Education and Information Technologies (2019) 24:3823–3840
https://doi.org/10.1007/s10639-019-09964-6

Investigation of pre-service computer science Teachers' CS-unplugged design practices

Polat Sendurur[1]

Received: 7 January 2019 / Accepted: 3 July 2019 / Published online: 8 July 2019

Abstract
Computer Science Unplugged (CS-Unplugged) has been evolving in computer science education. It is a promising approach especially for introductory programming skills and computer science concepts. The skills of the computer science instructors/teachers, who convert the approach into practice, can be crucial during the preparation of CS-Unplugged activities. In this study, CS-Unplugged activity creation skills of pre-service computer teachers were examined. The participants prepared CS-Unplugged activities for selected topics considering the target learners they selected. The prepared activities were subjected to content analysis. According to the results, it was found that participants were significantly incompetent to design CS-Unplugged activities. They failed to define necessary rationales and resulting context for the activities clearly. Although the activities generally do not include story or game structures, participants expressed that those types of structures could improve their activities in terms of student motivation and retention. On the other hand, different kinds of physical objects were integrated into the activities with necessary links to programming concepts and those activities were found easy to implement. Activity duration was found the most significant force in front of designing activity and grade level and age are the only variables that participants consider to define the context.

Keywords CS-unplugged · Design patterns · Pre-service teachers · Computer science education

1 Introduction

Computers and related technologies have been evolving dramatically, and their places in educational systems have been expanding day by day. It is inevitable that such a rapid improvement comes with new approaches of computer science education (CSE).

✉ Polat Sendurur
polat.sendurur@omu.edu.tr

1 Ondokuz Mayis University, Samsun, Turkey

Computer science unplugged (CS-Unplugged) is one of these approaches that have become popular for the last few years. The approach enables the educators to teach introductory computer science knowledge and skills without physical computers or any other digital information processors. Therefore, the educators using CS-Unplugged do not need electricity, computer, the Internet, or similar technological tools, and they can adapt their instruction for almost any environment. Moreover, the inclusion of game and entertainment elements can serve as the source of effective motivation for students.

The organization of CS-Unplugged was found by Tim Bell, Ian Witten, and Michael Fellows. They define these activities as simply using objects and materials that are easy to access to teach computer sciences. The main aim is not to develop a method to educate computer experts or scientists, but to transfer the basics of computer science to students and to make them aware of this discipline. In other words, CS-Unplugged approach provides the area of primary education with a wider spectrum of CSE activities. As a matter of fact, the CS-unplugged activities have been appearing more often in the curricula of different countries. Physical conditions, programing background of students, or unique structures of different curricula limits the number of already designed CS-Unplugged activities that teachers can reach. That' s why it is important to equip teachers with the competencies of designing and developing their own CS-Unplugged activities.

In (removed for blind review), all teacher-training programs were revised and updated in 2018 by (removed for blind review) Council of Higher Education. The redesigned program includes a new course named as "Approaches in Teaching Programming", which aims to equip teacher candidates with different skill-sets, and thus they can become competent in teaching programing to students of different grade levels. Some of the skills to be developed throughout this course consist of teaching block-based programing tools, utilization of games for programing education, and evaluation of students' programming abilities. On the other hand, there is not any sign of CS-Unplugged activities in the mentioned course. Additionally, neither did the former program had any course or course topic devoted for CS-Unplugged activities.

Teachers, who graduated either before or after the redesigned program of teacher training, are all supposed to teach the "information technologies and software" course in primary and secondary schools. Nevertheless, the curriculum of this course expected these teachers to use CS-Unplugged activities, however, the teachers had no formal training with regard to CS-Unplugged approach. They have to design, develop, and implement these types of activities, which means an inconsistency between the theory and practice. In other words, there is a need for questioning the abilities of graduates, teacher candidates, pre-service computer science teachers, and the training programs in terms of design and development of CS-Unplugged activities. In the light of all these considerations, this study aims to investigate CS-Unplugged activities that are designed by pre-service teachers in terms of a group of criteria. In addition, based on the findings, this study aims to propose suggestions for further renovation of computer science teacher training programs leading to train more competent teachers. The research questions are;

1. How are the design patterns of CS-Unplugged activities created by pre-service computer science teachers?

2. What are the distinctive features of CS-Unplugged activities created by pre-service computer science teachers?

2 Literature review

As educational paradigms evolve with technological developments, the allocation of personal resources for tools and training of technology enhances, too. In this respect, the importance of formal courses, especially the ones about computer science, have been gaining importance day by day. Today, computer science is not a field that is only owned by computer scientists any more. Instead, it has become one of the major issues of laypeople.

Computer science brings about certain higher-order thinking skills as well as contributing to the improvement of them. For example, computational thinking is one of these skills. Wing (2006) defined the term solving problems, designing the system, and giving meaning of human behavior using computer science and basic terminology. On the other hand, it might be wrong if one considers the computational thinking as one single skill. ISTE (2011) includes the skills of analysis, algorithmic thinking, generalizing, etc. within computational thinking category. Moreover, the elements such as exploring, and model building are also respected as cognitive competencies (ISTE 2016).

