Higher Education Studies; Vol. 6, No. 4; 2016
ISSN 1925-4741 E-ISSN 1925-475X
Published by Canadian Center of Science and Education

ICT Teachers' Acceptance of "Scratch" as Algorithm Visualization Software

Fatih Saltan[1] & Mehmet Kara[1]

[1] Department of Computer Education & Instructional Technology, Amasya University, Amasya, Turkey

Correspondence: Fatih Saltan, Department of Computer Education & Instructional Technology, Amasya University, Amasya, Turkey. E-mail: fsaltan@gmail.com

Received: October 19, 2016 Accepted: October 28, 2016 Online Published: November 22, 2016
doi:10.5539/hes.v6n4p146 URL: http://dx.doi.org/10.5539/hes.v6n4p146

Abstract

This study aims to investigate the acceptance of ICT teachers pertaining to the use of Scratch as an Algorithm Visualization (AV) software in terms of perceived ease of use and perceived usefulness. An embedded mixed method research design was used in the study, in which qualitative data were embedded in quantitative ones and used to explain the results. The data were collected from 214 pre-service ICT teachers studying in four large public universities. Data was gathered through a questionnaire adapted from David's Technology Acceptance Survey (1989) and through open-ended questions. T-test and Pearson correlation, as well as descriptive statistics, were used to analyze quantitative data and constant analysis techniques were used to analyze qualitative data. Both kinds of data were mixed and are presented in the results section. The results show that pre-service ICT teachers mainly have positive and similar Scratch acceptance scores in terms of usefulness and ease of use. The factors explaining participants' perceived usefulness are identified as visual interface (37%), pedagogy(36%), and computational thinking (27%). The majority of the participants also found Scratch to be easy to use. Pre-service ICT teachers explained that what makes AV software easy to use is color separation (40%), drag and drop (30%), and familiar interface (30%). Additionally, no significant difference between the acceptance scores of the participants was found in terms of gender, years of programming experience, programming background, and the high school they graduated from as indicators of programming experience. Results congruent with previous studies regarding Scratch were found by the current study.

Keywords: Algorithm Visualization, ICT teachers' perceptions, computer programming

1. Introduction

Teaching programming to students of all grade levels has gained importance, especially with the introduction of the term called Computational Thinking. Wing (2006) first introduced this concept as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33) with an emphasis on computer science. Cuny, Sinder, and Wing (2010) redefined the concept as "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". Based on the latter, computational thinking has clearly become an interdisciplinary concept based on, but not limited to computer science. Today, this concept is accepted as a 21th century skill by the scholars of education and a required skill for students of all grade levels. Obviously, this definition is based on computer science, particularly programming knowledge and skills. This means that acquisition of this skill requires teaching programming to all students, from the elementary to the university level. Acquisition of computational thinking skills is certainly not a concern restricted to educators and researchers; it is also a concern for countries worldwide, since they have begun to realize the importance of Information and Communication Technologies (ICT) instruction for technological and economic development (Wilson, Sudol, Stephenson, & Stehlik, 2010). Therefore, increasing attention is being given to computational thinking and related research topics (Grover & Pea, 2013), including the use of algorithms as the central element of computational thinking (Cortina, 2007).

However, programming instruction is not so straightforward, even for undergraduate students studying computer-related disciplines. There are lots of problems documented by researchers which can cause novice programming students to fail and dropout of the programming course (Federici, 2011). Some of the identified challenges of teaching programming to university students are lack of understanding of the larger elements, abstract concepts, application of what is learnt, and lack of practical and concrete learning situations (Lahtinen, Ala-Mutka, & Jarvinen, 2005). In a recently conducted study with the participation of students in a computer-related discipline, these challenges were reported as "programming knowledge, programming skills, understanding semantics of the program, and debugging" (Özmen & Altun, 2014). This study also underlines that lack of practice and lack of algorithm usage, along with lack of knowledge, are major problems causing student failure in programming courses.

