Improving First Computer Programming Experiences: The Case of Adapting a Web-Supported and Well-Structured Problem-Solving Method to a Traditional Course*

Murat Pasa Uysal
Turkish Military Academy, Turkey

Abstract

The introductory computer programming (CP) course has been taught for three decades in the faculty. Besides pursuing CP technology, one major goal has been enhancing learners' problem-solving (PS) skills. However, the current situation has implied that this might not be the case. Therefore, a research was conducted to investigate the effects of a web-supported and well-structured PS instructional method on academic achievements and PS perceptions of learners. This was a quasi-experimental study with a posttest-only design that included a control group. While the web-supported and traditional approach was adopted for the control group, the experimental group was treated with the web-supported and well-structured PS method. A cluster random sampling was used and the existing 18 sections were randomly assigned to the study groups. Consequently, 6 faculty members and 433 freshman undergraduate students participated in the study for one semester. The students' PS perceptions were assessed by the Problem Solving Inventory (PSI) and their CP performances were measured by an academic achievement test. The results indicated a significant difference between the groups in terms of CP achievements. Except for one factor of the PSI, there were also significant differences between the groups in terms of their PS perceptions.

Keywords: *Instructional design; Problem-solving; Web-supported instruction; Computer programming*

Introduction

Researchers have been searching the ways for integrating techniques to classes for overcoming the difficulties in learning or teaching computer programming (CP). Different studies are continually evolving in the hope of finding better tools and methods. Integrated techniques would also benefit students (Jerez, Bueno, Molina, Urda, & Franco, 2012), and thus, they are expected to alleviate some of the problems in learning CP. For example, recent studies indicate that learners' problem-solving (PS) ability, which is also an essential skill for CP, can be facilitated through the integration of PS techniques and computer technologies (Chen, 2010; Hwang, Chen, Tsai, & Tsai, 2011; Kim & Hannafin, 2011; Kuo, Hwang, & Lee, 2012). Therefore, besides pursuing computer technology, one of the major goals has been enhancing the PS skills of learners whether or not this is explicitly stated. However, much of the contemporary CP courses are not necessarily based on an instructional design theory of PS aiming to improve beginners' both PS skills and CP performances.

* A limited part of this study was presented at the International Conference on Future of Education in June 13, 2013, Florence, Italy.

The generalization of CP skills is one of the main troubles for beginners, and it may take quite a long time. Novice learners usually tend to forget programming details if they cannot effectively process and use information in meaningful instructional contexts. It is required for every programmer to acquire effective CP skills and knowledge before engaging with complex programming tasks (Palumbo, 1990). The works on software industry and university students additionally support the idea that learners have to structure their knowledge and they need to internalize CP skills long before attempting to produce qualified software solutions.

The PS techniques, such as top-down design or breakdown of a given programming task, have already been used by instructors (Kay, Barg, Fekete, Greening, Hollands, Kingston, & Crawford, 2000). Expert computer programmers consciously employ these techniques for planning, designing, coding and testing of a programming task, and they explicitly decompose programming problems into sub-problems (Thomas & Upah, 1996). However, novice programmers usually attempt to start programming without handling the task as a whole; and they may not plan or apply solution steps automatically (Jonassen, Strobel, & Lee, 2006). Moreover, low-performing students mostly lack the skills and cognitive strategies required for reflecting on a programming problem, organizing CP domain knowledge, and finally employing different solutions (O'Kelly et al., 2004).

Hence, some researchers attempted to investigate the relationships between CP and PS skills to address the needs of novice learners. For example, Dalton and Goodrum (1991) conducted an experimental study exploring the relationship between PS and CP instruction, their effects on PS skills, attitudes, and learning achievements. The results showed significant learning gains when CP was integrated with PS instruction, but teaching CP alone was not effective. The Allan and Kolesar (1996) study's focus was to predict the success in an introductory CP course. They found that a skill-based and PS approach to computer science had certain advantages for undergraduate learners. In the Lee and Thompson's (1997) study, the lack of cognitive strategies in solving problems was found as a contributing factor for the low performances of CP learners.

The most of research findings indicate that PS skills can be acquired in, and transferred from CP within the context of instructions focusing on the well-defined aspects of PS process (Choi & Repman 1993; Unuakhalu, 2004). These studies, therefore, report the positive effects of programming (Dalton and Goodrum 1991; Lee and Thompson 1997; Palumbo, 1990). Despite this intuitive appeal, there is also conflicting evidence on the relationship between CP and PS (Unuakhalu, 2004). The researchers in this group point out that the transfer of PS skills can occur under certain conditions, and therefore, they cannot support the use of CP as a basic method for teaching or improving PS skills (Liao & Bright 1991; Pea & Kurland 1984). However, all of the studies integrating PS and CP include micro-level problem-solving techniques rather than an instructional design theory.

When it comes to our faculty members and students, the traditional instructional method for the CP course is still preferred by the majority. However, it is away from preparing the students for the challenges of CP and PS in different instructional situations (Mills & Treagust, 2003). Moreover, it cannot enable the knowledge transfer to other learning domains (Dale, Weems, & Headington, 1997; Dunlap, 2005; Jonassen, Strobel, & Lee, 2006). Therefore, should PS strategies complement the CP course; then this may be more effective than the direct presentation of concepts, facts and rules only. Although teaching CP through PS techniques has been a topic for different researchers, there is little empirical evidence exploring the

effects of a CP course, which is based on the prescriptions of an instructional design theory for PS.

When taken together, the literature review and the current situation in the faculty where the present study was conducted suggest that the integration of PS and CP is not an easy job, and therefore, this process has to be grounded on appropriate design principles (Peng, 2010). In view of educational technology, it is thought that adopting an instructional design theory of PS for the CP course can be an affecting factor for improving the learners' PS perceptions while enhancing their performances in a rich and meaningful learning context.

The purpose of this study was to explore the effects of a web-supported and well-structured PS method on academic performances and PS perceptions of novice CP learners. Towards this research goal, the hypotheses were as follows: (a) When compared to traditional instruction, well-structured PS instruction will not have a significantly different effect on academic achievement; and (b) the instructional treatments will not make significant differences between the groups in terms of PS perceptions. The main argument of the study is that a CP course should be based on an instructional design theory for improving the well-structured PS skills of learners. The following sections present the related work, situation in the faculty, method, results and discussion parts of this article.