In the similar vein, model construction skills consist of a group of sub-skills. According to Jonassen (2006), model construction or modeling incorporates certain cognitive skills such as hypothesis testing and reasoning. In short, it can be inferred that the improvement of various cognitive skills is supported by both computer science and the way to be followed to teach.

In recent years, it can be clearly observed that CSE have become very popular. The number of coding and robotics workshops, schools equipped with STEM laboratories, free online programming courses, and many others show that computer science strikes attention of learners and teachers at any level. Today, one can even see introductory materials for coding in the toyshops, and thus the starting period of CSE may intended to start in pre-school. In some countries, CSE has already become a part of regular curriculum (Heintz et al. 2016). However, the integration of both current pedagogical knowledge and the field of practice regarding instructional methods into CSE still remains as a challenge in comparison to other subject areas. In fact, Cassel et al. 2007) define this situation as a crisis in CSE because there are both a decline in students' interest toward computing and a "lack of public understanding about computing and increasing complexity of introductory courses" (p.330) Moreover, fear, negative perceptions, and doubts about computing science can be listed as among other reasons of that crisis. For example, Bergin and Reilly (2005) found significant relationship between comfort level of students and their understanding of programming concepts. Askar and Davenport (2009) studied engineering students programing self-efficacy beliefs from different perspectives and concluded that there are a number of variables affecting students' self perceptions of whether they can be successful in programing or not.

In the literature, there are certain approaches that are utilized for CSE. Gamification (Swacha and Baszuro 2013) and storytelling (Kelleher et al. 2007) are some examples of those approaches. In addition, blended learning, which is built upon both face-to-face and online environments, is another preferred approach for CSE (Deperlioğlu and Köse 2010). Moreover, there is a more recent trend having a computer free approach, called CS-Unplugged. It aims to introduce basic computer science concepts to learners without using a computer or any digital elements in the field. Since pedagogical approaches in CSE can be named as an evolving field, CS-Unplugged may be one of the promising approaches in terms of contribution to CSE.

In programming education, the comprehension of conceptual structures, the construction of basic concepts, and the creation of model solutions for real life problems with the utilization of programming tools are all in demand of crucial cognitive skills. In other words, the inclusion of tools or elements of real life may play an important role for programming education. For example, Liu and Wu (2018) found that the integration of daily language and dialogues into CS-Unplugged activities resulted in less syntax errors besides better solutions of programming problems. Moreover, this might help both to improve programming skills and to save time allocated for learning. In their study, Alamer et al. (2015) observed that students, who experienced CS-Unplugged activities, transferred what they learned into projects or activities more easily in comparison to those experienced traditional activities. In another study, Giordano and Maiorana (2014) compared different programming education approaches but reported significant contribution of CS-Unplugged activities in terms of learning basic concepts. Moreover, this contribution is not limited to learning, but it also affects both perceptions and motivations towards computer science in a positive manner (Demšar and Demšar 2015; Thompson and Bell 2015). For instance, a three-year longitudinal study, which investigated effect of unplugged activities on 6th and 7th graders intrinsic motivation, showed that students have high intrinsic motivation in unplugged activities (Jiang and Wong 2018).

CS-Unplugged activities, which have been frequently used in recent years, have now become a teaching method or approach in CSE. Bell et al. (2008) emphasized that the inclusion of unplugged computer science teaching programs actually means the formation of a special teaching method for CSE.. There are a number of efforts embracing the idea of using CS-Unplugged as a teaching method. For example, Coffman-Wolph et al. (2018) prepared and used unplugged activities, which covers topics such as variables, loops, and sorting algorithms, and those were inspired by CS-Unplugged organization. In another study, 9–12 years old pupils participated in unplugged activities in order to improve mathematical problem solving and reasoning skills (Wohl et al. 2015). The concepts including input, output, and Boolean were introduced by unplugged activities they created. The current literature includes many other recent studies which either introduce or use unplugged activities as an instructional and special teaching method for CSE of students from the pre-school to tertiary level (Burke et al. 2016; Jiang and Wong 2018; Brackmann et al. 2017).

In short, there is a growing body of literature about the use of CS-Unplugged activities as a teaching method in CSE. In fact, teachers' perceptions about use of CS-Unplugged are quite positive. Sentance and Csizmadia (2017) reported that teachers think CS-Unplugged is one of the strategies working well to teach computing. At that point, there is a need for the pedagogically and technically appealing activities prepared by professionals. Therefore, it is important for instructors, who deliver CS-Unplugged activities, to prepare activities that meet the intended instructional goals within the boundaries of target learners' competencies. In such context, it can be inferred that teachers should possess the competencies for preparing and practicing those activities (Hazzan et al. 2014), because during CSE, instructional materials can become the main barrier (Mouza et al. 2016). That' s why training teachers, who are capable of preparing appropriate instructional materials of CSE, might be a key issue in teacher training. Current teacher training programs are also at high importance for the development and improvement of these skills.

