Available online at www.sciencedirect.com

ScienceDirect

Computers & Education 51 (2008) 1094–1102

COMPUTERS & EDUCATION

www.elsevier.com/locate/compedu

A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students

Nesrin Özdener *

Marmara University, Atatürk Faculty of Education, Department of Computer and Instructional Technologies, Göztepe, 81040 Istanbul, Turkey

Received 6 March 2007; received in revised form 19 October 2007; accepted 28 October 2007

Abstract

This study focuses on how students in vocational high schools and universities interpret the algorithms in structural computer programming that concerns time-efficiency. The targeted research group consisted of 242 students from two vocational high schools and two departments of the Faculty of Education in Istanbul. This study used qualitative and quantitative research methods together. During the study the students were asked to evaluate the algorithms given to them in terms of their run-time efficiency. The analysis of the data has used descriptive statistics and the results of the independent sample *T*-tests to compare the groups. The findings of this study slightly stress that the students have misconceptions about algorithms. Students in high schools and in computer education and instructional education departments have the same observed misconceptions, while mathematics education students are better at interpreting the misconceptions.

Keywords: Algorithms; Time-efficiency; Misconceptions

1. Introduction

The problems of teaching courses in traditional, cooperative, or vocational education programs are mostly based on the materials and techniques of instruction. Recent research and experimentation has led many of the key stakeholders in education to invest in computer technology developments by preparing individualized instruction, computer-assisted instruction, and so on. This study emphasizes the problems in classroom and laboratory-based computer-programming courses which can basically be solved by analytical and technical approaches.

The millions of people involved in traditional schooling are dissatisfied with the current programs, as the technology outside of the school system changes rapidly while the problems of the traditional education remain. These problems do not depend on manipulating the current technology into the courses. However, most of

* Tel.: +90 216 345 47 05x254; fax: +90 216 338 80 80.
E-mail address: nozdener@marmara.edu.tr

0360-1315/$ - see front matter
doi:10.1016/j.compedu.2007.10.008

the core problems can be handled by the application of a deep understanding of the theories and philosophies of education. Each program or course should be handled individually in order to solve its inherent problems.

Computer education in high schools and universities is a vital and attractive subject which seems to have enjoyed increasing numbers of student enrolments. As in every field, its basic concepts need to be continually strengthened for the discipline to maintain a solid foundation. Introduction to computer-programming courses are the foundation stones that affect the strength of departments of computer science (CS), as all individual courses and programs are interrelated in the same way that a knowledge of calculus in mathematics can help solve problems in a physics class. Similarly, painting courses can have a relationship with physics courses, as they can provide students with an aesthetic understanding. Computer-programming courses should be taught as close to perfectly as possible so all other relevant courses may build on them.

Students' successes with programming languages are based on such prerequisites as the courses taken, their technical skills, and their enthusiasm for programming, factors which are relevant to their cognitive and behavioral accomplishments. Computer programmers need to be skilled at problem solving and critical thinking (Hudak & Anderson, 1990). Because of this, mathematical education is important for computer education, as having mathematical knowledge creates a positive impact on learning computer programming (Ricardo, 1983; Ignatuk, 1986). Additionally, White and Sivitanides (2003) found that mathematics education has effects on structural computer programming and object-based computer programming, and that mathematical knowledge makes a positive contribution to these, while Taylor and Monnfield (1991) found a positive correlation between the level of mathematical knowledge and achievement in structural programming.

High school and university computer-programming courses measure their objectives by how well their students learn the commands in each course's specific computer language. However, courses should also address the methods of solving these problems. Problem solving depends on knowing how to analyze problems and then plan their solutions. Introduction to computer-language courses should teach how to build skills in the algorithmic approach to programming as well as problem analysis and solution (Gilmore, 1990). Novice programmers come up against many mental obstacles in their attempts to understand the functioning of programs or the construction of algorithms. In order to deal with these difficulties, Dagdilelis, Satratzemi, and Evangelidis (2004) developed various didactic scenarios.

In Turkey, the curriculum for high-school computer-programming language courses includes the upper-level computer-programming languages of Pascal, VBasic, and C (Turkish Ministry of National Education, n.d). The universities and the vocational colleges have almost the same sort of programming languages in their courses. The relevant courses mostly devote themselves to the teaching of the commands of each specific programming language. Ginat (2001) stressed that the same approach in programming courses is not only repetitive, but also that they could not teach more than the memorization of the commands basic to the handling of the variations in each programming language.

The design of efficient algorithms is one of the solutions for the problems found in computer-programming languages. Algorithm efficiency is a fundamental computer-science concept encapsulating the core topic of algorithm complexity. While this complexity is expressed in formal measures such as Big O, efficiency can be denoted in more flexible terms (Cormen, Leiserson, & Rivers, 1991). Gal-Ezer and Zur (2004) found that most high-school students have a misconception of the concept of efficiency in algorithms. Realizing the importance of teaching efficiency during the early stage of programs of study in CS and the difficulties encountered when introducing this concept, Ginat (2001) and Gal-Ezer, Vilner, and Zur (2004) advocated a different approach to introductory CS courses.

Determining students' misconceptions and their level of understanding involves the methods of concept mapping, interviews, and graphs. In addition, there is the issue of whether to test students with open-ended or multiple-choice questions. Related research into misconceptions found that the impact of given wrong answers depends to some extent on the nature of the multiple-choice items' distracters (Treagust, 1988; Tan, Goh, Chia, & Treagust, 2002). However, open-ended question are able to obtain a direct vision of students' thinking processes and an understanding of how deeply they interpret the concepts. Open-ended questions allow participants freedom of expression to explain their ideas relating to the topic, and therefore allow for monitoring the use of the particle model in students' explanations (Glazar & Vrtacnik, 1992; Birenbaum & Tatsuoka, 1987; Cassel & Johnstone, 1982; Friel, 1981). In addition, the multiple-choice question format might not be the best for obtaining information about students' rationale behind their answers. In closed-ended questionnaires the nature of the response is