Available online at www.sciencedirect.com
ScienceDirect
Computers & Education xxx (2007) xxx–xxx
COMPUTERS & EDUCATION
www.elsevier.com/locate/compedu

A computer environment for beginners' learning of sorting algorithms: Design and pilot evaluation

Maria Kordaki [a,*], Micael Miatidis [b], George Kapsampelis [a]

[a] *Department of Computer Engineering and Informatics, Patras University, 26500, Rion Patras, Greece*
[b] *Lehrstuhl Informatic V, RWTH, Aachen, Germany*

Received 17 July 2006; received in revised form 19 July 2007; accepted 22 July 2007

Abstract

This paper presents the design, features and pilot evaluation study of a web-based environment -the SORTING environment- for the learning of sorting algorithms by secondary level education students. The design of this environment is based on modeling methodology, taking into account modern constructivist and social theories of learning while at the same time acknowledging the role of hands-on experience, the significance of students' expressing their previous knowledge, the importance of interlinked multiple representation systems (MRS) and the role of constructive feedback on student learning. Although SORTING supports student learning of typical sorting algorithms such as Bubble-sort, Quick-sort and Selection-sort, it can also be adapted to integrate more sorting algorithms. The analysis of the data emerging from the pilot evaluation study of SORTING has shown that students used all the representation systems (RS) provided and found them attractive and easy to use. On the whole, student interactions within SORTING helped them to become aware of both the intuitive and the typical sorting procedures used, to conceptualize them, to overcome learning difficulties, to correct themselves and to make connections between different representations of the sorting algorithms used.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Secondary education; Interactive learning environments; Improving classroom teaching; Sorting

1. Introduction

Sorting is a significant universal activity in all cultures; associated with financial and commercial life, it has strong social currency, due to its basis in comparisons performed on countable entities of interest and value (Bishop, 1988). Sorting is also an element in science and technology as well as everyday life (Bishop, 1988; Knuth, 1973; Linderstrom & Vitter, 1985). It is worth noting that 25% of the total active time of computers is dedicated to sorting procedures, while the time used for account-sorting in large financial institutions is about 2 hours per day (Knuth, 1973; Linderstrom & Vitter, 1985). The general value of sorting is based on the fact that sorted entities usually offer opportunities for optimistic solutions to a variety of problems, thus

[*] Corresponding author. Tel.: +30 2610 993102; fax: +30 2610 997751.
E-mail address: kordaki@cti.gr (M. Kordaki).

0360-1315/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2007.07.006

necessitating automation of the sorting process using computers and, subsequently, the invention of fast sorting algorithms such as Bubble-sort, Selection-sort, Quick-sort, etc. (Knuth, 1973; Linderstrom & Vitter, 1985). Consequently, sorting is a necessary subject in any Computer Science Curriculum at both secondary and tertiary levels of education (ACM, 1991, 2003; Gal-Ezer & Zur, 2004).

However, understanding algorithms, especially computer-based sorting algorithms, is difficult for secondary level education students; this involves complex procedures based on different layers of abstraction and, therefore, understanding them requires students to use their higher mental functions (Matsaggouras, 1997; Vlachogiannis, Kekatos, Miatides, Kordaki, & Houstis, 2001). In fact, students have difficulties in understanding how a specific sorting algorithm works and also how this algorithm is described using basic algorithmic structures. Despite this, sorting algorithms are rapidly introduced into the curricula of secondary level education, with, students being introduced directly to the Bubble sort algorithm through its representation in pseudo-code using Pseudo Pascal. As a result, the sorting algorithm in focus becomes a meaningless procedure which, in most cases, students can only memorize rather than understand.

Understanding of typical sorting algorithms is a process of giving meaning to its various representations, such as those which are text-based using natural language, those text-based using specific expressions in the imperative, pseudo-code based representations and finally those in a specific programming language (Vlachogiannis et al., 2001). It is also acknowledged that student learning through hands-on experience while sorting concrete materials is significant for the learning of typical sorting algorithms (Geller & Dios, 1998). In addition, giving meaning to typical sorting algorithms pre-supposes an understanding of sorting from each individual student's point of view (Vlachogiannis et al., 2001).