3 Method

The single case study was used as a research design to understand dimensions of CS-Unplugged activity design practices of pre-service computer science teachers. Participants developed CS-Unplugged activities on the topics included in the official curriculum of Information Technologies and Programming course. According to Creswell (2014), the case study is a research approach used to explore a case or multiple case in depth. The study was designed as a single case study. In this case, pre-service computer science teachers were assigned to create CS-unplugged activities for the objectives they selected from the curriculum of information technologies and software course. They worked in pairs and prepared the activities. They submitted final version of their products through the course LMS. In order to have insights of their evaluations and perceptions on their CS-Unplugged activities, volunteer participants were interviewed after the analysis of activities. Unstructured interview sessions were conducted with participants and discussed about their activities in terms of the pattern introduced by Nishida et al. (2009). Results of interviews were used to triangulate the data collected through CS-Unplugged activities.

3.1 Participants

Convenient and purposeful sampling techniques were used together to form the group of participants of the study. These sampling techniques are appropriate sampling techniques for qualitative researches (Johnson and Christensen 2004). Since the purpose of the study is built around the CS-Unplugged activity creation skills of pre-service teachers, participants were selected among third year (junior) pre-service teachers. They completed major programming courses. One of the courses covers visual programing tool Scratch. Another one covers Java programing language. They also participated in the course named as Web Based Programming. There are other computer science courses that participants took before the study. The students also participated in different pedagogical and teaching methodology courses. In both computer science and pedagogy courses, students developed different programing projects and prepared teaching activities. They conducted micro-teaching lessons. They prepared lesson plans and different kinds of instructional materials for teaching computer science. For all of these reasons, they can be assumed as relatively experienced about programming concepts and special teaching methods, techniques and strategies. 64 third-year pre-service computer science teachers attending teaching methods course participated in the study. Convenient sampling was used for interviews, too. After completing the CS-Unplugged activities, pre-service teachers were asked to attend interviews about their CS-Unplugged design process. 14 participants accepted the request.

3.2 The role of the researcher

The researcher was also the instructor of the course that participants attending. He introduced the definition of CS-Unplugged and provided sample activities. He did not provide any guidance about the design process of a CS-Unplugged activity. Participants were expected to use their instructional design knowledge and skills that they already had. The researcher, as the instructor of the course, provided necessary guidance to determine the topics for activities. The researcher also provided feedback and answered student questions in order to prevent misconceptions and misunderstandings. Throughout the design and development of CS-Unplugged activities.

3.3 Data collection and analysis

Participants were informed about CS-Unplugged activities in the beginning of the course. The instructor also presented different sample activities as well as explaining and discussing about them during classes. Then, students were separated into pairs in order to optimize workload and to let them benefit from each other' s experiences, because they have different levels of programming knowledge and skills despite completing the same courses. On the other hand, they were not given any templates or checklists to help them during the creation process. The purpose for not delivering such scaffolds was to eliminate the possibility of affected (or manipulated) results and directed or biased data collection. Participants were asked to submit their activities at the end of fortnight period. All activities were submitted online via course LMS.

Themes and categories, which are necessary for content analysis, were developed by the pattern introduced by Nishida et al. (2009). Table 1 presents the themes about the pattern of CS-Unplugged activities with the categories inferred from activities and Table 2 presents distinctive features. During data analysis process, each activity was examined according to its distinctive features and patterns.

In order to increase the reliability of data analysis process, two researchers from the field of educational technology analyzed a randomly selected activity via the themes and categories provided in Tables 1 and 2. An inter-coder reliability between these two researchers were calculated by using Cohen's Kappa formula and it was found $\kappa = .77$. Any value between .61 and .80 indicates a substantial agreement between the coders. Therefore, it was assumed that there is not significant threat to reliability in the data analysis process.

In order to triangulate the data, unstructured interviews were conducted with 14 participants about their activities. Each activity was examined together with the participant who created it and the researcher. Following that examination, conversation sessions were held around existence of the pattern introduced by Nishida et al. (2009) in the activity. Sessions were recorded with the consent of the participants and transcriptions were shared with them, too. The data were subjected to content analysis and the results were used to triangulate the initial findings obtained from analysis of activities. Same themes and categories used for both analysis of activities and interviews.

Table 1 CS-Unplugged design pattern

Themes	Categories
Context	Age
	Grade level
	Age and grade level
	No information
Forces	Time
	Number of students
	Background knowledge of students
	Classroom environment
Solution/ Proposals	Data types
	Algorithm
	Variables
	Arrays
	Flow charts
	Loops
	Conditionals
	Basic code processing
	Coding logic
	Constants
	Differentiating constants and variables
	Problem solving
	No proposed solution
Resulting context	Understand logic behind algorithm
	Explain the concept of algorithm
	Explain the flow chart components
	Differentiate data types
	Understand the structure of arrays
	Give examples for if-else conditions
	Detect errors
	Differentiate constants from variables
	Understand the transformation between variables
	No defined resulting context
Rationale	Students' motivation to learn how to code can be increased
	Students' misconceptions can be easily detected

