among others. The rest of the children did not manage to successfully apply these concepts and thus they could only manage to program sequential linear scripts. We argue that learning to program and understanding some of the main CT concepts, which are for the most part very abstract, is a process that takes time and thus, extracurricular activities can be an effective method to complement formal education and help young students develop their CT and programming skills.

KEYWORDS
Computational Thinking, Computational Concepts, Computational Practices, Programming, Informal learning.

1. INTRODUCTION
Computational Thinking (CT) is a thought process focused on problem-solving that is deemed by many researchers and policymakers as a fundamental skill for dealing with the challenges of the 21st century society. Wing (2006) brought CT to public attention explaining the essence of this concept and advocating for its inclusion in the K-12 curricula. Grover & Pea (2018) argue that CT comprises a set of concepts and practices that are required for formulating a problem and expressing its solution effectively. Lu & Fletcher (2009) are more emphatic about the relevance of teaching CT in schools and argue that it should be taught to every student along with the other *three R's* (reading, writing and arithmetic). As a result of all this advocacy, an increasing number of countries around the world have started to impart knowledge related to digital competence, CT, and programming, as part of their K-12 curricula. Although some authors argue that CT is not solely about programming (Grover & Pea, 2018; Wing, 2006), learning algorithm design to program a computer, a robot or any other computerized machine is an essential skill when attempting to solve a problem through a computational solution, which is one of the main goals of CT (Grover & Pea, 2013).

However, to grasp the true nature of CT and to be able to program a computational artifact, there is a set of concepts and practices that must be understood (Brennan & Resnick, 2012; Grover & Pea, 2018). Some of these concepts are relatively abstract; concepts like algorithmic thinking and sequences, using conditionals, applying loops to repeat a given set of actions and storing data in variables, to mention just a few, could at first be somewhat difficult to fully grasp. Regarding CT main practices such as problem decomposition, iterative refinement, as well as testing and debugging, among others, the situation is not much different. For instance, some studies suggest that novice programmers in K-12 education face difficulties when attempting to detect and debug errors in their code (Carter, 2015; Haduong & Brennan, 2018). Similarly, other studies have focused on some of the common misconceptions regarding programming concepts and the difficulties that young students encounter when starting to learn how to program (Grover & Basu, 2017). Yet another study from Sanders & McCartney (2016) identified a few thresholds programming concepts that tend to be problematic for novice young programmers. Learning the basics of programming can therefore pose several challenges and that is why some authors suggest that this knowledge should be imparted at a very early age. Some scholars advocate for introducing children to CT and programming concepts as early as kindergarten education (Fessakis et al., 2013; Sullivan & Bers, 2016). Furthermore, Lu and Fletcher (2009) argue that students that have been introduced to CT at an early age tend to show higher probability of successfully learning more advanced programming later.

There is, nevertheless, a limited number of hours in the school curricula that can be dedicated to imparting these new subjects and therefore finding other instances to teach CT and programming to young students could be an effective way to help students learn programming concepts and practices. Informal learning activities addressing CT, such as after-school workshops can be an effective manner to complement K-12 formal education and let children acquire additional knowledge in this subject (Ker et al., 2021). In this paper we argue that informal learning activities can offer students the possibility to further explore and test programming concepts and practices, allowing them to deepen their understanding of these matters in a friendly environment and without the stress normally associated to formal education as extracurricular activities are not subject to evaluation in form of official grades. To test the potential of informal learning to foster and develop CT skills among young students we conducted a series of workshops during one week with a group of young students aged 11-14 that had little or no previous experience in programming. These workshops were conducted at the main public library in a city in southern Sweden. We used educational robots, Engino ERP[1], that the children had to assemble and then program so that they would execute a series of tasks and challenges. Considering all the above, we defined two research questions that guided this study: (1) *What are the main challenges when teaching CT concepts and practices to youngsters with little or no previous programming experience?* (2) *What is the potential of informal learning spaces as an alternative for complementing the teaching of CT and programming concepts provided by formal K-12 education?*

The rest of this paper is organized as follows: in section two we provide a background regarding the relation between CT and programming. Section three provides a description of the methodology used for this study. Section four presents the main results and lastly, section five ends this paper presenting our discussions and conclusions on the results.