Problem Solving and Computer Programming

In general, a problem is an unknown resulting from a situation in which an individual seeks to satisfy a need or to accomplish a goal. A problem is problem only when there is a "felt need" that motivates a person to search for a solution (Arlin, 1989). The PS process depends upon the problem solver's personal understanding, the problem representation, and goal-oriented activities for developing a solution to the problem. Problems can be grouped into the categories, such as ill-structured and well-structured. The ill-structured problems are similar to the ones in everyday practice or workplace. They are not constraint by the contents studied in schools. The situations in ill-structured problems are not well-defined and clear, and their solutions are usually unpredictable. On the other hand, the well-structured problems are restricted to certain situations that require application of finite number of rules, principles and concepts. Their attributes and components have to be presented to problem solvers, and thus, they have preferred or prescribed solutions. Computer programmers may encounter ill-defined problems; however, they primarily need to acquire programming skills in well-defined formats. Although the studies exploring well-structured and ill-structured PS processes in different contexts present some distinctions, the well-structured PS practices are believed to have improved CP skills.

CP and PS have much in common per se. PS is similar to the Structured Programming Paradigm that breaks programs into interacting modules, each of which is in charge of executing one of the program's functions. Both of the PS and CP disciplines require higher order cognitive skills and they engage a variety of cognitive components (Palumbo, 1990). Cognitive activities are mainly employed for (1) understanding programming problem, (2) problem decomposition, and (3) implementing the solution (Tegarden & Sheetz, 2001). Understanding problem and its sub-problems are associated with obtaining the requirements for the system, understanding behavior and components, and organizing the knowledge about CP domain. Decomposition involves breaking the current problem into smaller sub-problems to create a logical solution based on the programmer's understanding. When decomposing the system, the programmer

makes decisions on logical design and physical structure of the software system. Solution phase includes verifying the designs, and then writing the code to implement the design criteria. Consequently, writing a program requires learners to explicitly use PS skills, and therefore, CP can be regarded as a type of PS process (Benjamin, John, & Scot 2008; Brooks 1999; Hung, 2008).

Situation in the Faculty

The introductory CP course has been taught for about three decades in the faculty. It is a required course for the first year students majoring in different disciplines, such as electronic, mechanical, and industrial engineering. The intuitive belief has been that the learners would develop PS skills when learning and solving CP problems. It is also expected that they could transfer their learning experiences to other knowledge domains. However, the current situation has implied that this might not be the case. The main point is that the traditional teacher-directed method has been the primary method. Therefore, the faculty members have been using the direct instructional techniques for the CP course. Learning is assumed to occur if the students can understand presented knowledge, and then they transfer this knowledge to programming tasks in the labs. The main emphasis is put on the programming exercises during lab hours following the lectures. Yet, the transfer of knowledge from lectures to laboratory has been one of the headmost problems of novice learners.

Initially, it was very important to determine the instructional requirements. Therefore, the instructional and the root-cause analysis techniques were integrated to form the theoretical grounding of the study. The root-cause analysis (Evans & Lindsay, 2005) was helpful for focusing on various possibilities while brainstorming on causes of the problem (Figure 1). It provided a systematic approach to effectively identify the potential main or sub-causes before jumping to a conclusion. The instructional root-cause analysis guided how to assess the current needs, learners, environment, tasks and goals.

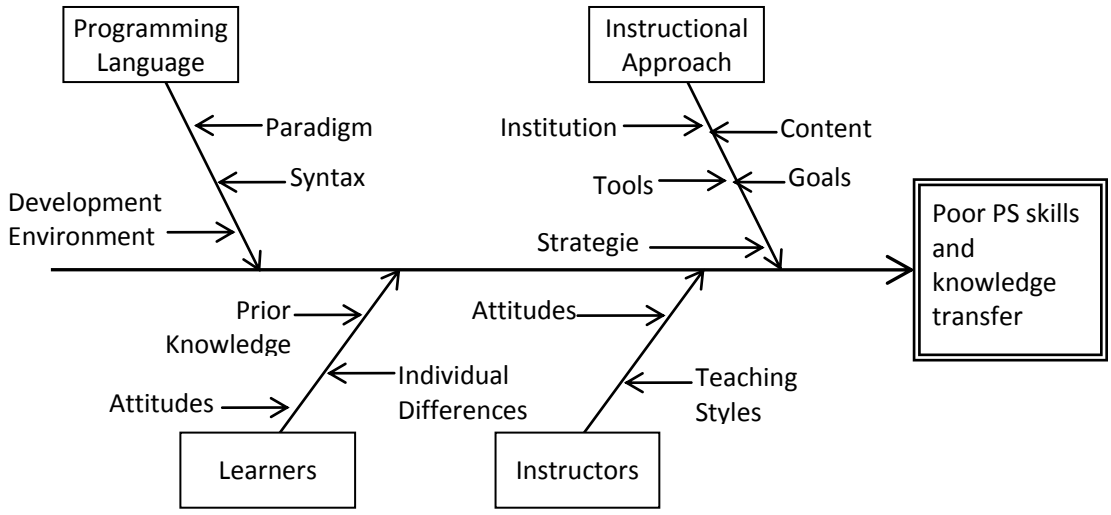

Figure 1. Instructional Root-Cause Analysis

Figure 1 depicts the possible sources of the problem in the form of a fishbone diagram providing the basis of decision making process. The “poor PS skills and knowledge transfer” is determined as the main problem statement. The main branches, “Programming Language”,

"Learners", "Instructors" and "Instructional Approach", form the possible major factors. The sub-branches, such as the "paradigm" of the programming language, learner "attitudes" to CP, instructors' "teaching styles", and the "learning goals" may be possible sub-causes of the problem. The outline of analysis presents the factors and current status of the CP course in the faculty as follows:

Programming language: C has been taught as an introduction programming language in the faculty, and it is regarded as one of the most widely used programming languages. With its unique characteristics, it also has influenced many contemporary languages, and C has been used as an intermediate for the implementations of other programming languages. It facilitates adopting the Structured Programming Paradigm by use of "functions" that contains executable code. Its statement and expression syntax rules, data models and structures have allowed large-scale programming. Consequently, C is an appropriate and motivating language for teaching CP through PS strategies.

Learners: Being a first-time programmer with no or little experience was an important issue. Studies indicate that anxiety owing to poor knowledge or lack of motivation might be some of the causes of academic failures in CP (Jerez et al., 2012). Motivation, attitudinal aspects, such as confidence and low anxiety, are important to a PS process, and thus they reinforce the role of contextualization in PS (Jonassen, 2000). Therefore, relating the course contents with students' majors or using CP knowledge in meaningful learning contexts as in PS can make the learners much more motivated and enable the knowledge transfer.

Instructors: As mentioned before, the instructors have preferred and used the teacher-directed approach, which is also an official instructional method of the faculty. They focused on the contents, and felt responsible for providing and controlling the flow of the CP course. The much of students' active participation was in the laboratory sessions, and little emphasis was put on modeling CP through different techniques during the lectures.