The provision of animation materials has been proposed as essential to overcoming student difficulties with learning sorting algorithms (Kann, Linderman, & Heller, 1997). In fact, a number of animation materials for the learning of sorting algorithms have been reported, including educational software (ES). Some well-known examples are: (a) 'Sorting out Sorting' (Baecker, 1981), based on video presentations, introducing the idea of representing data elements in a sort by bars, with the length of the bar corresponding to the sort key value of the element. These bars are presented in a sort using 9 sorting algorithms, (b) 'Balsa' ES (Brown, 1988), also based on visualization of sorting entities, but a variety of sorting algorithms can be used simultaneously, (c) Xtango ES (Stasko, 1990), again supporting visualization of sorting entities using a plethora of sorting algorithms. In addition, users can interact with these visualizations (and stop the procedure at any time) as well as create and run their own sorting algorithms, (d) 'GAIGS', web-based ES (Naps, 1990), which supports algorithm visualization as well as text-based information about sorting, and (e) Sort Animator (Dershem & Brummund, 1998), which supports visual presentation of the sort animation demonstrated in relation to its code. Students are allowed to specify different colors for the sort animation, the number of elements being sorted and the speed of the animation and its execution. In addition, lines of code are highlighted as they are executed by the animation. All the aforementioned ES are based on presentations (in visual or textual form) of an algorithm in execution and are mainly dedicated for use in higher education.

As for the learning of sorting algorithms, and despite the above, ES have not yet been reported that take into account student difficulties and provide them with opportunities to express their own approaches to sorting and to typical sorting algorithms as well as to interpret these approaches in a variety of representation systems and also to actively perform sorting procedures using hands-on experience while simultaneously receiving constructive feedback.

Appropriately-designed educational software can act as a catalyst in the learning and teaching of all subjects, providing promising opportunities for the construction of new meanings that can be applied to the whole learning process (ACM, 1991; Jonassen, 1999; Noss & Hoyles, 1996). In fact, many researchers acknowledge the essential role played by constructivist educational software provision in student learning, providing them with opportunities to: (a) obtain hands-on experience, (b) express individual knowledge in Multiple and Linked Representation Systems (MLRS), including their mistakes, (c) express knowledge in MLRS of different cognitive transparency, (d) explore dynamic representations, in terms of interactive animations of essential procedures regarding the learning subject and (e) receive constructive feedback (Ainsworth, 1999; Borba & Confrey, 1996; Jonassen, 1999; Kaput, 1994; Kordaki, 2005a).

Taking into account all the above, we constructed a web-based environment - the SORTING environment - aiming to provide students with opportunities to construct the concept of a sorting algorithm within a constructivist learning context through: (a) their active involvement in sorting entities using hands-on experience to express both their own intuitive sorting approaches as well as typical sorting algorithms provided by the literature and expressing these approaches in a variety of RS and (b) their exploration of interactive, dynamic representations (in the form of animations) of typical sorting algorithms in a number of linked RS. The features provided by SORTING are not only theoretically documented but have been tested in the field using actual students to verify their impact on student learning. Such a computer environment for the learning of sorting algorithms has not yet been reported.

The main research aims motivating this study are:

- to explore ways of designing educational software that effectively supports secondary level education students' learning of sorting and typical sorting algorithms;
- to examine the impact of the proposed software on students' conceptualization of typical sorting algorithms.

This paper is organized as follows: 1. The rationale for the design of SORTING. 2. The general architecture, the features of SORTING and a description of the activities that can be performed within this context. 3. Pilot-formative evaluation of this computer environment. 4. A discussion of the features of SORTING in relation to their impact on student learning. 5. Conclusions.

2. The rationale for the design of SORTING

A central and essential role in the design of a learning environment is played by the specific answers given - explicitly or implicitly- by its designers to vital questions such as what the subject of learning is and what basic concepts have to be learned by the students, and to this end which the essential tasks are that they have to perform, what the learners' profile is and how learners can learn this specific subject (Laurillard, 1993). Answering these questions is not a simple task, since it implies awareness of both learner profiles in their attempts to learn a specific subject of learning and also epistemological perspectives regarding both knowledge construction and the nature of the subject of learning in focus. As educational software is not only a technological artifact but also a learning environment the answers to these questions have to be explicitly addressed during its design. Of course, these answers and their respective interpretations in terms of design specifications are not unique but depend upon the designers' interpretations of the theoretical issues taken into consideration (Kaput, 1992).