Computer science educators and researchers have been striving to find pedagogical and technological ways to make the programming instruction process easier and more effective. One of these proposed solutions is to use Algorithm Visualization (AV), since many of the aforementioned problems in learning how to program stem from a faulty grasp of practical usage of algorithms. AV is defined as the graphical illustration of algorithms via software developed for this purpose (Hundhausen, 2002), which aims to facilitate student understanding of the way computer algorithms function (Haundhausen & Brown, 2008). There are many AV software programs aiming to assist novice programmers in learning programming concepts in a visual way, such as Scratch, Alice, Android Appinventor, Scriptease, Kodu, and so forth. These programs have gained popularity in teaching programming, especially as they provide novice programmers with the opportunity to focus more on design and development rather than on programming syntax (Grover & Pea, 2013); decrease the cognitive load through the avoidance of the handling of syntax errors (Kelleher & Pausch, 2005, p. 131); and a fun and comfortable learning context (Kelleher, Pausch, & Kiesler, 2007). In this respect, according to Brennan and Resnick (2012), these AV software programs help learners develop computational thinking skills through design and development of interactive media. Considering all these advantages, AV software is an innovative technology for teaching programming to novice programmers and young students.

Of all these AV software programs, Scratch was chosen for this study due to several reasons. Scratch is a popular AV software, which was particularly designed to teach young students programming (Maloney, Resnick, Silverman, & Eastmond, 2010). First of all, it includes more programming concepts than other AV software programs and it has a context supporting active learning (Koorsse, Cilliers, & Calitz, 2014). Secondly, in their evaluation of some visual algorithm software, Koorsse, Cilliers, and Calitz (2014) list the characteristics of Scratch as "Assists with developing knowledge of programming principles and concepts", "Constructivist to promote self-study", "Assists with the understanding of code execution", "Develops code comprehension", "Feedback to guide solution creation", and, most importantly, "Promotes problem solving and planning". Among these characteristics, problem solving and planning is a crucial one for the development of computational thinking skills. In the same vein, Brennan and Resnick (2012) reported seven computational thinking concepts that are used in Scratch projects which can be applied in programming or other disciplines. These concepts are "sequences", "loops", "parallelism", "events", "conditionals", "operators", and "data". Thus, due to its technological and pedagogical features, Scratch was chosen as the AV software for the current study.

The studies pertaining to the use of Scratch generally indicate improvements in teaching programming in spite of some drawbacks. The research studies revealed that Scratch is useful for helping students use computational constructs (Meerbaum-Salant, Armoni, & Ben-Ari, 2013); engage in programming processes (Resnick et al., 2009); acquire programming skills and motivation (Begosso & Silva, 2013), develop positive attitudes toward programming (Genç & Karakuş, 2012); gain experience working with advanced programming languages (Wolz, Leitner, Malan, & Maloney, 2009); and make reflections on their daily experiences (e.g., mathematical experiences) (Ke, 2014). However, there are still drawbacks to the use of Scratch for teaching programming. For example, Koorsse, Cilliers, and Calitz (2014) reported that although students perceive Scratch to be as useful as Robomind and B#, there was no significant evidence indicating that the students using Scratch or other AV software achieved better results in programming. After reviewing 22 experimental evaluations, this significant difference issue was reported and criticized by Hundhausen (2002) as the drawback of AV technology from the pedagogical standpoint. In another study, Meerbaum-Salant et al. (2013) found out that students still have difficulties with learning some concepts in spite of the use of Scratch. The final drawback of Scratch or other visual algorithm software programs is that they are generally used in extracurricular activities, such as summer campsor computer clubs (Meerbaum-Salant et al., 2013; Wolz, Maloney, & Pulimood, 2008). To conclude, the literature on Scratch usage reveals its numerous advantages, as well as the pedagogical or technological problems waiting to be solved to maximize its effectiveness for educational purposes.

