International Journal of Progressive Education, Volume 15 Number 4, 2019

Four-Component Instructional Design (4C/ID) Model Approach for Teaching Programming Skills

Zafer Güney [i]
Istanbul Aydin University

Abstract

The need for methods, techniques and approaches that we can develop high-level thinking skills in important activities increases day by day in order to achieve effective use of technology and change in information and communication technologies. In particular, the diversity, complexity of technical skills and to gain technical skills required to be learned in schools and through applications in industry is important. Teaching the programming as a technical skill during instructional design process (ID) and how effective and meaningful teaching can be taught is an important problem. Thus, instructional design models have been developed for the solution of learning problems in systemic, systematic and appropriate learning conditions and especially for the development of technical skills (van Merriënboer (1997). The instructional design model (4C/ID) activity mentioned here can be used for teaching the importance of instructional and technological stages by combining and supporting another multimedia project design, development and evaluation model. This study presents technical skills only by pointing to the future developers and designers of programming that an instructional design approach can be used to develop other programming skills. In addition, through ten steps proposed for complex learning (van Merriënboer & Kirschner 2007) and steps in achieving complex cognitive, high-level, algorithm based limited coding, technical skills, it is to provide a new different approach to program developers, instructors and designers by planning and discussing the design of the process within a basic frame as to be in four stages (van Merriënboer & Kirschner 2007). The purpose of this study is to adapt the principles of the model for teaching technical skills by using four-component instructional design model (4C/ID) within software programming. In this study, theoretical framework for teaching complex technical skills, learning theories and problem solving in programming are given. The relationships between components of 4C/ID model presented for teaching programming skills. At the end of study, the ID model components and their applications for future programming skills were indicated.

Keywords: Instructional Design, Programming, Learning Technical Skills

DOI: 10.29329/ijpe.2019.203.11

---

[i] **Zafer GÜNEY,** Assist. Prof. Dr., Istanbul Aydin University, Department of Computer and Instructional Technologies

Correspondence: zaferguney@aydin.edu.tr

INTRODUCTION

The need for methods, techniques and approaches that we can develop high-level thinking skills in important activities increases day by day in order to achieve effective use of technology and change in information and communication technologies. In particular, the diversity, complexity of technical skills and to gain technical skills required to be learned in schools and through applications in industry is important. Teaching the programming as a technical skill during instructional design process (ID) and how effective and meaningful teaching can be taught is an important problem. For this reason, instructional design models have been developed for the solution of learning problems in systemic, systematic and appropriate learning conditions and especially for the development of technical skills (van Merriënboer, 1997). In addition to technical skills, there are several factors and skills that affect the effective learning of programming. These are; reading comprehension, critical reasoning, systemic thinking; have cognitive components in problem definition, planning and solution production, creativity and intellectual curiosity, mathematical skill, situational reasoning, process based (procedural) thinking and temporary reasoning, analytical and quantitative reasoning, benefitting from different sources, being flexible and creative in producing new solutions (Ambrosio et al ., 2011; Lau & Yuen, 2011). The value of having these skills has been the subject of many researches in terms of learning, and it is clearly stated in many of these studies that the above-mentioned skills need to be acquired by the students.

Problem:

In order to gain cognitive (technical) skills in programming teaching in accordance with the study here, it will be discussed how four-component instructional design (4C/ID) model can be used for teaching technical skill in programming (van Merriënboer, 1997). To make programming in the way, teaching procedures should include all stages of multimedia project design, production and evaluation. In this context, the steps taken for programming education can be provided to learn a chosen topic related to the course in the process of teaching program / software development in an instructional design and learning strategy in line with the ID model and multimedia project design, production and evaluation model in education.

In programming, planning should be made primarily within the framework of concept teaching based on ID model. After learning the concepts, coding in the desired language will be more practical (such as C, C ++, C #, Php etc.). Thus, the issue how the operation series requested in order to develop an efficient learning environment must be with the principles of instructional design (ID) approach of this process, is a learning problem which overlaps with the solution of waiting and technical skills.

Purpose of the study:

The purpose of this study is to adapt the principles of the model for teaching technical skills by using four-component instructional design model (4C/ID) within programming skills and limitations mentioned above and to discuss the differences in instructional planning for the software development process. In addition, through ten steps proposed for complex learning (van Merriënboer & Kirschner (2007) and steps in achieving complex cognitive, high-level, algorithm based limited coding, technical skills, it is to provide a new different approach to program developers, instructors and designers by planning and discussing the design of the process within a basic frame as to be in four stages (van Merriënboer & Kirschner 2007; Kirschner & van Merriënboer 2006, 2008).

Importance of Study:

In the studies on programming teaching, it is not mentioned much about learning strategies and teaching design models for teaching effective and meaningful programming with the difficulties experienced in teaching. Therefore, in this study, it is revealed that the contribution of instructional

design approach and models for the transfer of knowledge can contribute to programming education effectively.

The principles of instructional design create opportunities for learners and teachers in the development of technical skills as a process that advances in this process and makes learning more efficient (Merriënboer & Paas, 1990). Designing a new learning environment for the programming process with the Four Component Instructional Design Model (4C/ID) for the technical skills determined in line with these principles has a particular importance in terms of discussing a new teaching design for this instructional process in order to be effective, meaningful and efficient in acquiring different technical skills.

Limits of the Scope of the Study

The four-component teaching design (4C/ID) model and the four stages of these four steps as a result of horizontal and vertical intersections analysis, skills separation, limited coding, methods activities and model ten steps, the design stages in programming teaching should be considered in a step-by-step way. In cases where activities are needed, it can be used as a multimedia project and can be used as a multimedia project by integrating the teaching design approach in case of programming instruction and multimedia programming. These activities, together with an important point of view to gain instructional technical skills in the design, development and evaluation of programming, will be able to draw on more than one of the different ID models in terms of learning technical skills.