In the case of SORTING, the respective answers to these questions were supplied through the construction of three models, namely; the student model, the subject-matter model and the learning model. In the context of the construction of each model, these theoretical answers were also interpreted in terms of software design specifications. A dominant role in the modeling process was taken by the construction of the student model, since the design perspective that was adopted emphasized a student-centered design (Land and Hannafin, 2000; Jonassen, 2000a). Computer modeling is used in education to support learning in artificial worlds (Kordaki & Potari, 1998; Mellar & Bliss, 1994), providing children with computer tools to enable them to create their own worlds, to express their own representations of the world and also to explore other people's representations. On the whole, the SORTING environment can be viewed as a synthesis of three models. As regards the construction of the learning model and of the subject-matter model theoretical considerations, provided by the literature, were taken into account while for the construction of the student model specific field studies - using real students- were performed. An early description of these models is presented in Kordaki et al. (2005). In the following section of this paper, the entire modelling process, including the theoretical considerations taken into account during the construction of these models, as well as interpretations in terms of operational specifications of the software, is presented.

2.1. The student model

In terms of student learning of sorting algorithms in secondary level education, there are a limited number of published works in this area. Consequently, the construction of this model was based on the difficulties

(12th grade, 18 year old) secondary level education students faced with regard to the Bubble sort algorithm that emerged from a field study (Vlachogiannis et al., 2001). In this study, a primary version of the software for the Bubble sort algorithm was used. Specifically, students seemed to encounter difficulties in understanding how a specific sorting algorithm works and how this algorithm is described using basic algorithmic structures such as iteration structures, control structures and assignment statements. Regarding iteration structures, students had difficulties in understanding the iteration pattern used in the comparison procedure, how nesting loops work and the meaning of both the variables and the counters used. In particular, students seemed to confuse the static meaning of a variable used in a mathematical equation (Khalife, 2006) with the dynamic meaning implied in a variable used in a computer-based algorithm. In addition, students seemed to lack the ability to initialize the counters used and usually paid no attention to initializing or terminating the whole sorting procedure. As for assignment statements, students seemed to ignore their dynamic character and tended to use them as mathematical equations. Finally, when the use of control structures was necessary, students found it difficult to determine the type, value and position of the conditionals needed. In general, it was acknowledged that, when beginners write programs, they frequently generate the source code without any organized thought process. They type their solutions seconds after the initial reading of a problem statement and iteratively proceed by modifying their source code as required until the program generates the correct outputs (Suchan, and Smith, as it cited by Khalife, 2006).

In addition, the data emerging from the previously mentioned study indicated that, in order to understand sorting algorithms, secondary level education students needed to: (a) sort entities using hands-on experience and receive immediate feedback, (b) reflect on their hands-on experience and try to gradually interpret this in a variety of representation systems including text using natural language and pseudo-code, (c) explore analytic animations of the sorting algorithms in focus that were automatically performed, and (c) experience practical sorting in combination with visual animations of sorting entities, dynamic visual flow-charts and the corresponding dynamic visual pseudo-code. With the above in mind, an attempt has been made to exploit the conclusions from the aforesaid study through the formulation of a number of operational specifications for the design of the SORTING environment. The specifications used for the construction of this environment are presented in Table 1 below:

Table 1
The student model of sorting: from theoretical considerations to design principles

The student model in performing essential tasks	
The tasks	Design specifications: Tools needed to help students to...
1. Sorting using student-based sorting procedures	Sort elements: • by using hands-on experience Interpret their hands-on experience in: • free-text • pseudo-code
2. Sorting using typical sorting algorithms	Sort elements: • automatically • by using hands-on experience Interpret their hands-on experience in: • free-text • pseudo-code Explore: • the pattern of iteration used in the comparison procedure • nested loops • conditionals in terms of their type, value and position • variables • counters and assignment statements • the initialization and termination of the entire sorting procedure

2.2. The subject- matter model

This model has been constructed by taking into account the knowledge necessary for the construction of a sorting algorithm and, more specifically, the basic sorting algorithms such as Bubble-sort, Selection-sort and Quick-sort. These sorting algorithms have been selected to give students opportunities to compare the logic and the speed among them and consequently to develop a broad view of sorting. In addition, these algorithms are part of the secondary school curricula in Greece. Each specific algorithm in focus consists of the following basic parts: (a) initialization and termination of the entire sorting procedure, (b) the variables and counters used, (c) the assignment statements used, (d) the kind of conditionals used, and (e) the iteration pattern used in the comparison procedure. Here, it is worth noting the importance of students comprehending that, although the entities to be sorted can be in a variety of forms, such as numbers, files, objects, etc., in each case all these entities must include measurable information. Understanding that the comparison between the entities to be sorted and their mutual transference is a fundamental operation implied in the sorting process is also significant.