It is obvious that the use of AV software (Scratch in particular) for teaching programming is a relatively innovative method and there are some problems regarding their integration into programming instruction. The source of the problems might be the pedagogical use of these software programs, as well as their technological features. From the pedagogical standpoint, the problems with the use of AV can be overcome through teacher training and subsequent teaching and support (Howland & Good, 2015; Robertson, Macvean, & Howland, 2013; Meerbaum-Salant et al., 2013). In this regard, the recommendations for the solution of the current problems point out teacher training. For example, for the learning problems especially experienced by young learners, Meerbaum-Salant et al. (2013) suggest "careful teaching, close, and effective mentoring". According to Robertson et al. (2013), the long term achievement of innovative technologies, including AV, relies on the degree to which teachers accepts them for use in their practices. In the same vein, Cordova, Eaton, and Taylor (2011) emphasized the key role of teacher training for the use of ALICE to teach programming. In addition, considering the difficulties of teaching programming at the university level, even in computer-related disciplines, it is a requisite to use AV for teaching novice programmers. In this regard, it is important to investigate preservice ICT teachers' acceptance of AV (Scratch in this case) for the successful adoption and integration of these technologies, both in universities and K12 schools.

1.1 Purpose of the Study

Although AV software's visual properties are motivating and encouraging, particularly for novice programmers, there is no comprehensive study on teachers' acceptance of AV software. The purpose of this study is to investigate pre-service ICT teachers' acceptance of Scratch as an AV software in terms of perceived ease of use and perceived usefulness. Specifically, the research questions of this study are as follows:

1) To what extent do ICT teachers accept usage of AV software for learning programming?

2) Is there a difference in acceptance of AV software related to gender, programming experience, programming background, and the high school ICT teachers graduated from?

2. Method

2.1 Research Design

The embedded mixed methods research design was used in this study. According to Creswell et al. (2007), in this design, one type of data can provide support to the other ones. It is especially useful when qualitative data are embedded within quantitative data. In the current study, the research data were mainly collected in quantitative form to determine participants' acceptance of Scratch. The qualitative data embedded in the results were collected to determine and explain the reasons why participants accept or do not accept Scratch.

2.2 Participants

Participants in the study were undergraduate students studying in the Computer Education and Instructional Technology (CEIT) departments off our large, public universities, which are in charge of training ICT teachers in Turkey. All pre-service teachers voluntarily participated in the study. Most of the participants (71%) graduated from vocational school. 48.6% of them were female (N=104) and 51.4% of them (N=110) were male. Pre-service teachers were somewhat experienced in using AV. All of them stated that they had used an AV program like Scratch within the scope of a course.

2.3 Instruments

Data was gathered through a questionnaire adapted from David's technology acceptance survey (1989) and open-ended questions. Participants were asked four demographic questions, five open-ended questions, and 18 likert-type questions, on which 1 means strongly disagree and 5 means strongly agree. David's technology acceptance survey (1989) consisted of two factors: perceived usefulness and perceived ease of use. The validity of the instrument was provided in its development study. As for the reliability, Cronbach Alpha coefficients were calculated to check the internal consistency of the scale. The Coefficient Alpha for Usefulness was obtained as .960, which indicates excellent reliability, the Coefficient Alpha for Ease of Use was obtained as .804, which indicates good reliability, and the Coefficient Alpha for the overall scale was obtained as .931, which indicates excellent reliability (see Table 1). The obtained values indicate that the internal consistency of the scale and sub-scales are satisfactory. Open-ended questions were developed based on the related literature and revised based on the reviews of subject field experts.

Table 1. Internal consistency of the scale

Scale	Number of Items	Coefficient Alpha
Perceived Usefulness	9	.960
Perceived Ease of Use	9	.804
Overall Scale	18	.931

2.4 Data Collection Procedure

Before the data collection phase, students studying in the CEIT departments of four public universities were invited to voluntarily participate in Scratch training. Then, Scratch was introduced to the volunteer participants for four weeks and, with the mentoring of the instructors, they were individually assigned to develop Scratch projects within this timeframe. The instructors provided feedback about their work during the project development process. Upon completion of their individual Scratch projects, all participants were asked to voluntarily answer the questionnaire items. They were also asked to answer the open-ended questions on the questionnaire, in order to collect qualitative data.

2.5 Data Analysis

Based on the research questions, descriptive and inferential statistics and constant analysis techniques were conducted. The quantitative data were analyzed descriptively and presented as means, standard deviations, percentiles, and frequencies. As for the inferential statistics analyses, t-test and Pearson correlation analyses were conducted to determine the mean differences between the groups and the correlations, respectively.