Instructional method: Although the faculty members seemed to believe that complementary or new strategies could contribute to the course, they were not enthusiastic about a radical change in their teaching patterns and practices. However, a new teaching approach, which would bring a different perspective to the course, would improve PS and CP skill and motivate the leaners. As previously stated, PS method has been one of the techniques used for CP instruction aiming at enhancing both meaningful learning and knowledge acquisition (Benjamin, John & Scot 2008; Hung, 2008).

Method

Research Design

This was a quasi-experimental study with a posttest-only design that included a control group (Table 1). The research purpose was to investigate the effects of a web-supported and well-structured PS method on academic achievements and PS perceptions. Therefore, the study had two mainstays: (a) adopting an instructional design theory for well-structured PS; and (b) integrating a web-based tool with this design. While the web-supported and traditional approach was used for the control group, the experimental group was treated with the web-supported and well-structured PS method. A cluster random sampling was used due to administrative reasons, and thus, the existing 18 sections were randomly assigned to the study

groups. Academic achievements and PS perceptions constituted the dependent variables of the experimental design. The hypotheses were as follows:

H-1. When compared to traditional instruction, well-structured PS instruction will not have a significantly different effect on academic achievement.

H-2. The instructional treatments will not make significant differences between the study groups in terms of PS perceptions.

Table 1. Research Design

Study Groups	**Instructional Treatments**	**Function of Web-Based Tool**	**Post-Tests**
Experimental	Well-structured problem-solving	Guiding & supporting PS activities, posting course materials	(a) CP achievement (b) PS perception
Control	Traditional	Posting only course materials	

Participants

The 433 first-year military undergraduate students, who were 2.3% female (n=10) and 97.7% male (n=423), and aging from 18 to 20, participated in the study at the second semester of the 2011-2012 academic year. They took the "Introduction to Computer Programming with C" course in 18 sections instructed by 6 different faculty members. The majority of the students had taken computer literacy courses during high school education but they did not have experiences in CP.

On the instructors' side, the faculty members in the experimental group had received training in conducting a course based on well-structured PS activities. Although they had seemed to be concerned at the beginning, it was later observed that they felt comfortable as they would conduct a modified version of the traditional course with PS activities guided by a web-based tool. The instructors using traditional method also had no experience in PS methodology but agreed to teach the whole content with direct-presentation format. The most experienced (10 and 14 years) and most inexperienced (2 years) instructors were assigned to the PS group to balance experience. All of the remaining instructors were between 28 and 41 years of age and averaged 9 years of teaching practice.

Instructional Treatments

Well-Structured PS Instruction

The instruction was designed according to the Jonassen's (1997) "Instructional Design Model for Well-Structured Problems" (Figure 2). Since the participants were first-time programmers, it was also expected that this approach would motivate the learners, and help them to internalize CP knowledge through well-defined PS activities. The first four steps were executed in the lectures, and the next three steps were in the lab hours. The main focus was on the

programming problems, their representations, and solution processes of the programming tasks. The lectures and labs were integrated to model CP as a form of well-structured PS process with its required activities.

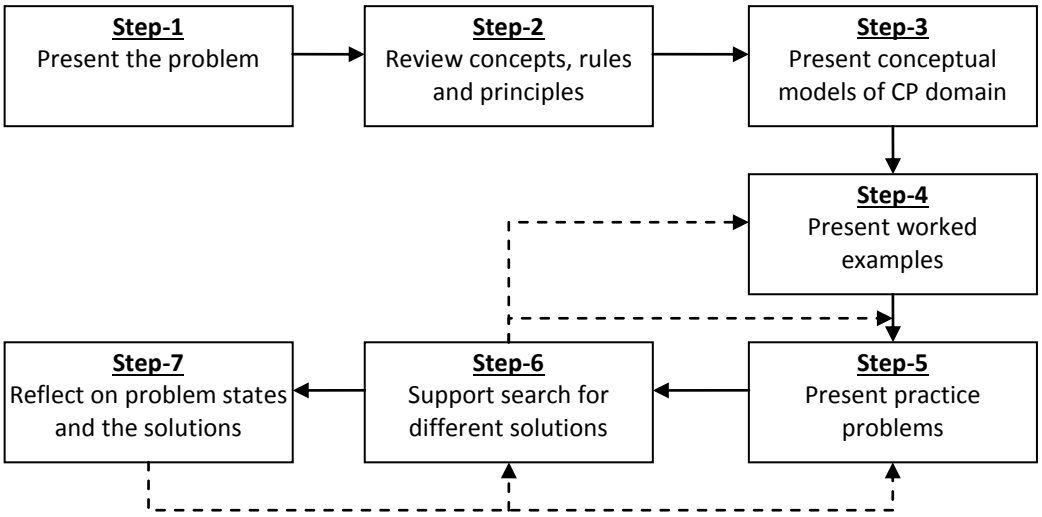

Figure 2. Well-Structured PS Instruction

Step-1: Presenting the problem; weekly lectures started with a well-defined programming problem followed by the presentation of programming facts and rules in the form of PS elements.

Step-2: Reviewing concepts, rules and principles required for solving the problem; solving a well-structured problem required identifying, selecting and applying CP domain skills and knowledge. Therefore, prior to presenting the current content, the learners initially reviewed their previous knowledge and concepts.

Step-3: Presenting conceptual models of the CP domain; the CP facts and rules related to the programming problem were presented as PS elements. The conceptual models contained the visual representation of the essential parts, states and attributes of the problem at an appropriate level of detail. They clearly represented the structural knowledge required to support PS process (Jonassen, Beissner, & Yacci, 1993). By using visual conceptual models, it was aimed to ease retention and recall of programming knowledge, and also to display the interactions between the problem and corresponding programming elements. This was to associate the states and attributes of the problem with its possible solutions.

The flowcharts and pseudo codes were combined to form the conceptual models and worked examples to help the learners to understand program logic and sequence. The data flow and essential steps in the PS process were presented using shapes and flow lines of the flowcharts. The pseudo codes were especially helpful for the high-level descriptions of the solution algorithms while using the natural language and conventions of the C language. This type of visualization also aimed to enhance the students' mental models pertaining to well-structured PS process.

Step-4: Presenting worked examples for modeling PS performance; there are studies showing that worked examples could be more beneficial for inexperienced learners since starting

directly with PS may impose heavy mental load (Kalyuga et al., 2001; Sweller et al., 1998; Uysal, 2013). It is also known that learners engaged in worked examples can adopt PS techniques and improve PS skills rapidly (Chandler & Sweller, 1991). Therefore, the worked examples aimed at modeling CP process using PS strategies. The learners reflected on these examples during lectures to learn how to construct the representations of programming problems. With this, it was intended to help learners to construct useful PS schemas as well as to categorize programming problems with similar solutions. The worked examples were primarily constituted by the snippets of program code, which were also combined with the explanatory flowcharts and pseudo codes.