Taking into account the results of the modeling performed in the previous section (Section 2.1), and to help students acquire the appropriate knowledge of all the aforementioned parts, it was decided to provide them with information in various forms and a variety of computer operations. The different forms of information are as follows:

- text-based information, explaining how each sorting algorithm in focus works.
- visual information, in the form of an analytic simulation of a sorting procedure where visual objects are sorted automatically using a typical sorting algorithm.
- pseudo-code, indicating how each typical sorting algorithm can be described in a language compatible with computers.
- visual information, in the form of a flow-chart indicating how the sorting algorithm in focus runs.
- interlinked information. Linking visual analytic simulation of the algorithm used with the corresponding flow-chart and pseudo-code.

Students can select to study the type of information they consider appropriate to their knowledge. The variety of actions provided by the software refers to the aforementioned parts and to the variety of RS provided. These actions are referred to in greater detail in the next section of this paper (Section 3) as they were produced by taking into account the students' model formed in the previous section (Section 2.1).

2.3. The learning model

The learning model was constructed after taking into account basic considerations of modern constructivist and social theories regarding knowledge construction (Duffy & Cunningham, 1996; Jonassen, 1991, 1999; Hannafin & Land, 1997; Vygotsky, 1978). Constructivist learning theories emphasize the active, subjective and constructive character of knowledge, placing students at the centre of the learning process. Rather than listening to the experiences or findings of others as summarized in texts or by teachers, learners have to actively carry out domain-related activities. Specifically, constructivist learning is based on students' active participation in problem-solving and critical thinking regarding a learning activity, which they found relevant and engaging (Hofstetter, 1997). Constructivist learning also stresses the need for student experimentation through acquiring hands-on experience while using primary sources of data (www.ncrel.org/sdrs/areas/issues/content/cntareas/science/sc500.htm). To this end, students have the opportunity to reflect on their experience and to articulate what they learnt. In general, by asking students to reflect on their learning they have opportunities to enhance their understanding regarding with the learning concepts in focus (Jonassen & Reeves, 1996). It is worth noting that, learners' experience, reflection, interpretation and meaning making are interconnected (Jonassen, 1999; Hein, 1999). Constructivist theory also proposes that students learn by constructing their own knowledge; thus, making mistakes is viewed as a learning opportunity. In the context of this theory, learning is idiosyncratic and therefore students have to be provided with opportunities to express their inter-individual learning differences. Rather than providing isolated information, constructivism emphasizes the role of intrinsic motivation where students are involved in 'authentic' real life learning activities related to their own interests (Bishop, 1988; Jonassen, 2000b; Nardi, 1996). Another value underlying constructivist learning is that individual beliefs and experiences provide a uniquely personal framework for new understanding. In fact, background knowledge and experience form the conceptual referent from which new knowledge is organized and assimilated (Piaget, 1976). Constructivism as a basis for instructional design pre-supposes student autonomy and minimal guidance in the form of process -or task- relevant information that is only available if learners choose to use it. Emphasis is also put on the role of different perspectives offered for the students to approach the learning task at hand (Wilson, 1997).

Social theories of learning stress the importance of psychological tools as mediators for the development of students' higher mental functions (Vygotsky, cited in Wertch, 1995). In particular, computer tools and learning environments can help students to express their own knowledge, to explore the knowledge of others integrated in these environments and also to provide appropriate feedback and to scaffold their thinking and actions in order to deepen understanding (Bereiter & Scardamalia, 1989; Noss & Hoyles, 1996). Tools can act as mediators of sociocultural development to help a student enhance their Zone of Proximal Development, which is essential in the development of their knowledge (Salomon, Globerson, & Guterman, 1989; Vygotsky, 1978). The mediating role of technology is based on the fact that computers can 'communicate' with learners through the variety of representations able to be constructed and displayed on their screens (Noss & Hoyles, 1996).