The qualitative data was analyzed using content analysis techniques (Patton, 2002; Miles & Huberman, 1994). Answers to the open ended questions were coded. Emerging codes were categorized and described with their percentage and frequency. In order to ensure internal validity, emerging categories were verified by two different researchers. Inter-coder reliability was calculated as 73 percent. The qualitative data were embedded within the quantitative data and mixed in the results part of the study.

3. Results

The results are presented under the research questions based on the qualitative and quantitative analysis.

3.1 To What Extent Do ICT Teachers Accept Usage of AV Software for Learning Programming?

The participants were asked to respond to 18 items with answers ranging from 1 to 5 on a Likert-type scale which evaluate two factors of the TAM model; namely, "*Perceived usefulness*" and "*Perceived ease of use*". The mean scores of the items ranged from 3.16 to 3.91. Item 16, which is a reverse item, was transformed for the analysis: "*Interaction via Scratch requires too much mental effort*", had the lowest mean score (*M*=3.16, *SD*=1.06). Item 10, which is "*It is easy to learn how to use Scratch*", had the highest mean score (*M*=3.91, *SD*=1.03). Overall, the "*Perceived usefulness*" factor had a mean score of 3.69 (*SD*=.98) and the "*Perceived ease of use*" had a mean score of 3.22 (*SD*=.47). These results show that participants mainly have positive and similar acceptance ratings for the items and the factors in the scale.

On the other hand, qualitative analysis indicated that most of the participants think that AV software is useful in learning computer programming. Their explanations of why they found it useful fell under three factors: visual interface (37%), pedagogy (36%), and computational thinking (27%) (see Figure 1). Participants mostly perceived AVs to be useful because they offers a powerful graphical interface to users. For example, one participant stated: "Of course, it (Scratch) facilitates learning because it saves programming logic from being boring by making it visual".

Also, some participants said AV software utilizes various pedagogical methods based on user needs. They indicated that in the traditional approach, every student completes the same programming activities, which have equal difficulties and numbers, but AVs provide an individualized learning environment. In addition, some of the participant indicated that AV improves users' computational thinking ability, especially problem solving and designing systems by utilizing the fundamentals of computer programming. In this regard, one ICT teacher said, "I had major problems with learning programming. After being introduced to Scratch, I have begun to create algorithms of the works to be done with ease and I now find programming more pleasurable".

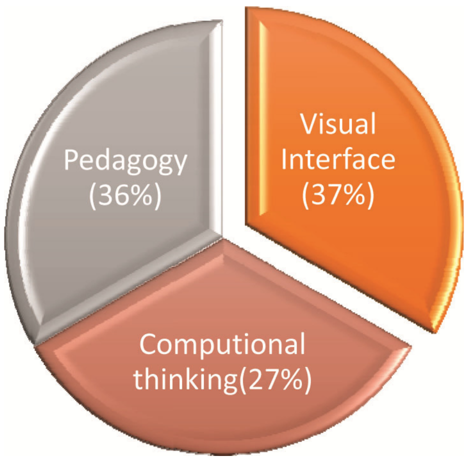

Figure 1. Categories explaining perceived usefulness

Qualitative analysis was also conducted to understand what participants think about ease of use of AV software. Results indicated that all but nine participants perceived AV software to be easy to use. Fifteen participants asserted that it should be easier to use. Eleven participants also said that it is easy to use, but users need a bit of practice. The resulting categories are listed in Table 2. For example, one participant indicated:

“The interface and the usage of the functions are quite easy and understandable. But, continual practice with different examples is needed to improve skills. In this way, students can progress their learning by comprehending the fundamentals and logic of programming”.

Table 2. Perceived ease of use

Categories	Frequency (N=128)
Yes, ease of use	170
Yes, but can be better	15
Yes, but need practice	11
No, not ease of use	9

Moreover, ICT teachers explained what makes AV software easy to use under three categories: color separation (40%), drag and drop (30%), and familiar interface (30%). In the AV program, different colors are used for code blocks that have different functions. Most ICT teachers indicated that that was what made AVs easy to use. It was also mentioned that users are able to drag and drop code blocks instead of writing them. In this regard, one participant said, “I find it easy to use. Since it is drag-drop, users are much more motivated in a visual environment by using code blocks representing codes”.