Step-5: Presenting practice problems; this instructional step was carried out in the lab hours. Automating the use of syntax rules and programming structures to solve programming problems could be a slow process and it may require extensive practices. Although PS schemas in learners may develop quickly, the worked examples may not be sufficient alone (Jonassen, 1997). Therefore, a variety of practice problems were used for facilitating programming schemas and also for allowing the transfer of CP knowledge to novel problems. They were made more realistic by withholding some programming elements or including irrelevant variables in the description of programming problems. Thus, the learners were expected to develop the skills for retrieving the information necessary to solve the problem, or identifying the critical information missing from it.

Step-6: Supporting the search for different solutions; it was important to provide learners with different problem solutions when helping to construct effective PS and CP schemas. Therefore, a variety of strategies supported the search for different solutions, such as “analogical problems”, “recalling a previously developed program” or “breaking down a problem into sub-problems”. These strategies were used for assisting the learners in developing skills for generating different solutions and algorithms. For example, analogical programming problems were similar to the ones previously solved. They were powerful scaffolds to support learning. They mapped a previous problem onto new one and made the use of prior experiences possible. The remaining techniques were providing feedbacks, hints and relevant cues when the learners engaged in the practice problems.

Step-7: Reflecting on problem states and problem solution; learners had to reflect on different problem states and conditions. It was essential to note to the characteristics of a programming problem, its known, unknowns and the situation in which they were stated (Jonassen, 1997). During this step, the learners were focused on the solution processes, which were especially effective or not effective in solving practice problems. The students were expected to associate the programming problems with successful solutions.

When executing the steps of PS instruction, the web-based tool supported the instruction not only by providing the instructional materials, conceptual models and diagrams, but also by guiding the instructors, presenting the practice problems and worked examples as well.

Traditional Instruction

The instructional activities were similar to the ones in traditional CP courses (Figure 3). The faculty members directed the instruction and controlled the flow of contents as in the PS instruction. However, the focus was on the C programming language itself instead of PS activities during the lectures and labs. The students worked in the labs, and practiced on what

had been presented in the lectures so far. The Web-based tool posted only the "MS Power Point" presentations, assignments and handouts.

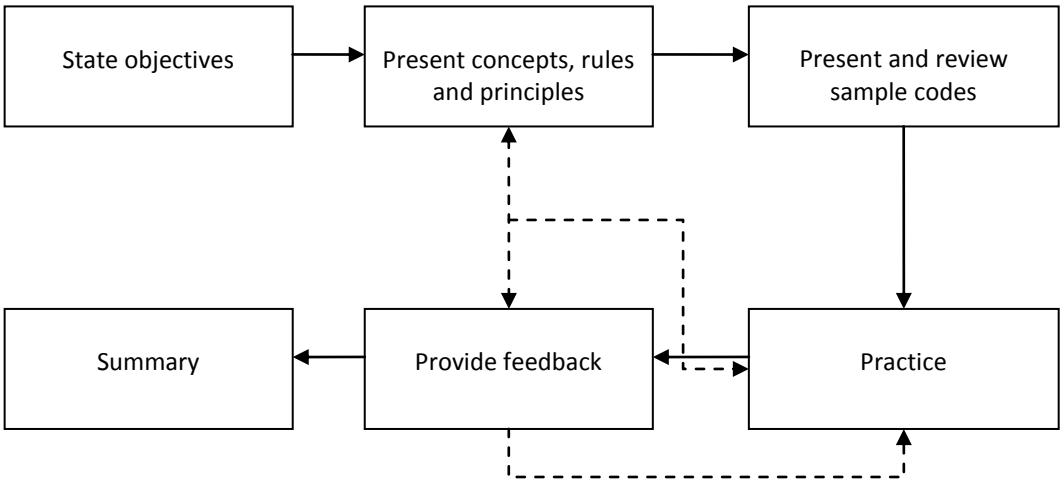

Figure 3. Traditional Instruction

Experimental Procedures

The CP course consisted of a two-hour lecture and a one-hour lab per week, with 15 weeks per semester. The students participated in the study in 18 sections, and therefore, 219 students were in the PS group and 214 students were in the traditional group. The course's scope and content were identical for the study groups, but the instructional designs differed (Table 2). The weekly contents were grouped into the units so that they were consisted with each instructional approach. Homeworks were assigned every two week to both of the groups, and they had to be handed in at the beginning of the next lecture. Besides participating as an instructor, the researcher played an advisory role and provided the support to the instructors throughout the study.

The lectures for the experimental group started with a well-structured programming problem. Therefore, learning was expected to promote when the learners were engaged in the PS activities towards this problem. Prior to presenting the content, the instructors and learners initially reviewed the prerequisite knowledge. For example, the use of "selection structures" was reminded before passing to the topic "repetition structures". The flowcharts and pseudo codes formed the conceptual models. The worked examples helped the PS learners to categorize similar solutions to well-structured problems. In an example, different types of loops, such as "while" and "for-loop", were used in different examples for the topic "repetition structures". During the labs, the PS learning group was confronted with a variety of practice problems that were aligned with the weekly problem, and the students reflected on the effective solutions. The learners of this group were relatively more active than the learners of the control group in lectures and labs. However, the instructors of the PS group took the main responsibility of problem-based flow of the course and acted as a guide for the PS activities.

As for the control group, a traditional, teacher-centered and content-driven instructional approach was adopted. The instructors of this group were mainly responsible for providing the course contents, such as syntax rules, CP techniques and principles. In the lectures, CP

concepts, rules and principles were presented before reviewing the sample codes or snippets. The instructors used only visual or oral presentation technique as the main strategy to transfer knowledge on a particular topic. In the labs, the learners of control group were responded only when they asked for a help. They were mostly active in the labs, and practiced on similar but discrete programming tasks.