There are many difficulties in the process of programming teaching. While learning a concept of programming in individuals, factors such as processing differences and cognitive differences in knowledge need to be taken into account in programming processes; in most studies, it is observed that the approaches and principles of instructional design model are ignored. For this reason, programming design and the process of realization of it should be tried to be gained through instructional design and multimedia programs design model. The instructional design model (4C/ID) activity mentioned here can be used for teaching the importance of instructional and technological stages by combining and supporting another multimedia project design, development and evaluation model. This study presents technical skills only by pointing to the future developers and designers of programming that an instructional design approach can be used to develop other programming skills.

Theoretical Framework and Definitions

Algorithmic Thinking: It includes technical tasks and procedures that design the method in the process. Algorithm is the whole set of rules that convert procedures, methods, and rules into a set of instructions designed to achieve a particular result. Algorithmic thinking, many professional people in the working environment to comply with the rules specified and the specified process to execute automatically and the application of algorithmic thinking is a result (Amorim, 2005).

The conceptual comprehension and skill development process that we can evaluate in both numerical and verbal progressive cases is a very important instructional achievement in terms of realizing the students what they are doing. This will enable the discovery of ways to realize the effectiveness of the activity in terms of student-centered learning and the elimination of memorization. Beyond perceiving the concept at all levels, the student has gained the ability of problem solving with high level cognitive skills and perceived the importance of the structure of the algorithm and at the same time created with his own conceptual words. Planning algorithms can be designed based on the instructional design model strategies in this study process by effectively and efficiently demonstrating how and why algorithms are difficult and complex to achieve in calculation and conclusion.

Relationship between Critical Thinking and Programming

One of the leading goals of education; to learn how to live, to learn how to live lifestyle can eliminate the problems faced by creative thinking skills gained by people who can develop design-oriented design is able to develop. Critical thinking can be evaluated as the process of obtaining, comparing and evaluating the knowledge after the process. It is a philosophical view that cannot be effective and useful without critical thinking. In short, the source of the theory without thought, practice cannot be practice. For this reason, the definition of the field of educational technology has been defined within the scope of theory and practice within the half a century time-period (Seels & Richey, 1994; Januszewski, & Molenda, 2008a,b).

It can be said that critical thinking is the acceleration of knowledge since humanity can progress through creativity, problem solving and critical thinking. Critical thinking is one of the factors affecting the development of the scientific process. In terms of development, original and creative ideas can not be won by acknowledging or repeating. Therefore, the recruitment of new skills will be carried out, and according to new tasks to be learned or technical skills as an IT. Phased teaching based on the model can provide effective learning. Therefore, there is a need for a learning environment based on multimedia program development models with 4C/ID. It is seen that their examples are adapted to the teaching of different skills in learning theories and approaches. For example, three top-tier analysis, evaluation and creation are presented and shown in recent adaptations to Bloom's taxonomy; it is seen as a definition of higher-order thinking (Ennis, 1993).

In critical thinking, rather than negative judgment, it is necessary to convey the fact that events can have different approaches, and that the truths should be examined in accordance with the circumstances, and that the individual should believe and believe by researching, researching and practicing in the culture. By using instructional design models in programming, we must learn how to teach our students how to write code, and we need to design the students in a student-centered framework based on instructional design strategies, how to think and think more by teaching them how to think. In this way, creativity, innovation-based production and applications will increase. Most importantly, the understanding of the work done and the steps taken by the student and the instructor will be understood very well.

Problem Solving and Computational Thinking

High-level learning skills (Gagné high-level learning skills) suggest that mental processes, such as high-level and complex behavioral changes and the problem-solving result in a new learning outcome. In addition, we can define these skills as uncertainty, perseverance, honesty, good faith and open-minded problems, analytically and systematically analyzing, comparing and evaluating them logically, into action and behavior, and as a lifestyle (culture). Since higher-order thinking skills necessitate complex cognitive activities in individuals, it may be a cause of conflict for individuals as they require different learning outcomes and gains.

A high level of learning may not be possible for every learner, for example, a student who fails to learn at a cognitive (learning-problem solving) level has to be equipped with higher skills that require higher skills and complex problem-solving skills. These learning situations, as Gagne (1985) states, are described at five different levels as learning outcomes. These are intellectual skills and skills that require verbal, attitudes, dynamic, problem solving and high level learning skills. From this point of view, problem solving and algorithmic thinking reveal the importance of verbal and cognitive learning outcomes in terms of computational thinking. In other words, revealing, processing and re-using the information necessitates a whole set of skills. The concept of higher-order thinking skills is recognized as the common name of the skills that enable them to be reorganized and used beyond the recall and understanding of existing knowledge (Doğanay, 2007). When examined in this framework, it is defined as remembering and knowledge level skills in terms of staging relations, while application, evaluation and creativity are considered as high-level skills (Gagne, 1985; Ertürk, 1972).

In the same way, the concept of thinking is defined as the effectiveness of the mind by comparing the information about a subject, examining the connections between them and making a judgment or a decision (Turkish Language Society, 2018). In this approach, while assessment and implementation are high-level thinking skills, recall is defined as low-level thinking skill (Saygılı, 2010).

It is a term used to describe the processes that an individual realizes his own cognitive processes, for his monitoring, supervision, and solving problem. A study by Erdoğan (2005) is considered as an indicator of academic and general success on programming success. In addition to this, programming education is associated with many parameters such as general ability, general academic achievement, mathematics achievement, abstract thinking ability, focus on detail, concentration level. A high level of relationship was found between the success of the students and the general success of the students who studied programming (Newsted, 1975; Hostetler, 1983; Whipkey, 1984; Byrne & Lyons, 2001).