Moreover, ICT teachers indicated that the interface of the AV software is similar to the interface of common computer programs. Therefore, participants do not have difficulty using AV to learn computer programming.

3.2 Is There a Aifference in the Acceptance of AV Software in Terms of Gender, Programming Experience, Programming Background, and the High School ICT Teachers Graduated from?

An independent samples t-test was conducted to test if there is a difference between the mean scores of the participants’ technology acceptance in terms of gender. The factors in the scale were adopted as the dependent variables. The results obtained are shown in Table 3.

Table 3. Independent samples t-test results for acceptance and gender

	t	*df*	Sig. (2-tailed)	Mean Difference	Effect Size (Cohen's *d*)
Perceived Usefulness	.033	210	.974	.004	.004
Perceived Ease of Use	1.041	210	.299	.067	.145

According to Table 3, the mean score of the male participants is higher than that of the female participants, with a mean difference of .004. However, there is no significant mean difference between the perceived usefulness scores of the female (*M*=3.690, *SD*=.978) and male participants (*M*=3.694, *SD*=.986) with a small effect size according to Cohen's (1998) standards; *t* (210)=0.033, *p*>.05, *d*=.005. Table 3 indicates similar results for perceived ease of use. Similarly, there is no significant mean difference between the perceived ease of use scores of female (*M*=3.182, *SD*=.428) and male (M=3.249, SD=.497) participants with a small effect size and mean difference of .067; *t* (210)=1.041, *p*>.05, *d*=.144.

Whether participants' acceptance of AV software differed depending on the high schools they graduated from was also checked through an independent samples t-test. For this purpose, the participants were grouped by whether they graduated from a vocational or non-vocational high schools. The assumption was that the participants who graduated from vocational high schools had programming education background. The independent t-test results are demonstrated in Table 4. The participants who graduated from vocational high schools have higher mean scores than those who graduated from non-vocational high schools, with a mean difference of .180 in terms of perceived usefulness. Table 4, however, shows that there is no significant mean difference between the perceived usefulness scores of the participants who graduated from vocational (*M*=3.728, *SD*=.995) and those who graduated from non-vocational high schools (M=3.549, SD=.922) with a small effect size; *t* (209)=1.105, *p*>.05, *d*=.187.

Table 4. Independent samples t-test results for acceptance and high school

	t	*df*	Sig. (2-tailed)	Mean Difference	Effect Size (Cohen's *d*)
Perceived Usefulness	1.105	209	.270	.180	.187
Perceived Ease of Use	1.206	209	.229	.093	.208

In the same vein, in spite of the mean difference of .093 in favor of the participants who graduated from vocational high schools, there is no significant difference between the perceived ease of use mean scores of those who graduated from vocational (M=3.237, SD=.480) and those who graduated from non-vocational high schools (M=3.144, SD=.412) with a small effect size; *t* (209)=1.206, *p*>.05, *d*=.208.

An independent samples t-test was conducted again to investigate whether the acceptance scores of the participants differ based on their previous higher education background before the ICT teacher training program. In accordance with this purpose, the participants were grouped according to whether they had graduated from a computer-related associate degree program (N=10) or had not (N=202). The independent samples t-test results are indicated in Table 5 below.

Table 5. Independent samples t-test results for acceptance and previous higher education background

	t	*df*	Sig. (2-tailed)	Mean Difference	Effect Size (Cohen's *d*)
Perceived Usefulness	1.167	210	.245	.370	.407
Perceived Ease of Use	.194	210	.846	.029	.069

The participants who previously enrolled in an associate degree program had a higher mean score than those who did not. But, according to Table 5, there is no significant mean difference between the perceived usefulness scores of the participants who previously enrolled in an associate degree program (*M*=4.044, *SD*=.827) and those who did not (*M*=3.674, *SD*=.985) with a small effect size; *t* (210)=1.167, *p*>.05, *d*=.407. A similar result was obtained for the mean scores of perceived ease of use. There is no significant mean difference between the perceived ease of use scores of the participants who previously enrolled in an associate degree program (*M*=3.244, *SD*=.370) and the ones who did not (*M*=3.215, *SD*=.470) with a small effect size; *t* (210)=.194, *p*>.05, *d*=.069.