Table 2. Design Criteria for the Instructional Methods

	Design Criteria	**Traditional Instruction**	**PS Instruction**
1	Instructors' Role	a. Providing and controlling only the flow of contents b. Providing support when asked	a. Providing PS flow of the contents, b. Guiding the learners during PS, c. Monitoring and providing support during the process of PS activities
2	Learners' Role	a. First take in information then apply it, b. Active mostly in lab hours	a. Process information with PS activities, b. Relatively more active in lectures and labs
3	Content Presentation	Presenting CP facts and rules only	Presenting CP facts and rules in a well-structured PS context
4	Practicing	a. Presenting sample code snippets and limited examples in lectures, b. Practicing discrete programming tasks in the labs.	a. Starting with a sample well-structured problem, b. Modeling well-structured PS processes with worked examples, c. Practicing with different versions of the same problem, d. Reflecting on effective solutions
5	Functions of the Web-Based Tool	a. Web-supported presentation of the course materials, b. Serving as a container for the course materials	a. Web-supported implementation of PS instruction, b. Multimedia presentation of the PS materials, c. Serving as an instructional guide for the instructors of PS, d. Serving as a container for the course materials

Instruments

The Problem Solving Inventory (PSI) was the instrument for assessing the participants' PS perceptions, their beliefs and attitudes associated with PS (Heppner, 1992; Heppner & Petersen, 1982). It is a 6-point Likert scale with 35 items including 3 filler items. The initial exploratory factor suggests three factors within the PSI: (a) Problem Solving Confidence factor (PSC: 11 items) represents the person's believe and trust in his/her own PS abilities; (b) Approach-Avoidance Style factor (AAS: 16 items) defines the person's general tendency to approach or avoid PS; and (c) Personal Control factor (PC: 5 items) measures to what extent the individuals believe that they can control their emotions and behavior when solving problems. The total score is used as a single measure for an effective PS ability, and it reflects a person's overall appraisal of his or her PS style. Low scores indicate the person's greater perception of an effective PS ability, which is also the same for the PSC, AAS and PC measures. The intercorrelation among the factors ranges from .39 to .69 in a variety of studies on PSI, and

the results also suggest that the factors are independent enough to be considered as separate factors (Heppner, 1982; Heppner, Pretorius, Wei, Lee, & Wang, 2002). Subsequent studies using either confirmatory or exploratory factor analyses indicate that the PSI factors can replicate well across different age groups with various backgrounds (Heppner, Witty, & Wayne, 2004).

There are also studies exploring the cultural validity of PSI (Sahin, Sahin, & Heppner, 1993). Nearly the thirty years of PSI have presented consistent patterns in the knowledge base, and showed that it could be used as an effective and valid tool in a wide range of disciplines (Heppner et al., 2004). The previous factor analyses of the PSI (Heppner, 1988; Heppner et al., 2002; Laporte, Sabourin, & Wright 1988) and a Turkish sample (Sahin et al., 1993) provided support for the original factor structure of this scale. However, the factor structure was reexamined by conducting a confirmatory factor analysis with 90 students in the faculty, and the Cronbach alpha (α) values of each factor were calculated. The results showed fewer items loading, and therefore some of the items of factors PSC (12th, 34th, and 35th items), AAS (13rd, 17th, and 30th items) and PC (32nd item) were excluded. The adapted version of PSI was employed to measure the PS perceptions of 433 students at the end of the course. The estimates of internal consistency were examined for each of the factors. Consequently, the Cronbach alpha coefficients were: .94 for PSI total; .87 for PSC; .89 for AAS; and .72 for PC. These figures suggest that the PSI and its factors have high levels of internal consistency.

As to the academic performances, a five-response multiple-choice test with 20 items measured the learners' achievements, and one correct answer was deducted for every four incorrect answers. The test was prepared and evaluated by the CP group of faculty members according to the course objectives and unit plans, and it was also examined by this group for content and face validity. At the end of the research study, the test was employed as a posttest to measure the learners' CP knowledge and skills. In order to assess the quality of the items and of the posttest as a whole, the Item Analysis Process was used to examine the learners' responses to individual test items. The item analysis included two statistics, such as the question difficulty and the question discrimination values. The results showed that the posttest had an acceptable range of question difficulty levels, and it was able to differentiate among the learners in terms of CP knowledge and skills.

Web-Based Tool

Research on the technology-supported instructional environments suggests that flexibility and access to different forms of instructional materials can be contributory. In this study, the main motivation behind the use of a customized web-based tool was driven from the need for a more accessible course as well as from the difficulties in implementing a PS instruction (Hoffman & Ritchie, 1997; Kinnunen & Malmi, 2005). The tool offered two types of interfaces for students, each of which was for the corresponding group instructed by either PS method or traditional method. Although the menu functions and the knowledge presentations were different, the same interface design principles were applied for the web-based application. For example, the text presented on a given page was limited and scrolling was avoided. The over-all layout of the web pages had the same structural meaning. Moreover, a special notice was given to the multimedia learning principles (Mayer, 2009). For example, "learning is better from words and pictures than from words alone". Therefore, the graphical representations, such as flow-charts and pseudo codes, supported the integrated display of text and images. The final considered principle was "spatial contiguity", which proposed that "people learn

better when corresponding words and pictures are placed near each other rather than far from each other on the screen".

Findings

Descriptive and inferential statistical techniques were used for analyzing the data and testing the hypotheses. The findings were depicted in the tables and their interpretations were presented in the corresponding paragraphs of relevant sections. The findings and discussions were organized under the titles of the dependent variables.

Academic Achievements

A summary of descriptive statistics about academic achievement scores is presented in Table 3. As seen, the average score for the experimental group (Mean=82,46) is relatively higher than the average score of the control group (Mean=79,83).

Table 3. Means and Standard Deviations for Academic Achievements

Groups	n	Mean	Std. Deviation
Experimental Group	219	82,46	8,94
Control Group	214	79,83	9,31
Total	433	81,16	9,21

Based on the tests of normality (Kolmogorov-Smirnov and Shapiro-Wilk), the academic achievement test was not normally distributed ($p<.05$). Therefore, the Mann-Whitney Test was used for the analysis procedures. As a result, a statistically significant difference was found in the academic achievements at the $\alpha = .05$ level of significance ($z=-3,211$; $p< .05$). It is possible to state that the learners instructed by the web-supported PS method displayed higher academic performances (Table 4).

Table 4. The Mann-Whitney Test Results of Academic Achievement Test

Groups	n	Mean rank	Sum of ranks	*z*	p
Experimental Group	219	236,08	51701,50	-3,211	.001
Control Group	214	197,47	42259,50		

Perceptions on PS Skills

A summary of descriptive statistics about PSC, AAS, PC and PSI (total) scores is presented in Table 5.