The role of representation systems in students gaining knowledge is acknowledged as significant by many researchers (Ainsworth, Wood, & Bibby, 1996a, 1996b; Ainsworth, 1999; Borba & Confrey, 1996; Dyfour-Q2 Janvier, Bednarz, & Belanger, 1987; ~~Kaput, 1987;~~ Kordaki, 2005a). In fact, it is acknowledged that MLRS can play a crucial role in student development of a broad view of the concepts in focus, as well as realization of the basic aspects and structure that constitute these concepts, translating and transforming a concept into the various RS provided and observing how variation in one RS can affect the other (Kaput, 1994; Schwartz, 1995). Students can also actively construct their own knowledge by experimenting with cognitive transparent RS. In addition, students can explore the knowledge of others by experimenting with 'cognitive opaque' dynamic RS. Moreover, the essential role of simulations which can be both automatically run by the system and manually handled by the students is emphasized (Davidovitch, Parush, & Shtub, 2006). In general, MRS can be used as supporting representations of students' mental internal representations related to a specific learning concept. In some cases, RS can be used as students' primary references to support their reflection processes for the learning of a specific learning concept (Kaput, 1989). MRS can also motivate students to be actively involved in their learning, especially when related to student interests (Ainsworth, 1999). The understanding of a concept can be also viewed as a process of giving meaning to its different representations as well as making connections between these different meanings. Furthermore, MRS can provide students with opportunities to make connections between the different kinds of knowledge they possess, such as intuitive, visual and symbolic knowledge. Finally, it is worth noting that it has been acknowledged that MR-based educational software plays a significant role in the learning of concepts regarding Computer Science and Engineering (Davidovitch et al., 2006; Kordaki, 2005b; Vlachogiannis et al., 2001; Zikouli, Kordaki, & Houstis, 2003).

Based on the above, an attempt has been made to interpret the aforesaid theoretical issues within the context of the computer, aiming at the formulation of a number of operational specifications necessary for the design of the SORTING environment. Table 2 provides a compact presentation of the theoretical considerations used (see Column 1) and how they were interpreted in terms of design specifications (Column 2), also taking into account the specifications addressed in the construction of the student model (Section 2.1). The resulting specifications are part of those used to construct SORTING. We do not claim that these interpretations are unique but they do express our own views regarding the aforementioned theoretical issues.

Providing secondary level education students with opportunities to sort measurable entities using hands-on experience and subsequently to reflect on their sorting actions is essential to organize their thinking and to their conceptualization of sorting algorithms, as in this way students can become aware of the sorting approach used (Du Feau, 1999). Encouraging also the students to understand sorting algorithms by expressing their individual sorting approaches and then gradually progressing to the use of typical sorting algorithms provided by the literature is also essential. In fact, each individual student sorting approach can play an

Table 2
The learning model: from theoretical considerations to design principles

Theoretical considerations	Design principles
The 'learning' model	
Encouraging learners to be active in their learning by: • Doing things • Solving 'authentic'' problems	*Providing students with the following functions…* • Interactivity of the RS used in the environment • Various tools to sort a number of elements from the students' world
Encouraging learners to express their inter-individual differences by expressing … • Their previous knowledge • Different kinds of knowledge	*Availability of tools to help students:* • Express their own knowledge by selecting the most appropriate RS for their cognitive development from among a variety of RS • Sort elements according to their own sorting procedures using MRS • Sort elements according to their views (including mistakes) about typical sorting algorithms using MRS
Encouraging learners to construct their knowledge by… • Acquiring hands-on experience • Experimenting • Reflecting • Experimenting with interlinked and dynamic RS • Enhancing students' ZPD	*Availability of tools to:* • Sort elements using hands-on experience • Sort elements using different sorting algorithms • Provide visual representations of sorting algorithms • Explore the knowledge of others included in dynamic RS e.g. visual animations of a sorting algorithm, dynamic flow-chart and dynamic pseudo-code RS • Provide appropriate constructive feedback on student sorting actions for self-correction • Provide appropriate scaffolding where students face difficulties

essential back-up role in supporting their construction of typical sorting algorithms. Moreover, the understanding of a sorting algorithm is viewed as a process of giving meaning to its representations in a variety of representation systems, beginning with sorting real life entities and progressing to explaining it in natural language, forming an appropriate flow-chart and consequently translating the whole procedure into pseudo-code and, ultimately, into a programming language (Vlachogiannis et al., 2001).

All the software specifications formed during the modeling process aiming the construction of the three aforementioned models described in this section were taken into account when implementing SORTING, the structure and features of which are presented in the next section of this paper.