2. THE ROLE OF PROGRAMMING IN CT

Many countries around the world are currently in the process of modifying their K-12 educational curricula to develop so called *digital competences* (Heintz et al., 2017) and therefore CT has increasingly gained more attention. CT was originally coined by computer scientist, Seymour Papert in his book *"Mindstorms: Children, computers and powerful ideas"* (1980). Papert was one of the pioneers of constructionism, a constructivist learning theory where students create knowledge by exploring, constructing, and testing. This is the reason why building plays a central role in constructionism. CT derives from this learning theory and consequently one of its main objectives is to design and build systems (Wing, 2006). Cuny et al., (2010) further developed the definition of this concept by explaining that CT is a thought process required to formulate a problem and to express its solution in an effective way so that it can be carried out by an information processing agent (such as a computer or even a person). Being able to instruct or program a computerized system is, therefore, a fundamental skill within CT (Grover & Pea, 2018; Kynigos & Grizioti, 2018).

To fully understand the relevance of programming within CT it is necessary to analyze how programming relates to CT and computer science (CS) as a whole. As can be observed in Figure 1, CT and CS are two fields of study that overlap only partly. In other words, CT is not solely about CS (and vice versa). Programming (coding) lies in the intersection between these two worlds and thus it is an important component of CT (Angevine et al., 2017). Although Wing (2006) argues that CT is an approach to problem-solving that is considerably broader than mere programming, she later clarifies and further develops this concept by explaining that CT is a thought process involved in formulating a problem and its solution so that this solution will be effectively carried out (Wing, 2011). CT means, therefore, using computer-based solutions to solve real-world problems and consequently programming is an essential skill necessary for applying CT.

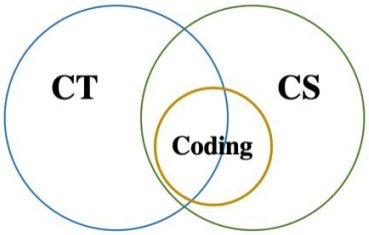

Figure 1. The relationship between Computer Science, Computational Thinking and coding (Angevine et al., 2017)

When considering the importance of programming and algorithm design within CT it is then necessary to consider what different authors call the concepts and practices of CT (Brennan & Resnick, 2012; Grover & Pea, 2018). Other authors use different terms such as CT skills (Mills et al., 2021) and when examining carefully all these terms it is not rare to find some authors using them interchangeably. However, regardless of the exact term used to refer to these different dimensions of CT, there is something that they have in common: they are all directly or indirectly related to the process of programming and building algorithms. Being able to have a good understanding of what these concepts and practices are all about is therefore essential to understand the process of designing algorithms and programming computerized devices. In this study we aim to analyze how the participants of the workshops mentioned earlier managed to make use of these CT concepts and

[1] https://www.engino.com/w/

principles when they were assembling and programming robots.

3. METHODOLOGY

In this section we present core aspects regarding the design of the study in terms of the workshops' settings, the participants, the technological equipment used for teaching and the type of data collected for the later analysis.

3.1. *Settings and Participants*

This study was carried out based on a series of workshops that took place in late June 2021. These workshops were conducted at the main public library in a city located in southern Sweden to kick off the summer vacation. Five workshop sessions were held, from Monday to Friday, each session lasted for four hours. The participants were seven boys and two girls aged 11-14 who, although having received education on digital competence in school, they had little or no previous experience in programming. The workshops were led by two tutors in charge of explaining the topics to be learned during each session and helping the participants in case they request assistance (see Figure 2).

3.2. *Workshops Design and Theoretical Foundation*

As mentioned above, for this study we conducted five workshops, one every day from Monday to Friday. Each workshop had a duration of four hours (a total of 20 hours for the entire workshop series). Each student was given a laptop computer where they could visualize the assembly instructions and run the software required for building the algorithms to program the robots. The activities carried out in each of the five workshop sessions were as follows: (1) assembling the robots, learning about the sensors and creating simple linear algorithms, (2) using loops and conditionals in algorithms and learning about Boolean data type, (3) learning more about conditionals, using logic and arithmetic operators and using Integer data type, (4) using variables and deepening on the use of loops and conditionals, and (5) free practice and testing what has been learned during the workshop series.

As for the programming-related activities, the main objective was that children would make use of the different types of sensors (ultrasonic sensors, infrared sensors and color sensors) and by designing algorithms they would program the robots so that they would interact with their surroundings and execute tasks such as, avoiding obstacles, following the borders of a path, deciding to turn left or right based on the clear space available on each side, among others. To instruct the robots for executing such tasks, the children would have to build algorithms that make use of programming concepts such as conditionals and loops, as well as using logical and arithmetic operators, among others. For this purpose, at the beginning of each workshop and before starting with the actual hands-on activities of the day, the tutors gave a brief keynote presentation where they introduced the students to different concepts of programming, explained how robots interact with the physical world and to which extent they are present in our daily lives.