Table 5. Means and Standard Deviations for PSC, AAS, PC and PSI (total) scores

Factors	Groups	n	Mean	Std. Deviation
PSC	Experimental Group	219	17,29	8,42
	Control Group	214	18,67	8,31
AAS	Experimental Group	219	24,90	11,49
	Control Group	214	27,05	11,31
PC	Experimental Group	219	8,31	4,21
	Control Group	214	9,77	4,62
PSI (Total)	Experimental Group	219	50,50	22,90
	Control Group	214	55,49	22,38

The PSI scores were not normally distributed ($p<.05$), and therefore, the Mann-Whitney Test was used for the analysis. The test results are presented in Table 6. Although the null hypothesis stated the opposite, there was enough evidence to conclude that there were statistically significant differences between the groups for the PSI, PSC, AAS scores (PSI: $z=-2,634$, $p<.05$; PSC: $z=-2,021$, $p<=.05$; AAS: $z=-2,222$, $p<.05$); but not for the PC score (PC: $z=-1,183$, $p>.05$). It is possible to state that the web-supported and well-structured PS instructional method effected and improved the PS perceptions of learners. In other words, the learners in the PS group perceived themselves: (a) more effective in PS (PSI score); (b) more tend to PS (AAS score); (c) and more confident (PSC score) in PS. However, the results indicated no statistically significant difference between the groups according to (d) their personal control on PS process (PC score).

Table 6. The Mann-Whitney Test Results of the PSI, PSC, AAS and PC scores

	Groups	n	Mean Rank	Sum of Ranks	*z*	p
PSC	Experimental	219	205,02	44898,50	-2,021	.043
	Control	214	229,26	49062,50		
AAS	Experimental	219	203,81	44633,50	-2,222	.026
	Control	214	230,50	49327,50		
PC	Experimental	219	209,86	44910,00	-1,183	.237
	Control	214	223,98	49051,00		
PSI (Total)	Experimental	219	201,35	44095,50	-2,634	.008
	Control	214	233,02	49865,50		

Results and Discussion

Academic Achievements

The research results mark the well-structured PS method as a significant factor for the academic achievements, and for the PS perceptions to some extent, as well. As previous work (Jonassen & Reeves, 1996; Pea & Kurland, 1984; Unuakhalu, 2004) had shown, learning CP through PS strategies affected the performances positively. This finding is similar to that of Hung's (2008) study suggesting that PS improved the understanding of a programming

language and increased the CP skills though his study included relatively limited PS techniques. One possible explanation for the observed relationship between the treatments and the academic achievements would be the integration of programming concepts and rules into a meaningful PS context (Allan & Kolesar, 1996). Presentation of the course contents with PS techniques provided the knowledge at a level of appropriate detail and familiarity. Thus, this provided a more learner-centered approach rather than simply presenting the CP facts and rules to the students.

Novice learners mostly attend to surface features, rather than deeper knowledge without being guided. They usually have difficulty in connecting the attributes of CP facts and rules with those of programming exercises, especially when knowledge presentation is separated from practice in time and/or space. Therefore, another reason for the higher achievements would be the meaningful connection of the labs and the lectures based on the principles of a well-structured PS design theory. By using well-structured PS techniques, the learners were simultaneously able to focus on the aspects of problems and on the CP structures. This naturally enriched the students' experiences, and also enabled the support of PS activities with additional strategies, such as providing hints and cues, guiding and directing the students' attention, elaborating, and eliciting knowledge from PS activities. These strategies, therefore, helped the students to reflect on CP context by providing meaningful abstraction of programming knowledge.

In view of the cognitive principles, there are also additional explanations. Although Davies (2000) reports the empirical results on the retrospective and forward-planning activities required for a well-structured PS process, he implicitly directs the attention to the Information Processing Theory for the discussion of these results. That is, how information is coded and processed affects the quality and effectiveness of a cognitive performance. It is thought that reflecting on programming facts and rules, which were in the integrated form of worked examples, practice problems and effective solutions possibly helped the learners to develop stronger schemas (Ge, 2010). As in the Pedersen and Liu's (2002) study, worked examples and PS strategies showed cognitive modeling, and they were appropriate especially for the inexperienced learners (Crippen & Earl, 2007; Kalyuga et al., 2001). Thus, the students were able to reflect on the facts, rules, and programming techniques easily. The use of worked examples in learning process, at the same time, lessened the load on working memory while enabling the focus on the states of programming and PS process (Sweller, 1998). Furthermore, task automation and schema acquisition are also important in learning a complex skill like CP (Merrienboer & Paas, 1990). This frees up the working memory, reduces the cognitive load, and information can be processed automatically without extra mental effort (Ericsson & Kinstch, 1995; Uysal, 2013). Therefore, PS instruction is thought to have provided the learners with automated CP and PS skills needed for high-level cognitive performances (Renumol, Janakiram & Jayaprakash, 2010).

Another important factor was the guidance that the well-structured PS instructional design emphasized for a meaningful learning context. It is thought that the added benefit of the instructor-led PS activities provided the students with a guide for interpreting syntax rules and programming structures (Perrenet, Bouhuijs, & Smits, 2000). Instructors of the experimental group provided this required guidance with the learners on a just-in-time basis at any phase of programming and PS activities. Without this, it would probably take more time for students to figure out how best to transfer knowledge to the solutions of programming problems (Kirschner et al., 2006). Thus, the learners in PS group were possibly able to build more CP domain-specific representations, and they were able to note the characteristics of CP in more

detail (Jonassen, 1997). Consequently, this situation made CP more understandable, enjoyable, and therefore, enabled the learners to reach to the solutions with fewer steps (Bude, Wiel, Imbos, & Berger, 2011).

Perceptions on PS Skills

The findings of this study are consistent with the literature on studies comparing PS instruction to traditional one (Dochy, Segers, Bossche, & Gijbels, 2003; Strobel & Barneveld, 2009). However, one important finding is the identification of non-significant difference between the students according to their Personal Control (PC) scores. This meant that the students did not perceive a great deal of emotional and behavioral control over their PS performances. This is not surprising to the researcher. Although the students in the PS group acquired CP skills using PS strategies, the instructors played the central role, and they guided and supported the learners when needed.

Contrary to the expectations, academic achievements did not correlate with PS perceptions ($r=.036$; $p>.05$). This may be explained by the Heppner's (1997) inferences, such that the PSI might be used to identify students at risk of academic failure, but not for predicting their course grade. On the other hand, the research on the PSI suggests that PSI factors can be positively and/or negatively associated with personal agency, positive affectivity, curiosity, anxiety etc. Even though the PSI has demonstrated adequate validity in comparisons with different measures, the possibility of confounding effects cannot be ruled out. The results, therefore, need to be interpreted with caution and future research is needed to address these issues, such as the self-perceived PS ability with subjective and objective behavioral indicators. Nevertheless, the results imply that the students' PS perceptions and their programming performances can be improved in classroom through the use of PS strategies.