4 Results

4.1 Patterns of CS-unplugged activities

Table 3 provides information about the context in which participants' CS-Unplugged activities take part. One activity does not include any clues of context. While two activities only point the necessary age limits to complete activity, 18 of them only

Table 2 Distinctive features

Themes	Categories
Game or challenging environment	No game/activity
	Games including algorithm creation/execution
	Competition based activities
	Labyrinth/maze-based games
Use of physical objects	Work sheets
	Cardboards
	Watches/chronometers
	Boxes
	Play cards
	Magnets
	Figures
	Color pens
	Rulers
Student directedness	Yes
	No
Easiness in implementation	Easy implementation
	Excessive workload for preparation
	Necessity for too much background knowledge
	Classroom management issues
	Time issues
Growing body of ideas	No growing body of ideas
	Adaptation to different learning environments and level of students
	Connection to other programing concepts
Sense of story	No sense of story
	Daily routines
	Finding the way

provide with grade levels. The information of both age limits and grade levels are available together in 18 activities. The activities do not involve any other kind of contextual information.

In order to triangulate the data collected throughout the activities, interviews were conducted with volunteer students. All of the interviewees did not indicate any other contextual information. On the other hand, there were not any contextual information in

Table 3 Context of CS-Unplugged activities

Contextual element	*f*
Age only	2
Grade level only	18
Age and grade level	18
No information	1

one of the activities and the interviewee who prepared this activity indicated that age and grade levels of students should be the fundamental contextual information.

The forces effective on the success of CS-Unplugged activities were summarized in Table 4. Four different forces were mentioned in the activities. The first one is the time limit. Almost all of the activities consider the time as a force that has influence on the success ($N = 38$). The number of students required to complete the activity is another force but only indicated by three different activities. Background knowledge of students ($\mathrm{N} = 3$) was also demonstrated among the forces. For example, the knowledge about basic algorithm is expected as background knowledge to teach conditionals through activity-24. The last perceived force was the classroom environment ($N = 1$).

Similar to the findings provided in Table 4, the participants emphasized the fact that classroom environment was not an effective factor in the success of the activities ($N = 8$). While these participants were underestimating the potential effects of classroom environment on the flow of activities, only one participant expressed the importance of that factor and he was the owner of the activity, which was categorized as assuming classroom environment as an effective force. He expressed his thoughts as;

> *Students need to move constantly in the classroom during the activities. If there is not enough space to move, the chaos is inevitable. Control of students and activity can be easily got lost. So the more students in the classroom, the more space needed. Otherwise, class management could be too difficult to get intended results.*

Even if the activates do not clearly define the problems claimed to be solved, some solutions were proposed as guides. The construction of the algorithm concept and the detection of misconceptions about algorithms ($N = 12$) are the most frequent proposal mentioned in the activities. Conditional related proposals ($n = 10$) were also found another frequently mentioned proposal by the participants. Other proposals were listed in Table 5.

During the interviews, participants were asked to give details about why they provided those solutions. Four of the interviewees proposed solutions to algorithm related problems through activities. Without any exception, all of them expressed that it was easy to imagine an algorithm activity without using computers. Interviewee-1 explained this situation as;

> *I prefer to create an activity for algorithm. It is easy, I think. I can give many examples from daily life. We have many routines and they are linear. We do these in a step-by-step manner, like preparing an omelet or brushing our teeth.*

Table 4 Forces effective on CS-Unplugged activities

Forces	*f*
Time	38
Number of students	3
Background knowledge of students	3
Classroom environment	1

Table 5 Proposals of CS-Unplugged activities for programming problems

Proposal	*f*
Data types and variables	4
Algorithm	12
Arrays	2
Flow charts	4
Loops	2
Conditionals	10
Basic code processing	1
Coding logic	3
Constants	1
Problem solving	2
No proposed solution	1

In addition to easiness of preparing CS-Unplugged activities for algorithm, two participants also emphasized that algorithm is a fundamental concept and it could be troublesome to correct algorithm related problems in the advancement of coding education process. The interviewees who prepared activities for variables ($N = 2$) and conditionals ($N = 3$) made similar explanations. With the following words, Interviewee-3, who prepared an activity about conditionals, explained why an activity in this topic is easy to prepare and fundamental to advancement in coding education;

> *I found this topic easy. In addition the concepts of "if-else" should not be taught directly in programming environment. It is necessary to make them concrete. I see that students have troubles to adapt these concepts to real life. I've experienced this and I also observe it in the schools we visited in the course of School Experience". They cannot see the conditionals in real life during coding.*

The majority of CS-Unplugged activities do not clearly define the resulting context of their proposed solutions ($N = 28$). On the other hand, 11 activities explain 9 different resulting contexts (see Table 6). The weakness in defining resulting context of the activities was also observed in the interviews. They were asked about the lack of information related with resulting context of their activities but majority of them did not express any thoughts. Only one participant (Interviewee-6) pointed out that he chose the topic of his activity from the official curriculum of information technologies and software course and the resulting context is already presented by the curriculum.