The assessment of the learning process for each workshop session was based on observing whether the robot was executing the task that the children had intended to program and by analyzing the actual algorithms that they had made using the block-based programming platform provided by the Engino ERP.

These workshops were designed taking in consideration the notions of constructionism, aimed at offering student-centered activities and allowing children to explore and test their ideas through building and collaborating with their peers and instructors (Papert & Herel, 1991). Constructionism, pioneered by Papert, puts the emphasis on allowing students to generate their own knowledge by building and experimenting while the educator plays the role of a consultant or coach. The idea was that during the workshop series the participants could learn about robots by showing them through examples that robots and other types of automated devices are increasingly present in our current society. By giving the children the chance to assemble their own robots and program them, so that they can interact with the environment, the children could not only learn CT concepts, but also get an insight on how robots work as well as understanding what is their potential to improve our lives and what are the risks associated with this technology.

Figure 2. Workshops at the public library

In addition, based on the ideas from Laurillard (2013), we regarded the process of teaching as a design science. For this study, the design of the workshop series was done creating learning activities based on the concepts of the TPACK (technological pedagogical content knowledge) framework for the effective use of technological tools to support and enhance the learning process (Mishra & Koehler, 2006).

3.3. *Educational Equipment*

For this study we used a set of educational programmable robots called Engino ERP, which is targeted to kindergarten, elementary and secondary students (depending on the model). Engino ERP is a line of construction kits that use various sensors that allow the user

to build and program robots that can interact with their surroundings. These robots can be programmed using a special software that offers a block-based programming environment to let children build algorithms in a syntax-free coding mode. The Engino ERP includes a wide range of sensors that allow the robots to execute different types of tasks as they measure different parameters from the environment. For this workshop series the children worked with three types of sensors: infrared sensor, color sensor and ultrasonic sensor.

3.4. *Data Collection and Assessment*

During all five workshops the researchers took field notes, photographs, and screenshots of the computers where the children were building their algorithms to program the robots. This data was analyzed using a qualitative approach to identify which were some of the most challenging computational concepts and principles in the process of learning to build and program the robots. By analyzing the children's code and the performance of their robotic creations we attempted to get an insight regarding how the children managed to use fundamental programming and CT concepts such as algorithmic sequences, conditionals, loops, and logical operators, among others. To assess the learning progress of the children during the workshops, we took into consideration the CT concepts and practices defined by Brennan & Resnick (2012).

4. FINDINGS

This section will present the most relevant findings based on the data collected during the workshop series. We divided these findings into two areas: (1) *physical assembly* and (2) *CT and programming.*

4.1. Physical Assembly

Two types of ERP sets were used during the workshop activities in order to explore how constructing methods influence the children in terms of their CT practices, such as being incremental, reusing and remixing, modularizing, testing and evaluating. The children worked with semi-built robots that were to be completed and modified either by following the step-by-step 3D instructions, that they had on their computers, or by resorting to their own inventiveness. All nine participants preferred to build the robots based on their own inventive ideas rather than by following the instructions. The children showed more engagement when constructing their own creations. Several children mentioned that building freely was more amusing than building by following instructions. In addition, the children tended to lose both interest and focus when they faced a situation where the assembly process was particularly difficult. A big challenge that the children faced in terms of the physical assembly was to find the best way to mount and position the sensors on the robots so that they would get an accurate reading of the surroundings. Whereas some children would become frustrated and annoyed when they could not manage to position the sensors correctly to get an accurate reading, others were particularly motivated to test many times until they found the best way to position the sensors.

4.2. *CT and Programming*

As mentioned earlier, the children participating in these workshops had practically no previous experience doing any type of programming. The brief keynote presentation that took place at the beginning of every workshop in combination with the hands-on activities allowed all the children to get a rough understanding of what an algorithm is. All nine children managed to design simple linear algorithms that could instruct the robots to execute simple tasks such as going forward, turning right, left, and stopping. However, only four children managed to successfully design algorithms that made use of some of the main computational concepts, such as conditionals, loops, and logical/arithmetic operators (see Table 1). Without these programming concepts, the robots could only be instructed to execute a fixed sequence of actions (linear algorithm in Table 1), but they would not be able to interact with the environment in any way.