Another point that has to be discussed is the use of a web-based tool throughout this study. At the first place, it provided a means to manage the learning resources for all students, and provided an efficient mechanism for the design and implementation of PS instruction (Walker, Recker, Robertshaw, Osen, & Leary, 2011). It, most importantly, established an application framework for the activities of PS across a large number of students. As the research on technology-supported learning suggests (Park & Ertmer, 2008), a rich form of access to knowledge and its presentation was achieved through multimedia technologies. Thus, the web-based tool is thought to have supported the students' information-processing ability essential for high-level cognitive performances (Chiou et al., 2009; Hwang et al., 2011; Kuo et al., 2012). The learners, therefore, could easily note and review the concepts, principles and techniques pertaining to either PS or CP process. The results illustrate that hypermedia can be used to enhance understanding of CP if it is guided by an instructor and supported by a PS context.

During this study, one critical issue was also the transition from an instructor role to the PS facilitator role in the classroom, and this would take time for teachers. It was highly probable and easy for the instructors to refer to their old habits, especially when students were struggling with understanding the programming paradigm. Therefore, adapting the web-supported PS method to the classroom established a more student-centered environment and supported the transition to PS learning context (Park & Ertmer, 2008).

As a result, the early indications in this study suggest that the PS instructional method has facilitated the shift away from surface learning, though it is still unclear how deep or to what extent learning has occurred.

Limitations of the Study

Although this study provides several important findings, certain limitations should be considered as well. First, the conclusions drawn from the study are limited by the students' profile and by the nature of the PS environment. The participants were military students living on the campus. Even if this is convenient for sampling and conducting the experimental study, it means no high population validity, and therefore, the findings may not be widely generalized. Second, the effects of the well-structured PS activities may be due to some aspects of the presence of instructors regulating the PS environment (Bude et al., 2011). Third, learners have different attitudes or preferences towards to take in and process information. However, this study did not address these individual differences though the conclusions drawn might vary depending on the different aspects of cognitive psychology, such as cognitive styles. Finally, the problems used in this study did not include ill-structured problems and they were not either in the scope of this study though they are ideally needed for a learner-centered instruction. It is thought that these issues and the limitations can be addressed by careful longitudinal studies that may also provide valuable information for the literature in the field of educational technology.

Conclusions

This study tried to identify whether a web-supported and well-structured PS instructional method would lead to higher academic performances and PS perceptions as well. The PS method was the treatment for the experimental group and the traditional method was adopted for the control group. A web-based tool enabled a more accessible course, provided effective knowledge presentation, and alleviated some of the obstacles in performing PS activities. The PS techniques were used for acquiring CP knowledge rather than exposing the learners directly to the concepts or syntax rules of C programming language. Therefore, the learners in the PS group framed their first-time programming experiences through a series of well-structured PS activities. The PS perceptions were assessed by the PSI, and the performances were measured by the academic achievement tests. The results indicated a significant difference between the groups in terms of CP achievements. Except for one factor of the PSI, there were also statistically significant differences between the groups in terms of their PS perceptions.

This study contributes to the research on educational technology in three ways. First, it adopts an instructional design theory for well-structured PS instruction. Second, it shows that well-structured PS method contributes to the pedagogy of Structured Programming Paradigm. Third, it integrates PS activities with web and multimedia technologies to enable more accessible PS instruction. As a consequence, the study can be seen as an attempt to the enhancement of a classroom learning environment, and therefore, it has showed the potential for using the web-supported PS method as a means of improving a traditional CP course. It is hoped that the present study may extend the previous knowledge both by the tools it has utilized and by the instructional approach it has adopted.

Acknowledgement

The study was conducted in the Department of System Engineering at the Turkish Military Academy. The author would like to thank the members of this department for their participation as well as their valuable support throughout the study.

References

Allan, V.V. & Kolesar, M. (1996). Teaching computer science: A problem-solving approach that and ineffective students during computer programming, *ACM Transactions on Computing Education*, *10*(3), 211-232. DOI = 10.1145/1821996.1821998.

Park, S.H. & Ertmer, P.A. (2008). Examining barriers in technology-enhanced problem-based learning: Using a performance support systems approach. *British Journal of Educational Technology*, *39*, 631-643.

Sahin, N., Sahin, H.N., & Heppner, P.P. (1993). Psychometric properties of the Problem Solving Inventory (PSI) in a group of Turkish university students. *Cognitive Therapy and Research*, *17*, 379-396. works. Proceedings of '96 National Educational Computing Conference, Minneapolis, MN.

Arlin, P.K. (1989). The problem of the problem. In J.D. Sinnott (Ed.), *Everyday problem solving: Theory and applications* (pp. 229-237). New York: Praeger.

Benjamin, A.R., John, G., & Scot, R. (2008). Problem solving through programming: motivating the non-programmer. *Journal of Computing Sciences in Colleges 23*(3), 61-67.

Brooks, R. (1999). Towards a theory of the cognitive processes in computer programming. *International Journal of Human Computer Studies*, *51*, 197-211.

Bude, L., van de Wiel, M. W. J., Imbos, T., & Berger, M. P. F. (2011). The effect of directive tutor guidance on students' conceptual understanding of statistics in problem-based learning. *The British Journal of Educational Psychology*, *81,* 309-324.

Chandler, P. & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition and Instruction*, *8*(4), 293-332.

Chen, C.H. (2010). Promoting college students' knowledge acquisition and ill-structured problem solving: Web-based integration and procedure prompts. *Computers & Education*, *55*, 292-303.

Chiou, C. K., Hwang, G. J., & Tseng, J. C. R. (2009). An auto-scoring mechanism for evaluating problem-solving ability in a web-based learning environment. *Computers & Education, 53*(2), 261-272.

Choi, W.S. & Repman J. (1993). Effects of Pascal and FORTRAN programming on on the problem solving abilities of college students. *Journal of Computing and Research on Computing in Education*, *25*(3), 290-302.

Crippen, K.J. & Earl B.L. (2007). The impact of web-based worked examples and self-explanation on performance, problem solving, and self-efficacy. *Computers & Education*, *49*(3), 809-821.

Dalton, D.W. & Goodrum, D.A. (1991). The effects of computer programming on problem solving skills and altitudes. *Journal of Educational Computing Research, 7*(4), 483-506.

Dale, N.B., Weems, C., & Headington, M.R. (1997). *Programming and problem solving with C++*. Sudburry, MA: Jones and Bartlett.

Davies, S.P. (2000). Memory and planning processes in solutions to well-structured problems. *The Quarterly Journal of Experimental Psychology*, *53*(3), 896-927.

Dochy, F., Segers, M., Bossche, P.V., & Gijbels, D. (2003). Effects of problem-based learning: A meta-analysis. *Learning and Instruction*, *13*, 533-568.

Dunlap, J.C. (2005). Problem-based learning and self-efficacy: How a capstone course prepares students for a profession. *Educational Technology Research and Development*, *53*(1), 65-85.

Ericsson, K.A. & Kinstch, W. (1995). Long-term working memory. *Psychological Review*, *102*(2), 211-245.