Finally, Pearson product moment correlation was conducted to investigate whether there is a relationship between participants' programming experience and their acceptance of Scratch in terms of perceived usefulness and perceived ease of use. The participants' responses regarding their programming experience in terms of years and their ratings for the perceived usefulness and perceived ease of use factors were used for the vicariate correlations. The programming experiences of the participants who provided their years of experience with the percentage of 93.87 (N=199) ranged from 1 to 10 years with the mean of 4.05 years. 6.13% of the participants (N=13) did not provide information about their previous programming experience. Table 6 demonstrates the results obtained from Pearson correlation between participants' programming experience and their perceived usefulness and perceived ease of use ratings.

Table 6. Pearson correlation between programming experience and scratch acceptance

		Perceived Usefulness	Perceived Ease of Use
Programming Experience	Pearson Correlation (*r*)	.112	.008
	Sig. (2-tailed)	.116	.916

According to Table 6, there is a positive correlation between programming experience and perceived usefulness. The Pearson correlation coefficient in this case indicates a small correlation according to Cohen's (1988) guidelines. This small positive correlation can be observed in Figure 2. However, there is no significant relationship between participants' programming experience and their responses about perceived usefulness; *r* (199)=.112, *p*>.05.

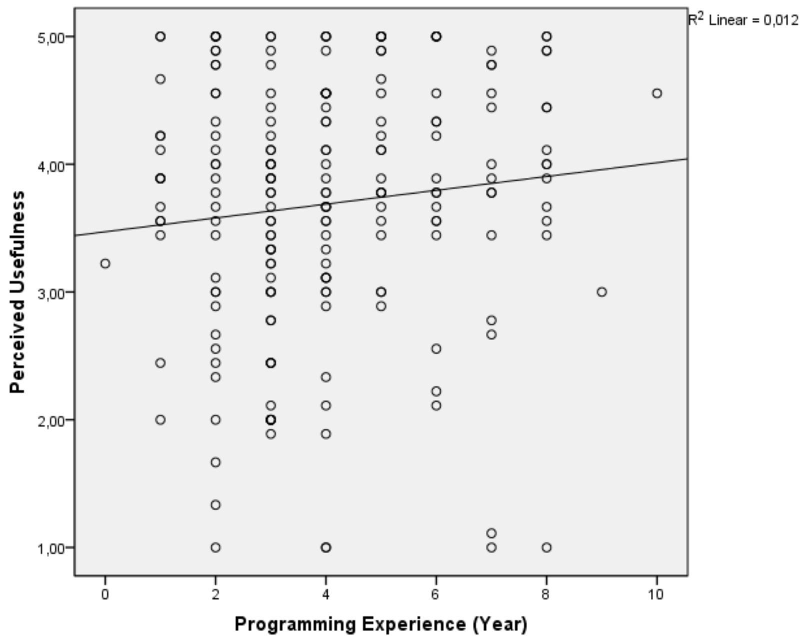

Figure 2. Pearson correlation between programming experience and perceived usefulness

The Pearson correlation analysis results revealed that there is no significant correlation between participants' programming experience and perceived ease of use; *r* (199)=.008, *p*>.05. The Pearson correlation coefficient obtained in the analysis of this case indicates a negligible relationship since it is approximately zero.

4. Discussion

The study results, first of all, revealed that preservice ICT teachers mainly have positive and similar acceptance of Scratch in terms of usefulness and ease of use. The qualitative results provided key information about the reasons why they have positive acceptance of it.