Clement and Gullo (1984) who analyzed the contribution of computer programming to small-age students in groups of seven years old, found that In addition, programming improves the student's ability to find solutions to a problem and the ability to analyze (Akpınar & Altun, 2014). Robins, Routree, & Routree (2003) reported that experts working with programming have advanced knowledge, problem-solving skills, and are very good in areas such as mathematics and chess. Those who are new to programming are less likely to have these characteristics than experts (Winslow, 1996). In a study by Gülmez (2009), it was stated that programming education is an important factor in computer literacy, and it is effective in cognitive processes such as analytical thinking and problem solving (Sleeman et al., 1984). He also stated that mathematics achievement is related to the ability to understand, to interpret, to interpret, to question and to analyze. In addition, Nowaczyk (1983), in his research revealed that there was a positive correlation between the academic achievement of students in English classes and their computer programming successes.

What is instructional design?

The concept of instructional design (ID) covers a process as a field and discipline. Therefore, as well as student-centered, the problem of the learning problem as well as the systematic and systemic approach to the solution of the problem identified is of special meaning. In this context, different definitions were made by different field experts. According to these, ID is a systematic and responsive process of transfer of learning and teaching principles for teaching materials, activities, information sources and evaluation plans (Smith and Ragan, 2005). In another definition, ID management is based on what we know about information systems system design, teaching and learning theories (Morrison, Kemp & Ross, 2001). According to the designers of another approach that is effective for both classroom teaching and project management, instructional design is defined as “the process of solving learning problems by systematic analysis of learning conditions “as discipline and process” (Seels & Glasgow, 1998).

Cognitive loading theory

ID theories are the guiding principles in which teaching method will be used. (Reigeluth, 1999). The aim of cognitive load theory is to develop instructional design steps based on human cognitive architecture model (van Merriënboer & Sweller, 2005). This architect approach accepts the existence of limited short-term memory (STM), the presence of cognitive schemas in non-limited long-term memory. Within this framework, learning is defined as a structure and automation within the scheme. The tasks that need to be carried out in the cognitive process are real as a direct function of the complex task that takes place and directly affect learning. In addition, as a result of the unnecessary process that does not affect learning, it can be mentioned that it does not contribute directly to learning, and it may be the healing dimension of true cognitive learning. Therefore, the programming process has to take advantage of the design principles and strategies in cognitive

processing theory. The diagrams defined in the 4C/ID model allow for the elaboration of complex thinking in the design of the programming process, and the ability to learn and implement the technical skills required for this and even offer the opportunity for automation. The operation of this software is indicated below.

Four-Element Instructional design model (4C/ ID) and complex skills in the programming process

The 4C/ID model, developed by Van Merriënboer (1997), is the recommended instructional design model for the development of technical skills. The basic assumption is that plans prepared for complex learning can always be identified by four basic components (Figure 1). Software skills also require technical and analytical skills such as thinking, analyzing, substituting values, calculating the necessary variables and establishing relationships between them. In addition, complex learning is a learning process that aims to integrate knowledge, skills, skills and attitudes that perform high-performing tasks. These complex skills are explored through four different components in the model as different learning activities in the programming process. These steps are used in the software (programming) process to eliminate some steps, even if not in order. For example, in programming, the subject of loops is based on the repetition of the necessary tasks, ie the repetition of the skill. For this reason, new loops are written and coded from simple to difficult variables. This also involves looping tasks and tasks, such as an automatic operation, and reinforces repetition. The supportive nature of the information for the programming process is effective in creating cognitive skills and mental model. Supporting knowledge creates a bridge during the design of learning tasks with the knowledge of the learners and provides the student with a clear understanding of the software that will be prepared to be equipped with these skills. As a methodological approach, the student repeats his/her technical knowledge and skills, proceeds step by step and continue with specialization. Methodological knowledge can provide the learning of the routine aspects of the learning tasks carried out by the learners, and the place and time should be presented when the student is in full need. Learners are expected to gain behavior as they become experts. Thus, it can make the technical skills functionally more easily by bringing analytical concepts to more easily. With partial task practice, it is necessary for ordinary learners to have a lot of practice in order to improve the habits of learners.

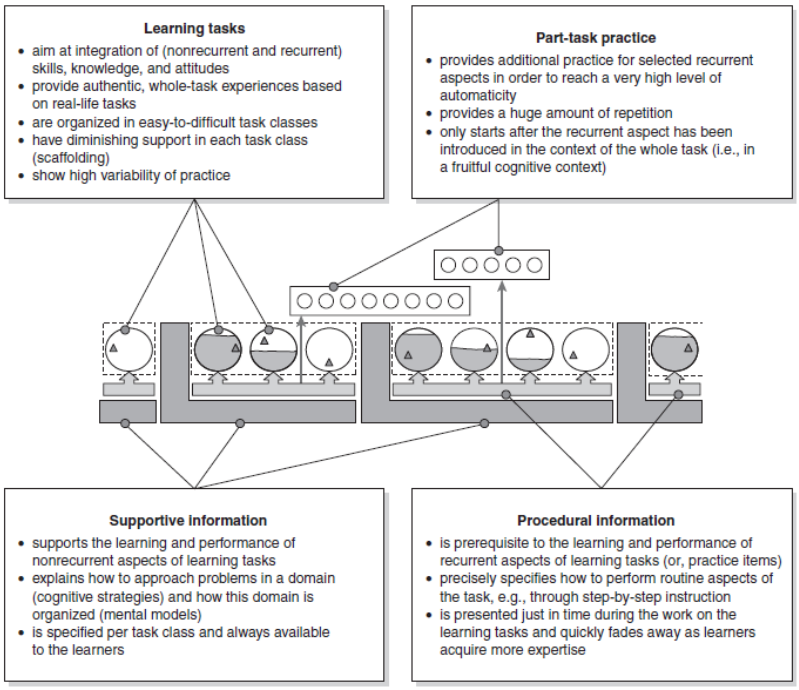

Figure 1. Basic Components in Four-Component Instructional Design (4C/ID) Model Schematic teaching design (Van Merriënboer & Kirschner, 2007)