3. Features of the ~~sorting~~ environment

The SORTING environment is organized in HTML pages, its features being presented on its main page and classified into three parts: (a) information, (b) workplace, and (c) animations. A diagrammatic presentation of the general structure of SORTING, including the RS provided, is presented in Fig. 1.

(a) *Information:* General information about sorting as well as specific information on the algorithms in focus is provided in the form of text.
(b) *Workplace.* Students are provided with four RS, in the context of which they are given the chance to express their own: (a) sorting procedures and (b) conceptions regarding typical sorting algorithms in focus and to receive immediate feedback from the system. The RS provided are:
- Simulated real objects (SRO).
- Text-based representation systems (TRS).
- Pseudo-code based representation systems (PCRS).
- Specific expression-based pseudo-code representation systems (SEPCRS).

The above RS are presented on the same page (see Fig. 2) and are described below:*Simulated real objects*
Students have the opportunity to experiment with a sorting procedure using hands-on experience. In fact, students can sort SRO familiar to them (e.g., Euro coins), firstly by using their own intuitive sorting algorithms

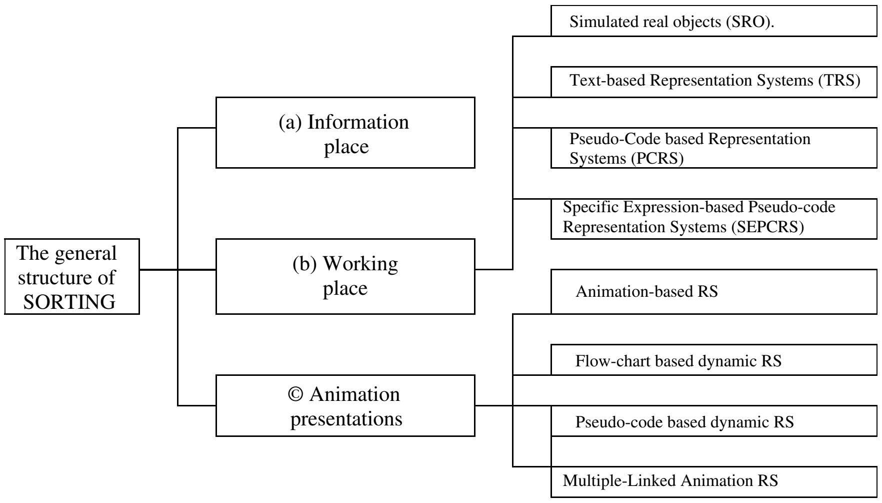
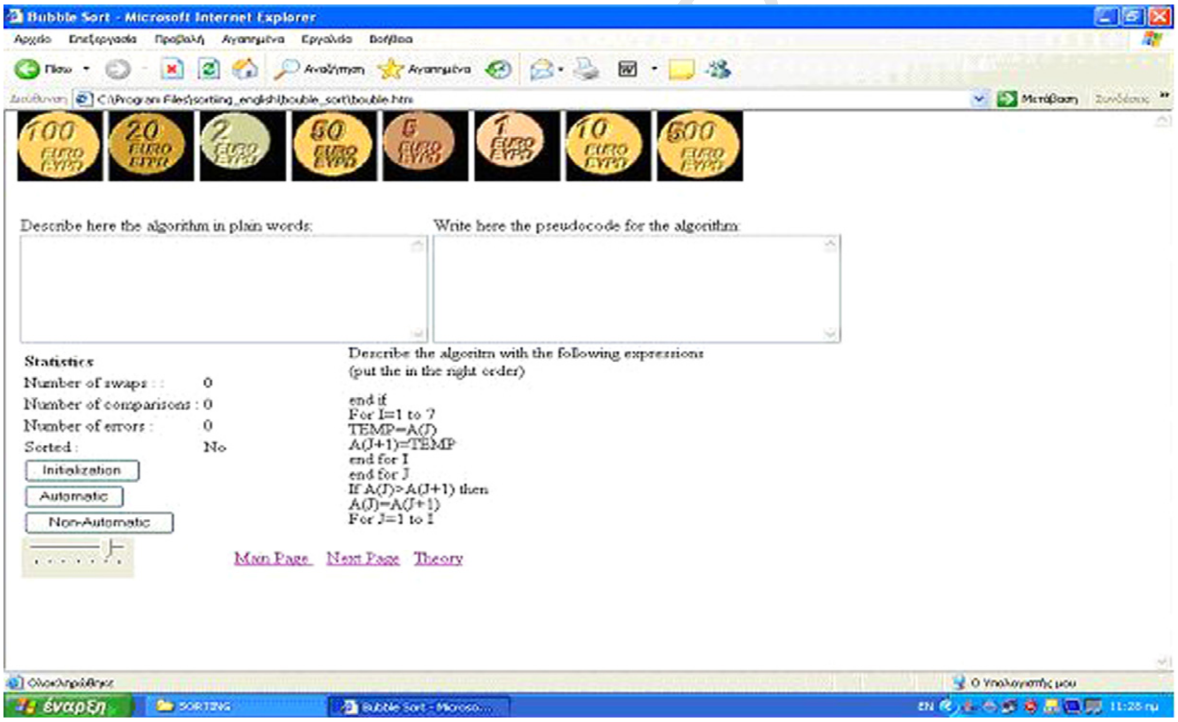