Evans J.R. & Lindsay W.M. (2005). *An introduction to Six Sigma & process improvement*. Thompson South-West Corporation, USA.

Ge, X. (2010). A cognitive support system to scaffold students' problem-based learning in a web-based learning environment. *Interdisciplinary Journal of Problem-based Learning*, *4*(1), 30-56.

Heppner, P.P. & Petersen, C. H. (1982). The development and implications of a personal problem-solving inventory. *Journal of Counseling Psychology*, *29*, 66-75.

Heppner, P.P. (1997). Applications of the Problem Solving Inventory. *Measurement & Evaluation in Counseling &, Development, 29*(4), 229-242.

Heppner, P.P., Pretorius, T. B., Wei, M., Lee, D. G., & Wang, Y. W. (2002). Examining the generalizability of problem solving appraisal in black South Africans. *Journal of Counseling Psychology*, *49*, 484-498.

Heppner, P.P., Witty, T.E., & Wayne, A.D (2004). Problem-solving appraisal and human adjustment: A review of 20 years of research using the problem solving inventory. *The Counseling Psychologist*, *32*(3), 344-428.

Hoffman, B. & Ritchie, D. (1997). Using multimedia to overcome the problems with problem based learning. *Instructional Science*, *25*, 97-115.

Hung, Y. C. (2008). The effect of problem-solving instruction on computer engineering majors' performance in Verilog programming. *IEEE Transactions on Education*, *51*(1), 131-137.

Hwang, G. J., Chen, C. Y., Tsai, P. S., & Tsai, C. C. (2011). An expert system for improving web-based problem-solving ability of students. *Expert Systems with Applications*, *38*, 8664-8672.

Jerez, J.M., Bueno, D., Molina, I. Urda, D., & Franco, L. (2012). Improving motivation in learning programming skills. *International Journal of Engineering Education*, *28*(1), 202-208.

Jonassen, D.H. & Reeves, T. C. (1996). Learning with technology: Using computers as cognitive tools. In D.H. Jonassen (Ed.), *Handbook of research for educational communications and technology*. New York: Macmillan.

Jonassen, D.H. (1997). Instructional design models for well-structured and ill-structured problem-solving learning outcomes. *Educational Technology Research & Development*, *45*(1), 65-94.

Jonassen, D.H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development*, *48*(4), 63-85.

Jonassen D., Strobel, J., & Lee, C.B. (2006). Everyday problem solving in engineering: Lessons for engineering educators. *Journal of Engineering Education*, *95*(2), 139-151.

Kalyuga, S., Chandler, P., Tuovinen, J., & Sweller, J. (2001). When problem solving is superior to studying worked examples. *Journal of Educational Psychology*, *93*, 579-588.

Kay J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J.H., & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education*, *10*(2), 109-128.

Kim, M.C. & Hannafin, M.J. (2011). Scaffolding problem solving in technology-enhanced learning environments (TELEs): bridging research and theory with practice. *Computers & Education*, *56*(2), 403-417.

Kinnunen, P. & Malmi, L. (2005). Problems in problem-based learning-experiences, analysis and lessons learned on an introductory programming course. *Informatics in Education*, *4*(2), 193-214.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, *41*, 75-86.

Kuo, F.R., Hwang, G.J. & Lee C.C. (2012). A hybrid approach to promoting students' web-based problem solving competence and learning attitude. *Computers & Education*, *58*, 351-364.

Lee, M.C. & Thompson A. (1997). Guided instruction in Logo programming and the development of cognitive monitoring strategies among college students. *Journal of Educational Computing Research*, *16*(2), 125-144.

Liao, Y. & Bright G. (1991). Effects of computer programming on cognitive outcomes: A meta analysis. *Journal of Educational Computing Research*, *7*(3), 252-268.

Mayer, R. E. (2009). *Multimedia learning* (2nd ed.). New York: Cambridge University Press.

Merriënboer, J.G., & Paas, G.W.C. (1990). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior*, *6*, 273-289.

Mills, J. & Treagust D.F. (2003) Engineering education-Is problem-based or project-based learning the answer? *Australasian Journal of Engineering Education*, *4*, 2-16.

O'Kelly, J., Mooney, A. Bergin, S., Gaughran, P., & Ghent, J. (2004). An overview of the integration of problem based learning into an existing computer science programming module. *Pleasure by Learning*, *1*, 1-4.

Palumbo, D. B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research*, *60*(1), 65-89.

Pea, R. D. & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137-167.

Pedersen S. & Liu, M. (2002). The effects of modeling expert cognitive strategies during problem-based learning. *Journal of Educational Computing Research*, *26*(4), 353-380.

Peng, W. (2010). Practice and experience in the application of problem-based learning in computer programming course. *Proceedings of International Conference on Educational and Information Technology* (ICEIT). Chongqing, China.

Perrenet, J.C., Bouhuijs, P.A.J., & Smits, J.G.M. (2000). The suitability of problem-based learning for engineering education: theory and practice. *Teaching in Higher Education*, *5*(3), 345-358.

Renumol, V.G., Janakiram, D., & Jayaprakash, S. (2010). Identification of cognitive processes of effective and ineffective students during computer programming. *ACM Transactions on Computing Education, 10(*3). Retrieved on 21 June 2014 from http://doi.acm.org/10.1145/1821996.1821998.

Strobel, J. & Barneveld, A. (2009). When is PBL more effective? A meta-synthesis of meta-analyses comparing PBL to conventional classrooms. *International Journal of Engineering Education*, *3*(1), 44-58.

Sweller, J., Merriënboer, V., Jeroen, J.G. & Paas, F. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, *10*, 251-295.

Tegarden D.P. & Sheetz, S.D. (2001). Cognitive activities in OO development. *International Journal of Human-Computer Studies*, *54*, 779-798.

Thomas R. A. & Upah, S.C. (1996). Give programming instruction a chance. *Journal of Research on Computing Education*, *29(1)*, 96-108.

Unuakhalu, M.F. (2004). Effect of computer programming instruction on the problem solving capability of college level introductory computer students (Unpublished doctoral dissertation). The University of Kentucky, Lexington, USA.

Uysal, M.P. (2013). Towards the use of a novel method: The first experiences on measuring the cognitive load of learned programming skills. *Turkish Online Journal of Distance Education*, *14*(1), 166-184.

Walker, A., Recker, M., Robertshaw, M.B., Osen, J. & Leary, H. (2011). Integrating technology and problem-based learning: A mixed methods study of two teacher professional development designs. *Interdisciplinary Journal of Problem-Based Learning*, *5*(2), 9-27.

Correspondence: Murat Pasa Uysal, Associate Professor, Learning and Research Center, Turkish Military Academy, Ankara, Turkey