Table 1. Types of Computational Concepts that the participants managed to successfully use in their algorithms (*yes* means used successfully).

Student ID	Linear Algorithm	Loop	Conditional	Logic and arithmetic operators	Variables
1	Yes	No	No	No	No
2	Yes	Yes	Yes	Yes	No
3	Yes	Yes	Yes	Yes	No
4	Yes	No	No	No	No
5	Yes	No	No	No	No
6	Yes	Yes	Yes	Yes	Yes
7	Yes	No	No	No	No
8	Yes	No	No	No	No
9	Yes	Yes	No	No	No

As for the use of sensors, among the children that used sensors in their robots, the one that was used the most was the infrared sensor. The children mentioned that it was fun to use this sensor because it allows them to do many different tasks with it and it was easy to set up. The other two sensors (color sensor and ultrasonic sensor) were used very seldom. According to the children, the color sensor was hard to use because the calibration process to set it up required a considerable amount of trial and error to get it working correctly. The ultrasonic sensor, although easy to set up because it did not require any type of calibration, was used successfully by only one of the participants. It is important to mention that the infrared sensor uses Boolean data type (data that has one of two possible values: true/false). The color and ultrasonic sensor, on the other hand, use Integer data type (in this case positive whole numbers and zero). According to the children, working with Boolean data was easier and more straightforward than working with Integer data, which may explain why only few students managed to successfully use the color and ultrasonic sensor.

In the next and final section, we present our discussions and conclusions based on the data we collected during the workshop series. We divided it into five subsections to make it easier to connect the discussions with the topics of the research questions that guided this study.

5. DISCUSSIONS AND CONCLUSIONS

5.1. *The Importance and Challenges of Learning CT Concepts*

Learning to program requires being able to understand a set of CT concepts and practices, which can be a challenging and long process. Taking into consideration the computational concepts defined by Brennan & Resnick (2012), such as sequences, loops, and conditionals, we can notice that after having completed the workshops, all nine participants understood that an algorithm is an expression of a sequence of individual instructions that a computerized machine executes. Indeed, all nine children managed to instruct the robots to execute tasks such as making it move forward and then turn at certain points to describe, for example, a geometrical shape (linear algorithms). However, only three children managed to successfully design an algorithm that would include a computational concept that would allow the robots to use the data coming from the sensors to be able to interact with the environment. Basic control structures such as *if-else conditionals*, *for loops* and *while loops* are fundamental computational concepts to have a robot or other computerized machine make decisions based on the information that is coming from the sensors. Learning to build algorithms using these programming concepts will not only allow the robots to interact with the environment but it will also serve the children as a means of exploration and a way to create and express computer-based solutions to real-world problems. Engaging in programming offers young students the possibility to exercise a set of different computational concepts and higher order thinking skills, such as reasoning, analyzing and evaluation (Falloon, 2016).

5.2. *The Challenges of Building and Debugging Collaboratively*

The results based on the workshops we conducted suggest that teaching programming concepts poses some difficulties. Not only was it challenging for the students to fully understand some of the programming concepts as they struggled when asked to explain their own algorithms, but it was relatively hard for them to be able to identify the origin of the problem in their code when the robot was not able to execute the task successfully. The difficulties in detecting and debugging errors in the code of novice programmers are not uncommon and they occur even among students at college level (Carter, 2015). The situation is more evident among young students in primary and secondary schools. Haduong & Brennan (2018) argue that best practices of code debugging are, for the most part, undefined in K-12 education. Not being able to fix a piece of code can be extremely frustrating and demoralizing for young students and therefore it is extremely important to understand the relevance of teaching young students some basic rules regarding how to identify and debug errors in the code and other types of problems that may arise. The physical assembly was too a relevant aspect of the CT development and a challenge for the children as this activity required them to build together, communicate and exchange ideas, for example, when the children were discussing the best way to position the sensors, and when they were evaluating their construction methods and results.

5.3. *The Convenience of Block-based Programming for Novice Programmers*

Programming activities that use block-based programming for introducing children to CT and algorithm design are a good choice as the graphical interface allows them to build algorithms and focus on computational concepts and practices without the need to take care of the syntax associated with text-based programming. Mladenovic et al. (2018) sustain that most novice programmers often focus on the syntax of the programming language instead of the meaning and logic of the algorithm itself, a problem that can be overcome with visual block-based programming. For instance, during the workshops all the participants had some difficulties when learning about the main difference between *if-else* and *while* as a control structure when programming the robots to *make decisions* based on the data coming from the sensors. We noticed that the children felt very comfortable using the block-based programming environment of the Engino ERP as they could easily switch between different control structures (such as *if-else* and *while*) just by dragging and dropping the respective block elements to test different possibilities quickly and easily.