They find Scratch useful due to the visual interface, pedagogy, and computational thinking factors. The first factor, visual interface, provides them with an engaging learning environment rather than a boring and threatening one. According to Grover and Pea (2013), visual interface provides learners with an opportunity to concentrate more on design and creation instead of programming syntax. So, visual interface is a critical element for students' acceptance of Scratch. Another factor underlined by the participants which affects their acceptance is pedagogy. They believe that Scratch is useful for learning and teaching programming because it facilitates the algorithm creation process in an easier and more concrete manner. Creating algorithms for problems, lack of practice and knowledge are reported by Özmen and Altun (2014) as the major problems in learning programming. Additionally, the abstract nature of programming and lack of practical and concrete learning contexts are the reasons for difficulty, for especially novice learners (Lahtinen, Ala-mutka, & Järvinen, 2005). Congruent with these previous studies, the concrete nature of Scratch for algorithm creation is another factor for participants' acceptance. The final factor stated by the participants which explains the usefulness of Scratch is Computational thinking. They stated that Scratch is not only a useful tool for teaching programming, but also useful for helping students gain computational thinking skills since it facilitates the formulation of the encountered problems and create algorithms for their solutions. As Cortina (2007) stated, formulation and use of algorithms are major components of computational thinking. Similar results were obtained in the literature revealing the facilitating role of AV for obtaining computational thinking skills (Grover & Pea, 2013; Lye & Koh, 2014). Therefore, Scratch's facilitating role for learners to gain computational thinking plays a crucial role in its acceptance.

The majority of the participants also found Scratch to be easy to use. They explained why Scratch is easy to use by underlining color separation, drag and drop, and familiar interface features. The results demonstrated that color separation and drag-drop features makes programming understandable and motivating. Similar results were found in the literature that lack of motivation is a problem in learning programming (Lahtinen, Ala-mutka, & Järvinen, 2005) and AV software provides learners with motivation and engagement (Begosso & Da Silva, 2013; Grover & Pea, 2013). The third factor affecting their acceptance in terms of ease of use is familiar interface. Lye and Koh (2014) reported that Scratch has interface features similar to traditional visual programming languages. In conjunction with their conclusion, the results of this study showed that Scratch has interface properties similar to traditional software and this enabled students to master Scratch and consequently have positive acceptance.

In addition, the study sought to determine whether participants' acceptance varies depending on gender, years of programming experience, programming background and the type of high school they graduated from as indicators of programming experience. The results indicated that there is no significant difference between their acceptance scores in terms of gender, programming background, and the high school type they graduated from and there is also no significant correlation between programming experience and their acceptance. These results mean that students have acceptance of Scratch regardless of gender and programming experience, although it was reported in the literature that the challenge of learning programming varies depending on gender (Howland & Good, 2015; Yurdugül & Aşkar, 2013) and experience (Lau & Yuen, 2011; Özmen & Altun, 2014). For this reason, the non-significancy has important implications for programming instruction at the university level, since this could be considered an advantage of Scratch for its widespread adoption by all learners, particularly novice programmers.

5. Conclusion and Implications for Future Studies

The overall results obtained in this study indicate that preservice ICT teachers have positive acceptance of Scratch due to its distinguished features and, more importantly, their acceptance is independent of their gender and programming experience. As Guzman and Nussbaum (2009) stated, teacher training plays a central role in the successful integration of technology in schools. In the same vein, since the acceptance of a technology in all school levels relies on teacher acceptance (Robertson, Macvean, & Howland, 2013), these results suggest that the introduction of Scratch in ICT teacher training programs may encourage its widespread acceptance and integration in schools. Additionally, the results also suggest that its facilitating, motivating, and engaging nature for learning and teaching programming will be helpful for preservice ICT teachers to gain required programming and computational thinking skills.

This study was conducted based on a four-week Scratch training at the university level. Therefore, the focus of future studies must be on the integrated use of AV in the introductory programming courses at the university level. Moreover, as indicated in the literature, AV software is used in K12 settings only within extracurricular activities such as computer clubs and summer camps. For this reason, the focus of future studies must also be on the widespread acceptance and integration of AV in K12 education.

References

Begosso, L. C., & da Silva, P. R. (2013, October). Teaching computer programming: A practical review. In *2013 IEEE Frontiers in Education Conference (FIE)* (pp. 508-510). IEEE. https://dx.doi.org/10.1109/fie.2013.6684875

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.

Cordova, J., Eaton, V., & Taylor, K. (2011). Experiences in computer science wonderland: A success story with Alice. *Journal of Computing Sciences in Colleges*, *26*(5), 16-22.

Cortina, T. (2007). An introduction to computer science for non-majors using principles of computation. *ACM SIGCSE Bulletin*, *39*(1), 222. https://dx.doi.org/10.1145/1227504.1227387

Creswell, J. W., Plano Clark, V. L., Gutmann, M. L., & Hanson, W. E. (2003). Advanced mixed methods research designs. *Handbook of Mixed Methods in Social and Behavioral Research*, 209-240.