The general structure of SORTING
(a) Information place
(b) Working place
© Animation presentations
Simulated real objects (SRO).
Text-based Representation Systems (TRS)
Pseudo-Code based Representation Systems (PCRS)
Specific Expression-based Pseudo-code Representation Systems (SEPCRS)
Animation-based RS
Flow-chart based dynamic RS
Pseudo-code based dynamic RS
Multiple-Linked Animation RS

Fig. 1. The general structure of the SORTING environment. (a) Information place. (b) Working place © Animation presentations.


Bubble Sort - Microsoft Internet Explorer
C:\Program Files\sorting_english\bouble_sort\bouble.htm
Initialization
Automatic
Non-Automatic
Main Page
Next Page
Theory

Describe here the algorithm in plain words:

Write here the pseudocode for the algorithm:

Statistics

Number of swaps : :	0
Number of comparisons :	0
Number of errors :	0
Sorted :	No

Describe the algoritm with the following expressions
(put the in the right order)

```
end if
For I=1 to 7
TEMP=A(J)
A(J+1)=TEMP
end for I
end for J
If A(J)>A(J+1) then
A(J)=A(J+1)
For J=1 to I
```

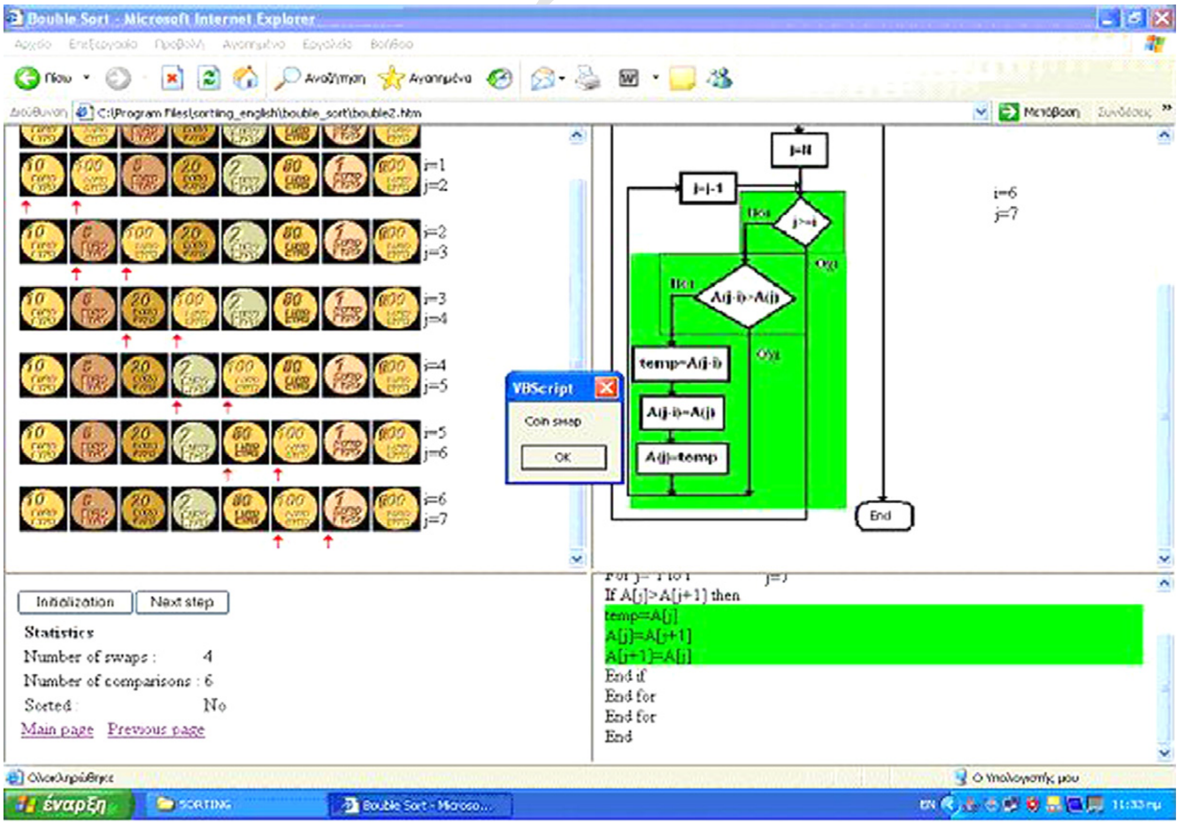
Fig. 2. The work place element of the SORTING environment.