Ten steps for Complex Learning (Van Merriënboer & Kirschner, 2007) is a practical and improved version of the four-component teaching design (4C/ID) model, recently made by van Merriënboer. The ten steps described here are mainly based on more rules, and teachers aim to provide practicality by practicing (4C/ID model) for field experts and less experienced instructional designers interested in education or training design. Learning the skills required for the subjects in each chapter can be designed within programming courses, learning a whole skill set based on these categories (Figure 2).

Basic Components of Four-Component Instructional Design (4C / ID) Model and Ten Steps for Complex Learning	
Basic Components of the Model	Ten Steps to Complex Learning
Learning Tasks	1. Learning Tasks for Design 2. Sequential Task Classes 3. Setting Performance Targets
Supporting Information	4. Supporting Information Design 5. Analyzing Cognitive Strategies 6. Analyzing Mental Models
Methodology	7. Methodical Information Design 8. Analysis of Cognitive Rules 9. Prerequisite to Analyze Information
Partial Task Applications	10. Partial Task Application Design

Figure 2. Steps in teaching programming skills with Four-Component Instructional Design (4C/ID) (van Merriënboer & Kirschner, 2007).

First, the task of learning is the experience of performing the original, complete task based on real-life tasks that focus on integrating skills, knowledge and attitudes. For this reason, using an instructional design model, defining the logical paths and technical skills in the programming process can be performed by analyzing the students by using algorithmic methods with the motivation of the students to the program software (Branch & Dousay, 2015).

The basic principle of ID theories, which aims to realize instructional design for complex learning, is to provide students with authentic-artistic tasks. Real missions are "tasks that are relevant and useful to the real world, which integrate these tasks in the curriculum, provide appropriate levels of complexity, and allow students to choose appropriate levels of difficulty or participation" (Jonassen, 1991). In order to design a directive, authentic tasks lead the student to solve complex real-world problems. The general assumption here is that such tasks help students integrate the knowledge, skills and attitudes necessary for effective task performance; it gives them the opportunity to learn to coordinate the founding skills that make up the complex task performance, and ultimately transfer them to their daily lives or work environments. The focus is on authentic and practical tasks such as project-based training, event methodology, problem-based learning and competency-based learning; In the process of programming and teaching of technical skills, the teacher can enable the use of scenario theory based on the goal of the course and the students can develop (van Merriënboer et al., 2003). A stream of software developed within this framework can serve to develop software skills in terms of learning, organizing, and analytical thinking.

LEARNING WITH 4C/ID MODEL

Instructional design diagrams for sample programming

Preparing teaching instruction in accordance with instructional design models will enable high-level thinking skills that will contribute to the effective, efficient and engaging of teaching. High-level thinking and learning skills; It is a process that determines the lifestyle (culture) and social interaction which constitute the process of problem solving and decision-making according to the purposes, which involves many mental processes based on inquiry. For this reason, design and design

of tasks and tasks in the instructional design model, design skills for the subject, ordering of supporting information, analysis of cognitive and mental paths, as well as access to methodological information, as well as solutions of cognitive rules, are designed with teacher partial applications in the software process. Thus, students can realize complex skills in adaptation and transformation behaviors at the stage of learning, for example, schemas and loops as concrete technical skills.

Many different skills in the process of gaining programming skills; logical thinking, algorithm formation, problem solving skills, analytical thinking skills can be gained. The identification, analysis and design of these stages are among the universal objectives of instructional design (IT) models. At the end of these processes, it provides the concrete solution of each programming step, the analytical solution of the processes in the software process, the strategies to be followed and the logical approach, in short, it contributes to the solution of the problem that the individual or the society needs and provides a solution for the solution. The student-centered instructional design process provides the opportunity to identify and systematically implement and evaluate solutions. In this context, the new generation requires individuals to use their high-level thinking skills, to think systematically, to look at the problems from different perspectives and to produce solutions, to create a cause-effect relation and to think creatively. Because it is the process of developing and implementing the algorithm that will serve to solve a problem due to its programming structure (Akçay & Çoklar, 2016). It also supports and technical skills as 21. The most effective way of developing computational thinking in the 20th century is defined as computer programming skills (Lye & Koh, 2014).

The assumptions followed in the process of programming and gaining technical skills, four-part instructional design (4C / ID) model in the dimension of theoretical approaches (van Merriënboer (1997), technical skills learning and ten steps for complex learning (Van Merriënboer & Kirschner, 2007; Kirschner & van Merriënboer, 2008) can be used for the software course . If we examine the model of these steps from top to bottom; It consists of 4 layers (Ipek, 2004). For example, it requires specifying the basic skill for the delivery of a software subject, that is, the hierarchical relationship of the basic cognitive skills in layer 1. 2nd. It includes the analysis of skills, the algorithm and related information, which is the totality of knowledge required for the software subject in step. In 3 steps, understanding of software subject, limited coding and repetition of related skills are selected in the course of software course. 4. It is the selection of the learning environment in which the labs and computers are used to teach the software course. Thus, the implementation of the software course in accordance with the teaching principles and steps in the course plan process will be provided by learning strategies for the software subject. As a four-element instructional design (4C/ID) model, it aims at organizing, teaching and developing knowledge for complex learning environments (van Merriënboer, Clark & De Croock, 2002). The general structure of the model, the layers discussed and the interrelations of the other sections are given in Figure 3.

