International Journal of E-Adoption
Volume 11 • Issue 2 • July-December 2019

The Role of Self-Efficacy and Perceived Enjoyment in Predicting Computer Engineering Students' Continuous Use Intention of Scratch

Ibrahim Arpaci, Gaziosmanpasa University, Tokat, Turkey
https://orcid.org/0000-0001-6513-4569
Pınar Onay Durdu, Kocaeli University, Izmit, Turkey
Alev Mutlu, Kocaeli University, Izmit, Turkey

ABSTRACT

Scratch has been implemented as a preliminary programming environment to improve students' programming experience, motivation, and success in the Computer Science (CS) or Engineering departments. Acceptance and use of this visual programming environment by CS or Engineering students is a significant research area. Accordingly, this study investigated acceptance and use of Scratch programming environment by developing a theoretical model based on the Technology Acceptance Model (TAM). Structural equation modelling approach was used to validate the research model based on data collected from a sample of 186 Computer Engineering students. Results indicated that perceived enjoyment was significantly associated with the perceived usefulness and attitudes. Further, the results suggested that self-efficacy was significantly associated with the ease of use perceptions.

KEYWORDS

Perceived Enjoyment, Scratch, Self-Efficacy, TAM

INTRODUCTION

With the growth of 17%, software related jobs were announced as being among the top 20 highest projected numeric change in employment for the following decade in the U.S. (Bureau of Labor Statistics, 2017). A similar trend can be seen in other countries as well (World Economic Forum, 2016). Therefore, there is a clear need to fill in those positions with new computer scientists who are competent in programming and software development.

Despite the increasing need for graduates in the IT field, retention rates in software related degree programs are reported as a major problem in several studies (Sloan & Troy, 2008; Rizvi, Humphries, Major, Lauzun, & Jones, 2011; Rizvi & Humphries, 2012). For example, an important number of the CS students changes their major after they have taken the initial CS course on programming.

DOI: 10.4018/IJEA.2019070101

High attrition among the CS students can be attributed to various factors such as academic background or gender (Sloan & Troy, 2008). However, one of the important factors is that the computer programming is a "notoriously" difficult and complex activity (De Kereki, 2008). Most of the programming languages "look like Greek" for the novice programmers when they first introduced in introductory programming courses (Malan & Leitner, 2007, p. 223). Students generally attend the first computer-programming course with a limited background in programming concepts and theories. Further, they are expected to learn all these abstract concepts in a short period of time. Engaging the students is critical for meaningful learning and this urged some CS departments to create new methods to attract students' attention (Rizvi et al., 2011).

In order to attract today's non-traditional students' attention, departments should integrate interactive programming environments that support computational thinking of their students such as Alice (Cooper, 2010), Greenfoot (Kölling, 2010) or Scratch (Maloney et al., 2004). These tools were generally considered as gateways to learn programming languages such as C (Wolz, Leitner, Malan & Maloney, 2009), C++ (Rizvi, Humphries, Major, Lauzun & Jones, 2010) or Java (Malan & Leitner, 2007; Wolz et al., 2009; De Kereki, 2008). They were used as a preliminary programming environment to introduce programming concepts mainly semantics rather than syntactical focus in introductory CS courses (Fernández Leiva & Civila Salas, 2013).

Studies in recent years questioned effectiveness of these kinds of visual programming tools on either students' success or their attitudes. Alice and Greenfoot were implemented with small groups in a university and the preliminary results were not as expected De Kereki (2008). The results suggested that these tools were not easy to use and had a high learning curve. In another study, the findings revealed the fact that Scratch had a remarkable potential in higher education as well for the CS introductory programming courses (Malan & Leitner, 2007). A recent study investigated whether Scratch could reduce students' computer programming anxiety or not. The results suggested that using Scratch as a remedial instruction in a C++ programming course could increase the students' learning motivation (Wu, Chang, & We, 2010). Thus, acceptance and use of this programming environment by the CS or engineering students can be considered as a significant research area that needs further investigation.

Although there are various studies on the effect of Scratch on the retention rates and success level of students, there was no study that focused on the CS students and investigated these students' attitudes and intentions toward using Scratch. Therefore, this study focused on the CS students and extended the TAM proposed by Davis (1989) to better explain students' acceptance and use of Scratch. Current findings may help curriculum or program developers in making decisions on the integration of Scratch programming environment in higher education.

LITERATURE REVIEW

Lifelong Kindergarten group at the MIT Media Lab developed and officially distributed Scratch in May 2007 (Resnick et al. 2009; Maloney et al., 2010). Computer programs can be easily developed by using command blocks that are dragged and dropped in the Scratch. This motivates experimentation, prevents syntax problems, and enables focusing only on problem solving or algorithm design (Wolz, Maloney, & Pulimood, 2008). Since it includes many of the programming constructs such as repetition, threads, conditionals, communications, user inputs and variables, Scratch has a potential to introduce most of the basic concepts for CS students. Although it is primarily not object oriented, it does enable use of object concepts by sprites and their associated scripts (Rizvi & Humphries, 2012). Although it was not primarily proposed for college or university level students (Resnick et al., 2009), thanks to the advantageous features, it has the potential to be integrated in those settings.

In the literature, numerous studies focus on integration of Scratch programming environment at both primary and middle school levels (Shin & Park, 2014; Falloon, 2016). A review study reported that Scratch and Logo were two common programming software used in the K12 (kindergarten through the 12th grade) educational settings (Lye & Koh, 2014). However, Scratch was also used in a CS course