It was observed that the participants demonstrated similar patterns in defining the rationales for CS-Unplugged activities during the definition of the resulting context of activities. Only two groups made the rationales explicit for their activities (see Table 7). According to them, the 8th activity can be used especially to determine students' misconceptions about the algorithm, while the 26th activity was designed to make students understand the coding process to develop an interest in this issue.

Unlike the analyzed activities, interviews with participants demonstrated that students have some rationales for their activities. Some of them (N = 3) strongly enhanced that CS-Unplugged activities can reduce students' perception of complexity in terms of coding. Interviewee-7 said that game nature in the activities might help to break

Table 6 Resulting context

Context	*f*
Understand logic behind algorithm	2
Explain the concept of algorithm	2
Explain the flow chart components	1
Differentiate data types	1
Understand the structure of arrays	1
Give examples for if-else conditions	2
Detect errors	1
Differentiate constants from variables	2
Understand the transformation between variables	1
No defined resulting context	28

students' initial fear or biases against coding. Six different participants also repeated the second rationale presented by Table 7.

4.2 The features of CS-unplugged activities

According to Nishida et al. (2009), a CS-Unplugged activity should possess certain features, and being computer free is one of the important ones. Although the participants prepared their activities on computer, their practice required no computers at all. In other words, all participants designed all activities in computer free format, which is parallel to statement made by Nishida et al. (2009).

The design of activities should also be constructed upon the principles of games or include challenging environment. Table 8 summarizes the information about the game structures of the designed activities. The majority of them ($N = 20$) did not propose any game or challenging environment. However, labyrinth/maze structure constituted basis for 10 of the activities. There were rules and assigned tasks (missions) to be conducted within those mazes. 7 activities, on the other hand, consist of scenarios referring to either algorithm construction or algorithm tracking. Only 4 activities had competitive structure among students or against time.

Half of the activities prepared by the interviewees do not propose any game or competition based environments ($N = 7$). Interestingly, six interviewees think that their CS-Unplugged activities could offer a game based environments even though they did not provide any rule or definition for the game structure. For example, Interviewee-11 stated she did not think to include game elements in her activity and continues with the following words;

Table 7 Explanation of rationale for the CS-Unplugged activities

Rationale	*f*
Students' motivation to learn how to code can be increased	1
Students' misconceptions can be easily detected	1

Table 8 Game or challenging environment

Game/activity	*f*
No game/activity	20
Games including algorithm creation/execution	7
Competition based activities	4
Labyrinth/maze-based games	10

The group I targeted is composed of younger students. I thought they might have difficulty to connect game elements with programming concepts that I intended to teach. But it was in my mind to turn the activity into a small game.

Another important characteristic of CS-Unplugged activities is the existence of physical objects, which are either easy-to-access or easy-to-prepare. In this study, this dimension was obviously achieved within the activities. Watches, chronometers, magnets, little game figures, cardboards, crayons with different colors, and ruler can be listed as the ones easily accessed (see Table 9). The activities prepared by participants include the utilization of worksheets, game cards, and boxes as accessible objects. In addition, all activities about material preparation were supported by explanations besides visuals.

Participants were supposed to explain why they used the objects listed in Table 9. Except for one participant, they stated that being inexpensive and easy to access was their first criteria to choose the physical object for the activity. Additionally, nine participants emphasized that the materials should be easy to prepare, too. The importance of reusability of the materials and prepared objects was expressed by Interviewee-3. One participant (Interviewee-1) stated "preparing attractive materials was the first thing that I consider".

Having students explore their self-potential in different subjects/fields is another characteristic of well-designed CS-Unplugged activities. In other words, the activities should allow self-management/self-control of students either fully or partially. The role of students in the activities was summarized in Table 10. More than half of the activities show no patterns of student-directedness. Instead, the students were supposed to be in charge of instructions and directions, whereas, the teacher is the one who directs the activities. However, in 15 activities the students located at the center of activities. For

Table 9 Use of physical objects

Objects	*f*
Work sheets/papers	19
Cardboards	25
Watches/chronometers	2
Boxes	1
Play cards	11
Magnets/Stickers	3
Figures	10
Color pens	14
Rulers	2

Table 10 Student directedness

Rationale	*f*
Yes	21
No	15

example, in activity 3, the exploration of variable types was assigned as a student task. In activity 22, they were expected to develop decision structures based on daily conditions. Interviews demonstrated that some of the students might have misconceptions about student-directedness. Seven of the students participated in the interview sessions prepared activities not including the patterns of student-directedness. On the other hand, some of them ($N = 4$) thought that their activities are student-directed.

Easy implementation of activities is another key characteristic. In this study, a considerable amount of prepared activities meets this dimension ($N = 30$) (see Table 11). It was observed that the provided details of implementation processes can enable the teachers to practice within expected time and facilities. However, 7 activities failed to ensure this characteristic. In one activity, the preparation time was too long, and the workload was too much. In another activity, the issue about classroom management was neglected. It was also found that the match between expected and practiced time to complete the activity was not realistic. Apart from these issues, the expected prerequisite knowledge of students was not appropriate at all, because in 4 activities, the activity settings demanded students with higher cognitive abilities than the real target learners usually have.