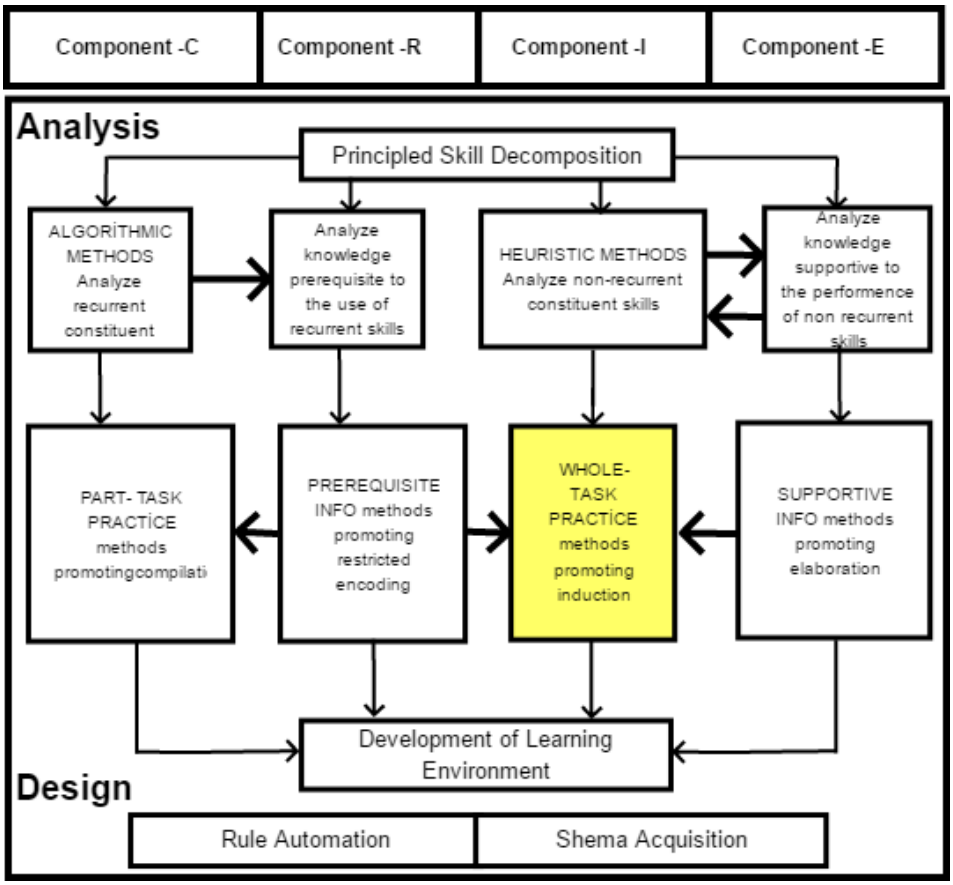

Figure 3. General structure of four-component instructional design model (4C/ID-Model) (Van Merriënboer & Kirschner, 2007)

The layers 1 and 2 of these layers express the analysis of complex cognitive skills. Likewise, layers 3 and 4 indicate the instructional strategy design or learning environment for this skill. Any skill can be separated or separated within the hierarchy of the skills that make up the whole. The activities performed in the analysis section are listed under part C (item-section). Part C, especially before the activities of the R segment. In the design part (layer 3), the activities are listed below the I part. The activities of the I part come particularly before all the other 3 parts. But there is no certainty in this order. The sections in Figure 3 refer to the main characteristics of the model (İpek, 2004).

- Compilation (C). Express the algorithmic task analysis in the second layer of the model. Here are the procedures and special rules. The programming process demonstrates the ability to use these rules. This information is classified as a continuum of all formed skills. Instructional process is designed to support and contribute to the functioning of procedures or the competence of ongoing skills through compilation of information.

- Restricted encoding (R). For the second layer, this section shows the analysis of the information. Here are facts, concepts, plans and principles. These skills are classified as continuing skills. In the third layer, it refers to the selection of teaching methods for all practical or practical tasks. The teaching process is designed to support and contribute to the automatic functioning of the ongoing skills and the limited coding of procedures. Teaching is a prerequisite for the ongoing aspects and appearance of skills.

- Elaboration (E). It refers to the analysis of information for the second layer. This analysis includes conceptual models, goal-plan hierarchy, cause-effect relationship, mental models. In the third layer, all task practice refers to the selection of teaching methods for its scope. With

the elaboration of the information, instructional design is planned to support the realization of the cognitive scheme.

- Induction (I). For the second layer, it discusses task analysis with heuristic learning and presentation with artistic discovery or experience. These are the results of artistic activities to solve problems, systematic approaches. These task analyzes are classified to determine the appearance and direction of complex cognitive skills. The third layer focuses on the selection of teaching methods for the practice of all tasks in the process. Instructional design is done to support the realization of cognitive schema through abstract problems or examples.

As described above, the C and R divisions relate to the category of learning processes. They are shown as rule automations. Parts E and I are related to the category of learning processes and are recognized as the realization of the scheme. This step is also the heart of the model. Based on this model, the learning program is developed as a problem-based, event-based or a scenario-based instruction. For this reason, this model has the feature and strength to be an instructional design model that can be used for the structural approach, especially for cognitive weight.