5.4. *The Potential of Informal Learning Environments for Developing CT Concepts and Practices in Children*

Programming is an activity that deals with abstract concepts and therefore one of the main challenges of teaching CT concepts and practices to people with little or no previous programming experience is to clarify misconceptions regarding computational concepts such as conditionals, loops, variables, and Boolean logic (Grover & Basu, 2017). Based on our findings it is possible to notice that reaching a full understanding of how to apply computational concepts and being able to successfully use control elements like loops and conditionals, associated with logical operators, can be a challenging process that takes time. Extracurricular activities like these workshops conducted at the public library can play a relevant role as a complement to formal education. Informal learning instances for developing CT and programming skills give children the chance to explore and test their computational creations in a friendly grade-free learning environment. It is, however, essential that the teaching activities are thoroughly designed so that the tutors will be able to use the computational tools they have (robots in this case) in a meaningful way and with a strong focus on constructionism. Also, the tutors must be able to explain,

both with words and by doing, the fundamental CT concepts and practices to avoid some misconceptions that tend to arise when teaching these abstract concepts to novice young programmers. Lastly, by continuously giving examples of robotic systems used in the real world, children become more motivated to learn how to program robots, as they see them as something more real and more meaningful, giving the learning experience another level of authenticity.

5.5. *Future Work*

In future studies we intend to explore in which way the programming interface of educational robots may influence how students understand CT and programming concepts, especially among novice programmers.

6. REFERENCES

Angevine, C., Cator, K., Roschelle, J., Thomas, S. A., Waite, C., & Weisgrau, J. (2017). *Computational Thinking for a Computational World*.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (Vol. 1, p. 25).

Carter, E. (2015). Its debug: Practical results. *Journal of Computing Sciences in Colleges*, 30(3), 9–15

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress, referenced in http://www. cs. cmu. edu/~ CompThink/resources/TheLinkWing. pdf*.

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning*, *32*(6), 576-593.

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.

Grover, S., & Basu, S. (2017, March). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 267–272). Seattle, Washington: 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.

Grover, S., & Pea, R. (2018). Computational Thinking: A competency whose time has come. C*omputer science education: Perspectives on teaching and learning in school*, London: Bloomsbury Academic, 19-37.

Haduong, P., & Brennan, K. (2018, February). Getting unstuck: new resources for teaching debugging strategies in scratch. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 1092-1092).

Heintz, F., Mannila, L., Nordén, L. Å., Parnes, P., & Regnell, B. (2017, November). Introducing programming and digital competence in Swedish K-9 education. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 117-128). Springer, Cham.

Ker, C.L., Wadhwa B., Seow, P. S.K., & Looi, C.K. (2021). Bringing physical computing to an underserved community in an informal learning space. In C. K. Looi, B. Wadhwa, V. Dagiené, P. Seow, Y. H. Kee, & L. K. Wu (Eds.), *Proceedings of the 5th APSCE International Computational Thinking and STEM in Education Conference 2021* (pp. 101-106). Asia-Pacific Society for Computers in Education.

Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating Turtle geometry, dynamic manipulation and 3D Space. *Informatics in Education*, *17*(2), 321-340.

Laurillard, D. (2013). *Teaching as a design science: Building pedagogical patterns for learning and technology*. Routledge.

Lu, J. J., & Fletcher, G. H. (2009, March). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 260-264).

Mills, K., Coenraad, M., Ruiz, P., Burke, Q., & Weisgrau, J. (2021). *Computational Thinking for an Inclusive World: A Resource for Educators to Learn and Lead*. Digital Promise.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record*, *108*(6), 1017-1054.

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies*, *23*(4), 1483-1500.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas (1st Edition)*. New York, Basic Books.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, *36*(2), 1-11.

Sanders, K., & McCartney, R. (2016). Threshold concepts in computing: Past, present, and future. *Proceedings of the 16th Koli Calling international conference on computing education research*, Finland.

Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, 26(1), 3–20.

Wing, J. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-36.

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The link magazine*, *6*, 20-23.