Interviews also showed that students found their activities easy to implement. None of the interviewees found his/her activity difficult for a computer science teacher to understand, to prepare the materials of and to apply the activities in the classroom. On the other hand, interviewee-2 and interviewee-13 pointed out another issue, which is the transfer of learning as a result of CS-Unplugged activity into real coding environment. The teacher might face with difficulty to establish the connection between programing concepts in these two different environments. Interviewee-13 remarked the following except;

> *In order to create a successful activity, we need a plan. We should define how students would use what learned in the computer free activities in real coding environments. I think without that plan, the success of the activity becomes controversial*

Table 11 Easiness in implementation

Easiness in implementation	*f*
Easy implementation	30
Excessive workload for preparation	1
Necessity for too much background knowledge	4
Classroom management issues	1
Time issues	1

Table 12 Growing body of ideas

Idea	*f*
No growing body of ideas	24
Adaptation to different learning environments and level of students	11
Connection to other programing concepts	3

The statement of ideas in activities should lead to the emergence of new ideas both for students and teachers. The activities were all examined for the patterns of promising ideas that highlight for further improvements. The results indicate that 24 activities do not have any patterns of growing body of ideas. Nevertheless, a few activities ($N = 11$) were found as open for improvements of students from various levels and characteristics. Three activities, on the other hand, have potential to enable students to discover new concepts as well as to let teachers link upcoming topics. Table 12 summarizes the findings about the growing body of ideas. The majority of the activities did not include any sign of growing body of ideas, whereas interviews pointed out slightly different situation. More than half of the interviewees ($N = 9$) expressed that their activities can be adapted to various learning environments as well as to different level of students.

The integration of stories into CS-Unplugged activities may be a good strategy for increasing both learners' motivation and the level of engagement. The activities did not fulfill this expectation (see Table 13). The processes were shaped around the stories only in three activities, one of which constructed upon daily routines. The other two activities referred to escaping from a situation or a place. The activities of the interviewees do not consist of any story-based structure. On the other hand, they agreed that designing their activities around a story would yield better results. Most of the interviewees ($N = 10$) expressed that the inclusion of stories into activities could make them more attractive besides enhancing retention.

5 Conclusion and discussion

This study examined the design processes of CS-Unplugged activities of pre-service teachers. The findings simply revealed that pre-service computer science teachers had limited competencies to prepare their own CS-Unplugged activities. In the literature, general structures of CS-Unplugged activities demonstrate constructivist features from the pedagogical perspective. Since pedagogical courses in computer science teacher training programs mostly include constructivist contents, it may be assumed that participants were skilled enough to prepare successful CS-Unplugged activities. Nevertheless, the results were not in line with this assumption.

Table 13 Sense of story

Story elements	*f*
No sense of story	34
Daily routines	1
Finding the way	2

First of all, having a deeper look at the results, it can be observed that the definitions made in terms of activities' contexts are inadequate. The only provided information was about age or grade level of the students. Other contextual information, which is necessary for correct implementation of activities, was not included in most of the activities. Interviews supported this result because the participants failed to define any contextual issue affecting the flow of activities. The contextual details are not only important for such activities, but also any other types of instructional activities. That' s why, it could be a next step for further investigation whether this situation is similar in the process of designing different kinds of activities.

It was observed that deficiencies in the inclusion of contextual information are similar in defining the forces that may affect the flow of activity. Moreover, the environment in which the activity will be implemented was not explained in detail. It was also found that the conditions (or problems) that the instructors may face during the implementation of activities were not sufficiently determined by the participants. As a matter of fact, the participants focused on situations related with timing of activities. The number of students, their background-knowledge, and situations that may arise, and classroom environment were also mentioned partially. However, it is clearly observed that the points, which have the potential to affect the design process, were overlooked in the instructional design process. Actually, interviews did not point out details about whether participants give enough credit to identify the relationship between context and success of an activity. The lack of real teaching experience might hinder them to understand the connections, thus observing the effects of context on any teaching activity can remain blur. On the other hand, the nature of CS-Unplugged activities could be another reason of the deficiency in defining context of teaching environment.

Although the designed activities include problems in terms of contextual environment and effective forces, the programming skills they set as the main goal of the activities spread over a wider range. Almost all of the concepts of introductory programming education were available through participants' CS-Unplugged activities. They also targeted different programming skills, which are not touched by the literature or public CS-Unplugged activities. Arrays and functions can be among the examples of this situation. In short, it can be inferred that participants of this study can exhibit an entrepreneurial attitude in the name of designing their own activities when they need activities in computer free formats.

It was found that the information about the resulting context also suffers from necessary details within the designed activities. The number of events offering certain definitions for resulting context is much greater than the ones not offering them. The main reason why participants provided limited information about context, forces, and resulting context might be related with the over-focusing on mechanisms of activities and thus ignoring some of fundamental details. For example, the majority of the participants clearly explained what the students should do in a step-by-step manner. Analysis of different aspects of instruction ranging from learners to context is among basics of almost all instructional design models. However, participants of this study generally ignored the analysis process. In other words, learner analysis, content analysis, need analysis and similar steps are missing nearly in all of the activities. The information related with the students' background knowledge, attitudes, and perceptions are crucial for CSE. According to Cassel et al. (2007), prejudice and negative attitudes of students constitute a barrier in front of success in programming education.