Complex learning and instructional design model (4C / ID Model)

Today's instructional designs are known to be linear, slow in progress and require a lot of energy. For this reason, design-oriented, creative technology-based designs (Driscoll & Dick, 1999; İpek, 2002)should be highlighted in the working life and especially in the industry with complex information in order to guide the rapid advancement in technology with creative ideas.

In line with the systematic design, by designing alternative project loops for the future by focusing on creative and instructional purposes towards creative and requirements, virtual simulations of possible cultural changes can be designed with prototype project loops. The scope of this process can then be transferred to active projects and implemented. As a result of the existing developments, complex information is required in the workplace (van Merriënboer, Clark & Crook, 2002). The systematic design research and development studies carried out over the last two decades have focused particularly on teaching principles.

Instructional Design approach for complex cognitive skills

Here, the design and creation of learning environments for complex cognitive skills is discussed. In the teaching system for teaching any skill, there are differences between the design of the presentation of information and the design of the application (practice). First, the information follows a ranking that has not been finalized for the application of skills. On the macro level, skills are related to the multiplication of holistic skills in the appropriate curriculum. The order at the “meso” level, which is the sorting of the event types, occurs at the macro level. In this sequence, the meso-level ordering is a set of examples of the problems and the expression of event types for each skill. The second step concerns the problem-solving process of systematic approaches. This performance is all complex skills or their meaningful perspectives. The heart of this model is the application design of all tasks. The samples studied in the design period, the problems in different formats are informed to the student for each case study in micro level. In this section, briefly, these issues come to the fore (İpek, 2004).

Part-task practice is supported and encouraged by the rule autonomy compilation time. These are briefly stated as follows.

- This task is not always required. This design rule saves automation and compilation appearance. This is exactly the same as in the operation of the flow diagram.

- For the complex algorithm, the application agents may be in the form of an example of different ordered-whole approaches sorted. For example, it can express a functional process in terms of fragmentation (segmentation), simplification and the ability to explain and apply the concepts of classifications such as decimal fractions. We can apply and demonstrate these concepts with the following examples.

- At the right time, the presentation supports the discovery of ongoing holistic skills with high-level information, which is the special rules. The purpose of this presentation is to apply complex skills to the ongoing appearance during limited coding. It introduces the special rules to the student and provides the student to gain this technical skill as a presentation in the process of introducing the variables to the variables and introducing the sample concepts. Just in time, the process consists mainly of processes and rules. Therefore; a) rules and processes define the accuracy of the performance of ongoing skills. b) There are also behaviors, concepts, plans and principles. These are prerequisites for learning. c) The process of rotation indicates the quality of the performance. If the instructional designer cannot control the problems of all tasks, the students' learning assistance systems should be developed. Demonstration and sampling should be used as teaching techniques. Just-in-time informational feedback provides information about correct and incorrect results. This process results in different types of misconceptions and values entered in the teaching of concepts. Even at this point, students continue to gain a technical skill. As a result, they can learn as a technical skill (non-working code) by experimenting with the knowledge that problem solving is not realized with these variables.

Another aspect of the model is to define instructional strategies and tactics for strategic information and supporting presentations. For this parsing and comprehension, the whole event-sample case, deductive-demonstration strategy, induction-case study and induction-demonstration strategy techniques are used. The aim of the strategic and supportive presentation is the elaboration of the materials shown by the student and the formation of schemes. These presentations consist of feedback on the quality of performance, modeling of event situations and instances. In the event that the desired tasks are not fulfilled, the Four-component instructional design model proposes the technique in which induction and models are presented as an approach. In addition to this, if students have high level of pre-knowledge and adequate teaching time, they can use the approach of developing and developing models. They can use the types of concepts to be taught (eg, loops, control, function, etc.) by applying them to gain the ability to learn to use. In this model, in general, case studies are used for different knowledge categories of teaching tactics. In addition, after the problems involving all tasks in elaboration-understanding, feedback is given for complex cognitive skills. The types of returns include functions such as motivation, the subject of the return and the time of return. In some cases, it is possible to see examples of the subject that is planned to be taught in a software process piece, to gain application technical skills, to gain knowledge and skills. These affect learning environments in digging skills using other variables in the software process. In the development of learning environments, steps such as the selection of teaching materials, the development of the learning environment, the transfer of effects and the dynamism of the systems are very important (van Merriënboer, 1997).

This model generally does not guide in the production of teaching materials in very detailed. Detailed instructional design models can be used for this (Dick & Carey, 1996; Ipek, 2001; Seels & Glasgow, 1998; Smith & Ragan, 1999). In order to access the desired learning, the choice of material is influenced by the process of the curriculum, the specific conditions and the contradictions in the characteristics of the target groups. The priority materials used in the implementation of all tasks are used in learning or in the learning environment where the tasks are based on simulation. Production and process-centered samples are combined based on simulation, problem, case study, or scenario-based environments. In this model, the interpretations and procedures obtained in the learning environment provide more performance transfer than traditional teaching as the transfer of information for teaching. This model has a systematic approach. Input and output (product) relationships are among the clusters of activities that are part of the model. The creation of the programming process

shows meaning and similarity with the realization of the input and output relationship. In addition, it supports systemic approaches to instructional design. Because this model allows changes between analysis and design parts and changes with various programming codes between activity clusters. The use of these processes in the programming reveals the function of this model.