Cuny, J., Snyder, L., & Wing, J. (2010). *Demystifying Computational Thinking for Non-Computer Scientists, Work in Progress*.

Federici, S. (2011, October). A minimal, extensible, drag-And-Drop implementation of the C programming language. In *Proceedings of the 2011 conference on information technology education* (pp. 191-196). ACM.

Genç, Z., & Karakuş, S. (2012). Tasarımla Öğrenme: Eğitsel Bilgisayar Oyunları Tasarımında Scratch Kullanımı. In *5th International Computer & Instructional Technologies Symposium*.

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43. https://dx.doi.org/10.3102/0013189X12463051

Guzman, A., & Nussbaum, M. (2009). Teaching competencies for technology integration in the classroom. *Journal of Computer Assisted Learning*, *25*(5), 453-469. https://dx.doi.org/10.1111/j.1365-2729.2009.00322.x

Howland, K., & Good, J. (2015). Learning to communicate computationally with Flip: A bi-modal programming language for game creation. *Computers & Education*, *80*, 224-240. https://dx.doi.org/10.1016/j.compedu.2014.08.014

Hundhausen, C. D. (2002). Integrating algorithm visualization technology into an undergraduate algorithms course: Ethnographic studies of a social constructivist approach. *Computers & Education*, *39*(3), 237-260. https://dx.doi.org/10.1016/S0360-1315(02)00044-1

Hundhausen, C. D., & Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. *Computers & Education*, *50*(1), 301-326. https://dx.doi.org/10.1016/j.compedu.2006.06.002

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education*, *73*, 26-39. https://dx.doi.org/10.1016/j.compedu.2013.12.010

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

Kelleher, C., Pausch, R., & Kiesler, S. (2007, April). 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.

Koorsse, M., Cilliers, C., & Calitz, A. (2015). Programming assistance tools to support the learning of IT programming in South African secondary schools. *Computers & Education*, *82*, 162-178. https://dx.doi.org/10.1016/j.compedu.2014.11.020

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005, June). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14-18. https://dx.doi.org/10.1145/1151954.1067453

Lau, W. W., & Yuen, A. H. (2009). Exploring the effects of gender and learning styles on computer programming performance: Implications for programming pedagogy. *British Journal of Educational Technology*, *40*(4), 696-712. https://dx.doi.org/10.1111/j.1467-8535.2008.00847.x

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. https://dx.doi.org/10.1016/j.chb.2014.09.012

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16. https://dx.doi.org/10.1145/1868358.1868363

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, *23*(3), 239-264. https://dx.doi.org/10.1080/08993408.2013.832022

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Sage.

Özmen, B., & Altun, A. (2014). Undergraduate Students' Experiences in Programming: Difficulties and Obstacles. *Turkish Online Journal of Qualitative Inquiry*, *5*(3). https://dx.doi.org/10.17569/tojqi.20328

Patton, M. Q. (2002). *Oualitative Research-Evaluation Methods* (3th ed.). ThousandOaks, CA: Sage.

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

Robertson, J., Macvean, A., & Howland, K. (2013). Robust evaluation for a maturing field: The train the teacher method. *International Journal of Child-Computer Interaction*, *1*(2), 50-60. https://dx.doi.org/10.1016/j.ijcci.2013.05.001

Wilson, C., Sudol, L., Stephenson, C., & Stehlik, M. (2010). *Running on empty: The failure to teach K-12 Computer Science in the digital age*. Tech. Rep. ACM.

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

Wolz, U., Leitner, H. H., Malan, D. J., & Maloney, J. (2009, March). Starting with scratch in CS 1. *ACM SIGCSE Bulletin*, *41*(1), 2-3. https://dx.doi.org/10.1145/1539024.1508869

Yurdugül, H., & Aşkar, P. (2013). Learning programming, problem solving and gender: A longitudinal study. *Procedia-Social and Behavioral Sciences*, *83*, 605-610. https://dx.doi.org/10.1016/j.sbspro.2013.06.115