In addition, comfort levels (Bergin and Reilly 2005), self-efficacy beliefs (Askar and Davenport 2009), learning styles and problem-solving skills (Gomes and Mendes 2007) can be named as other issues, which have influence on the success level. As a result, computer science teacher training programs might need to enable pre-service teachers to put more attention on analysis phases of CS-Unplugged activities. Participants also have deficiencies in defining the objectives targeting development of particular programming skills. This makes it harder to evaluate students' improvements at the end of any CS-Unplugged activities. Although they have trained enough both to state objectives and to assess and evaluate learning, they need more practice while transferring that knowledge and skills into CS-Unplugged activities.

Furthermore, the majority of the activities designed by the students showed patterns of student directedness. They generally propose to distribute the control of classroom among teachers and students. They also led active participation of students. Student participation and students' direct access to activity management are important signs of student-centered activities. Pre-service teachers who were interviewed after they prepared CS-Unplugged activities demonstrated the signs of misconceptions in terms of student directedness. Such kind of activities may have specific challenges. Management of the event, balancing the control between the teacher and the student, and ensuring the active participation are some of the important points in the design phase. Therefore, there are many points that may cause difficulty in the implementation of the CS-Unplugged activities. The possibility of making mistakes to create student-directed environment is another important point because of the misconceptions about student directedness mentioned before could be decisive on the success.

On the other hand, when the activities were examined, it was determined that this situation was mostly overcome and there were not many challenging points in activities in terms of applicability. As a matter of fact, the given detailed steps of activities in the implementation phase are in parallel with this point. The shortcomings of contextual and effective forces show that the participants' focuses were more on the implementation steps of activities and the other elements were not cared enough. According to Bell (2014), a CS-Unplugged activity must also provide students with the information discovered at the end as well as including the details of how to conduct the activity. Participants' activities need to be improved in this perspective.

It has been found that a significant number of CS-Unplugged activities do not offer environments for growing body of ideas. It is important because a CS-Unplugged activity should not only focus on how computers work. Bell et al. (2008) expressed that they focus on utilization of computer science concepts to reach solution of any problem. In parallel with the previous results, it can be inferred that the focus was on the steps necessary for the implementation of the activity, so that the activities were limited in this sense. However, the fact that some activities can be organized according to different learning environments and different student profiles can be mentioned as positive aspects of activities. Although there are different positive and negative aspects, it can be deduced that activities were poor in terms of making connections with real life and gaining skills for modeling everyday problems. There were no story elements in most of the activities and participants did not mention any intention to establish story structure. Story elements are important especially for motivation. There are several studies indicating the positive contribution of CS-Unplugged activities to the motivation of students (Jiang and Wong 2017, 2018; Mano et al. 2010). Therefore, the activities designed by participants may suffer from gaining students' attention or keeping them motivated. In addition, several activities provided limited amount of evidence for connection to real life situations. For that reason, the level of competence of pre-service computer science teachers should be questioned in this sense. This is because the main objective of computer science courses at this level is to make meaningful contributions to students' cognitive skills rather than training them as software developers.

In the light of all the findings, it can be concluded that the pre-service teachers participated in this study experienced some difficulties while developing CS-Unplugged activities. Among all reasons about this situation, especially the content of teacher training programs appears on the stage. It is important to note that the design and implementation of CS-Unplugged activities must be a part of teaching method courses of computer science teacher training programs. Such a change could lead positive improvements in two fundamental ways. First of all, pre-service teachers can graduate with a new competence that is parallel to twenty-first century skills. The ability of each teacher to design their own CS-Unplugged activities or to organize the pre-designed activities according to their needs could positively affect the success in the long term. The other positive result would be the pedagogical improvement of the activities in a broader range. Since computer scientists have conducted considerable amount of the existing literature, it seems that pedagogical and instructional design issues within CSE may remain intact, which may lead vulnerabilities in the practical dimension. In addition, the model presented by Nishida et al. (2009), which is at the center of this study, needs a revision in terms of instructional design principles. In the future, computer scientists and computer science educators would develop a new integrative framework that has a focus for each pedagogical and technical sides of CS-Unplugged activities.

References

Alamer, R. A., Al-Doweesh, W. A., Al-Khalifa, H. S., & Al-Razgan, M. S. (2015). Programming unplugged: Bridging CS-unplugged activities gap for learning key programming concepts. In *e-Learning (econf), 2015 Fifth International Conference on* (pp. 97-103). IEEE.

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java programming among engineering students. *Turkish Online Journal of Educational Technology, 8*(1), 26–32.

Bell, T. (2014). Ubiquity symposium: The science in computer science: Unplugging computer science to find the science. *Ubiquity, 2014*(March), 3.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2008). Computer science without computers: New outreach methods from old tricks. In Proceedings of the 21st annual conference of the National Advisory Committee on computing qualifications.