CONCLUSIONS AND SUGGESTIONS

The model described in this study is based on experimental studies in the field of instructional technology. The 4C/ID model is the result of a prolonged study that began in the 1980s as a process of developing technical and complex skills. There are experimental evidence for many teaching strategies and tactics proposed by this model. This model is based on two basic procedures. One of them is the implementation of the model based on deep expertise and better performance. Secondly, model-based learning environments will be superior to traditional learning environments. This is due to the fact that the transfer duties of the advanced education are different from the tasks given by the training. This study shows that the basic functions included in the model overlap the technical in complex skills of the software-programming process. The characteristics of the model, which the numerical and mental thought contributes to the development of technical skills, such as analyzing, providing automation and creating schemas, are also important for concepts and mental skills that are essential in the learning and application of technical skills. The programming process therefore needs to develop complex and cognitive skills. At the end of this study, the instructional design model approach's teaching strategy steps are effective in teaching to develop software-programming skills.

As with information and task analysis in the second step of the model, mental models are crucial for the performance of complex cognitive skills. At another level, the selection and the tactics of teaching strategies are included in each of the four stages of the model. In the last step of the model, the development of learning environments seems to be a difficult task in terms of both time expenditure and difficulty in starting from a curriculum draft. It is feasible to implement instructional strategies and tactics with computer teaching models, gain programming skills, and analyze complex skills, and design and develop programming for different environments of learning in a software environment.

The 4C/ID Model gives designers the opportunity to solve problems and learn more about complex cognitive skills that do not continue as a teaching model approach. Some code in the programming is continuous and maintains the process. Some terminate the process for the given input and value. The basic element here is that all the desired tasks focus on the appropriate design in order to achieve the goals. For the complex learning process, this model explains the ease of use of many suitable models in terms of learning skills, learning problem solving function. The main thing is to design all of the tasks in learning all the ongoing or non-cognitive skills. One of them is the problem solving process in the realization of the programming of the individual. As mental models, what is the problem, how it is organized and how a job is done is important teaching questions. In this respect, the concepts of model and software development process discuss a new approach as a thought by revealing the importance of industrial institutions, businesses and organizations for the solution of problems that require complex cognitive skills as well as scientific techniques, tactics and suggestions for designing effective learning environments.

As a result, with an Instructional Design Model which can be used for gaining complex and cognitive skills, the operation, structure and techniques that can be used in vocational and technical education are explained with concepts. In this process, the ID Model discussed in this process is based on new technologies and the relationships between the programming system and behavioral-cognitive and structural teaching approaches (moderate-radical) was tried to explain as possible and possible relevance, thought development skills have been revealed. Thus, it has been pointed out how this model can be used to realize permanent, effective and interactive teaching in the field of programming. In addition to how the functions of new instructional designers can be realized for the future and the integration of other models and software development education can be more effective, the model

approach mentioned here in terms of gaining new complex, mental and cognitive skills has been put forward in terms of software development.

REFERENCES

Akçay, A., & Çoklar, A. N. (2016). Bilişsel becerilerin gelişimine yönelik bir öneri: Programlama eğitimi. *Eğitim Teknolojieri Okumaları*.

Akpınar, Y. & Altun, A. (2014). Bilgi toplumu okullarında programlama eğitimi gereksinimi. *Elementary Education Online, 13(1),* 1-4.

Ambrosio, A. P., Costa, F. M., Almeida, L., Franco, A., & Macedo, J. (2011). Identifying cognitive abilities to improve CS1 outcome. Paper presented at the Frontiers in Education Conference (FIE), 2011.

Amorim, C. (2005). Beyond Algorithmic Thinking: An Old New Challenge for Science Education, Eighth International History, Philosophy, Sociology & Science Teaching Conference, July 15 to July 18, 2005, University of Leeds, England

Branch, R. M.& Dousay, T. A. (2015). *Survey of instructional design models*, Fifth edition, Indiana, AECT.

Byrne, P. & Lyons, G. (2001). The effect of student attributes on success in programming. *ACM SIGCSE Bulletin, 33(3),* 49-52.

Clements, D. H. & Gullo, D. F. (1984). Effects of computer programmimg on young children's cognition. *Journal of Educational Psychology, 76(6),* 1051-1058.

Dick, W. & Carey, L. (1996). *The systematic design of instruction (4th Ed.).* New York: HarperCollins College Publishers..

Doğanay, A. (2007). Üst Düzey Düşünme Becerilerinin Öğretimi, A. Doğanay (Ed.) *Öğretim İlke ve Yöntemleri*, Ankara, Pegem Akademi. Yayı.

Driscoll, M. & Dick, W. (1999). New research paradigm in instructional technology: An inquiry. *Educational Technology Research and Development, 47(2),* 17-18.

Ennis, R. H. (1993). Critical thinking assessment. *Theory into Practice, 32(3),* 179-186.

Erdoğan, B. (2005). *Programlama başarısı ile akademik başarı, genel yetenek, bilgisayara karşı tutum, cinsiyet ve lise türü arasındaki ilişkilerin incelenmesi*. Yüksek Lisans Tezi, Marmara Üniversitesi Eğitim Bilimleri Enstitüsü, İstanbul.

Ertürk, S. (1972) *Eğitimde Program Geliştirme.* Ankara, Yelkentepe Yay.

Gagné, R. M. (1985). *The conditions of learning and theory of instruction.* New York: Holt, Rinehart and Winston.

Gülmez, I. (2009). *Programlama öğretiminde görselleştirme araçlarının kullanımının öğrenci başarı ve motivasyonuna etkisi*. Yüksek Lisans Tezi, Marmara Üniversitesi Eğitim Bilimleri Enstitüsü, İstanbul.

Hostetler, T. R. (1983). Predicting student success in an introductory programming course. *ACM SIGCSE Bulletin, 15(3),* 40-43.

İpek, İ. (2001). *Bilgisayarla öğretim: Tasarım, geliştirme ve Yöntemler*. Ankara: Tıp ve Teknik Kitapevi Yayınları.

İpek, İ. (2002). Öğretim Tasarımı Sistemlerinde Gelişmeler, Yaklaşımlar ve Öğretim Teknolojisinde İlerlemeler. XI. Eğitim Bilimleri Kongresi, 23-26 Ekim 2002 Yakın Doğu Üniversitesi, Lefkoşa, KKTC.

İpek, İ. (2004). Karmaşık Bilişsel Teknik Becerilerin Öğretilmesi Sürecinde Öğretim Programı Tasarımı: Dört Ögeli-Öğretim Tasarım Modeli. IV. International Educational Technologies Symposium. 24-26 November, 2004. Sakarya University. Sakarya.

Januszewski, A. & Molenda, M. (2008a). *Educational Technology, "A definition with Commentary",* New York: Lawrence Erlbaum Associates.

Januszewski, A. & Molenda, M. (2008b). Definition. In A.Januszewski & M. Molenda (Eds.), *Educational Technology: A definition with commentary*. New York: Lawrence Erlbaum Associates.

Jonassen, D.H. (1991). Objectivism vs. Constructivism. Do we need a philisophical paradigm shift? *Educational Technology Research and Development. 39(3),*5-14.

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, 46(2),* 75–86.

Kirschner, P. A. & Van Merriënboer, J. J. G. (2008). Ten steps to complex learning: A new approach to instruction and instructional design. In T. L. Good (Ed.), *21st century education: A reference handbook (pp. 244-253).* Thousand Oaks, CA: Sage http://web.mit.edu/xtalks/TenStepsToComplexLearning-Kirschner-Van Merriënboer.pdf)

Lau, W. W. F. & Yuen, A. H. K. (2011). Modeling programming performance: Beyond the influence of learner characteristics. *Computers & Education, 571(1),* 1202-1213.

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.

Nelson.T.O. (1999). Cognition Versus Metacognition, in;P.J. Sternberg(Ed), *The Nature of Cognition, 625-641,* Cambridge,MaA:MIT Press.

Morrison, G. R. Ross S. M. & Kemp, J. E. (2001), *Designing Effective Instruction*. Michigan University, Wiley/Jossey-Bass education,

Newsted, P. R. (1975). Grade and ability predictions in an introductory programming course. *ACM SIGCSE Bulletin, 7(2),* 87-91.

Nowaczyk, R. H. (1983). Cognitive skills needed in computer programming. (ERIC Document Reproduction Service No. 236466).

Reigeluth, C.M. (1999). *Instructional design theories and models: A new paradigm of instructional design (vol.2)*. Mahvah, NJ: Lawrence Erlbaum Associates.

Robins, A. Rountree, J. & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13(2),* 137-172.

Saygılı, G. (2010). *Öğretim teknolojilerinin fen ve teknoloji dersinde kullanımının ilköğretim Öğrencilerinin problem çözme becerilerine öğrenme ve ders çalışma stratejilerine üst düzey*

düşünme becerilerine fen ve teknoloji dersine yönelik tutumlarına ve ders başarısına etkisinin incelenmesi. Doktora Tezi, Dokuz Eylül Üniversitesi Eğitim Bilimleri Enstitüsü, İzmir

Seels, B. & Glasgow, Z. (1998). *Making instructional design decisions (2nd Ed.)*. Upper Saddle River, NJ: Prentice Hall, Inc.

Seels, B., & Richey, R. (1994). *Instructional technology: The definition and domains of the field.* Washington DC: Association for Educational Communications and Technology.

Sleeman, D., Putham, R. Baxter, J. & Kuspa, L. (1984). Pascal and high-school students: A study of misconceptions. Technology panel study of stanford and the schools. (ERIC Document Reproduction Service No. ED258552).

Smith, P. L. & Ragan, T. J. (1999). *Instructional design. (2nd ed.)*. Upper Saddle River, NJ: Prentice Hall.

Smith P. L. & Ragan, T. J. (2005). *Instructional Design (3rd Ed.)* John Wiley & Sons, Inc.

Türk Dil Kurumu, (2018) (Turkish Language Society) Türkçe Sözlük. Retrieved January 2018, fromhttp://tdk.gov.tr/index.php?option=com_bilimsanat&view=bilimsanat&kategoriget=terim&kelimeget=d%C3%BC%C5%9F%C3%BCnme&hngget=md

Van Merrienboer, J. J. G., and Paas, F. G. W. C. (1990). Automation and schema acquisition in learning elementary compute programming: Implications for the design of practice. *Computers in human behavior 6.3* 273-289.

van Meriénboer, J.J.G. (1997). *Training complex cognitive skills: A four-component instructional design model for technical training.* Englewood Cliff, NJ: Educational Technology Publications.

van Meriénboer, J. J. G. ,Clarck, R. E. ve de Crook, M. B. M. (2002). Blueprints for complex learning: The 4C/ID-Model. *Educational Technology Research and Development.50(2),* 39-64.

van Merriënboer, J. J. G., Kirschner, P. A., & Kester, L. (2003). Taking the load of a learner's mind: Instructional design for complex learning. *Educational Psychologist, 38(1),* 5–13.

van Merriënboer, J. J. G. & Kirschner, P. A. K. (2007). *Ten steps to complex learning: A systematic approach to four-component instructional design*. NJ: Lawrence Erlbaum Associates, Inc.,

van Merriënboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review, 17,* 147–177.

Whipkey, K. L. (1984). Identifying predictors of programming skill. *ACM SIGCSE Bulletin, 16(4),* 36-42.

Winslow, L. E. (1996). Programming pedagogy – A psychological overview. *ACM SIGCSE Bulletin, 28(3),* 17-22.