Bergin, S., & Reilly, R. (2005). Programming: factors that influence success. In *ACM SIGCSE Bulletin* (Vol. 37, No. 1, pp. 411–415). ACM.

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. In Proceedings of the 12th workshop on primary and secondary computing education (pp. 65–72). ACM.

Burke, Q., O'Byrne, W. I., & Kafai, Y. B. (2016). Computational participation: Understanding coding as an extension of literacy instruction. *Journal of Adolescent & Adult Literacy, 59*(4), 371–375.

Cassel, L., McGettrick, A., Guzdial, M., & Roberts, E. (2007). The current crisis in computing: What are the real issues? In J. Dougherty & S. Rodger (Eds.), *38th SIGCSE technical symposium on computer science education* (pp. 329–330). New York: ACM Press.

Coffman-Wolph, S., Gray, K., & Pool, M. (2018). Work-in-Progress: Research plan for introducing problem solving skills through activities to an introductory computer science course.

Creswell, J. W. (2014). *Research design: Qualitative, quantitative and mixed methods approaches* (4th ed.). Thousand Oaks: Sage.

Demšar, I., & Demšar, J. (2015). CS-unplugged: Experiences and extensions. In *International conference on informatics in schools: Situation, evolution, and perspectives* (pp. 106–117). Cham: Springer.

Deperlioğlu, Ö., & Köse, U. (2010). Web 2.0 teknolojilerinin eğitim üzerindeki etkileri ve örnek bir öğrenme yaşantısı. *Akademik Bilişim*, 10–12.

Giordano, D., & Maiorana, F. (2014). Use of cutting edge educational tools for an initial programming course. In *Global engineering education conference (EDUCON), 2014 IEEE* (pp. 556–563). IEEE.

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. In *International conference on engineering education–ICEE* (Vol. 2007).

Hazzan, O., Lapidot, T., & Ragonis, N. (2014). Teaching methods in computer science education. In *Guide to teaching computer science* (pp. 105–135). London: Springer.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. In Frontiers in education conference (FIE), 2016 IEEE (pp. 1–9). IEEE.

ISTE (2011). Operational definition for computational thinking. Retrieved October 15, 2018 from: http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf.

ISTE (2016). ISTE standards for students. Retrieved October 15, 2018from http://www.iste.org/docs/Standards-Resources/iste-standards_students-2016_one-sheet_final.pdf?sfvrsn=0.23432948779836327.

Jiang, S., & Wong, G. K. (2017). Assessing primary school students' intrinsic motivation of computational thinking. In *Teaching, Assessment, and Learning for Engineering (TALE)*, 2017 IEEE 6th international conference on (pp. 469–474). IEEE.

Jiang, S., & Wong, G. K. (2018). Are children more motivated with plugged or unplugged approach to computational thinking? In Proceedings of the 49th ACM technical symposium on computer science education (pp. 1094–1094). ACM.

Johnson, R. B., & Christensen, L. B. (2004). *Educational research: Quantitative, qualitative, and mixed approaches*. Boston: Allyn and Bacon.

Jonassen, D. H. (2006). *Modeling with technology: Mindtools for conceptual change*. Upper Saddle River: Pearson Merrill Prentice Hall.

Kelleher, C., Pausch, R., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 1455–1464). ACM.

Liu, X., & Wu, D. (2018). From natural language to programming language. In *Innovative methods, user-friendly tools, coding, and design approaches in people-oriented programming* (pp. 110–130). IGI Global.

Mano, C., Allan, V., & Cooley, D. (2010). Effective in-class activities for middle school outreach programs. In *Frontiers in Education Conference (FIE)*, 2010, IEEE.

Mouza, C., Pollock, L., Pusecker, K., Guidry, K., Yeh, C. Y., Atlas, J., & Harvey, T. (2016). Implementation and outcomes of a three-pronged approach to professional development for CS principles. In Proceedings of the 47th ACM technical symposium on computing science education (pp. 66–71). ACM.

Nishida, T., Kanemune, S., Namiki, M., Idosaka, Y., Bell, T., & Kuno, Y. (2009). A CS unplugged design pattern. In G. Lewandowski & S. Wolfman (Eds.), *Proceedings of the 40th SIGCSE technical symposium on Computer Science Education*. Chattanooga: ACM, New York.

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies, 22*(2), 469–495.

Swacha, J., & Baszuro, P. (2013). Gamification-based e-learning platform for computer programming education. In X world conference on computers in education (pp. 122–130).

Thompson, D., & Bell, T. (2015). Virtually unplugged: Rich data capture to evaluate CS pedagogy in 3D virtual worlds. In *2015 International Conference on Learning and Teaching in Computing and Engineering (LaTiCE)* (pp. 156–163). IEEE.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wohl, B., Porter, B., & Clinch, S. (2015). Teaching computer science to 5-7 year-olds: An initial study with scratch, Cubelets and unplugged computing. In Proceedings of the workshop in primary and secondary computing education (pp. 55–60). ACM.