and, secondly, by using the typical sorting algorithms in focus. In this way, students have the opportunity to reflect on their sorting actions and to be aware of the sorting algorithms they use. The system can initiate the coins automatically on clicking the appropriate button. The system also counts the number of swaps, comparisons and all incorrect sorting actions performed when a typical sorting algorithm is used. When a student performs an incorrect sorting action, they receive an automatic response from the system. In addition, the system sends a message when the list of coins is sorted. *Text based representation systems.* After experimentation and reflection on a specific sorting procedure or algorithm used to sort the coins provided, as described in the previous section, students are asked to interpret their experience using a text-based representation system. This RS has been selected to provide students with the opportunity to reflect on their sorting actions and come up with an informal description of the sorting algorithm used. By trying to formulate this description, students have the opportunity to be more conscious about the sorting algorithm they are experimenting with. *Pseudo-code based representation systems*. Students are asked to translate into pseudo-code the free-text description of the sorting algorithm they used. By using pseudo-code, students have the opportunity to move from the 'I' situation of the procedure, where they describe how they performed a specific sorting procedure, to the 'You' situation, where they instruct the computer to perform this procedure using a language compatible with it. The transfer from 'I' to 'You' has been deemed necessary as it is difficult for students to implement (Zikouli et al., 2003). *Specific expression-based Pseudo-code representation systems*. While students encounter difficulties in formulating a typical sorting algorithm using pseudo-code, the system provides them with the opportunity to correct their attempts by trying to put a number of appropriate specific expressions provided in the correct order. If the result is correct, the system automatically sends a message.

(c) *Animation*. Students are provided with extra RS:

- Animation-based RS.
- Flow-chart based dynamic RS.
- Pseudo-code based dynamic RS.
- Multiple-Linked Animation RS.

Animation-based RS. Animations visualizing the sorting procedure of elements to be sorted, using the sorting algorithms in focus, are also provided. These elements are bars whose length corresponds to the sort key value of the element. By observing the process of sorting, these bars using different sorting algorithms



Statistics
Number of swaps : 4
Number of comparisons : 6
Sorted : No

```
For j= 1 to i
If A[j]>A[j+1] then
temp=A[j]
A[j]=A[j+1]
A[j+1]=A[j]
End if
End for
End for
End
```

Fig. 3. Multiple ad Linked Animation RS, integrated within the SORTING environment.

and students can acquire a primary sense of each of them. Students also can observe the animation for sorting the Euro coins – described in the previous section - using the aforementioned algorithms (see Fig. 2). In this way, students can reflect on these animations and correct their own individual approaches to these algorithms.

Multiple-linked animation RS (MLARS). Interactive animations, demonstrating the visual history of sorting the Euro coins step by step and using each of the algorithms in focus, are also provided (see Fig. 3). Students can also specify the speed of animation. In addition, these animations are interlinked with the *pseudo-code RS* as well as with *flow-chart-based RS*. As the sorting of coins progresses, the corresponding section of the flow-chart and the pseudo-code is highlighted. When a coin swap is performed, two pointers indicate the coins exchanged and a message is also displayed. The number of swaps and comparisons as well as the value of counters regarding each algorithm execution step is also displayed. Thus, students have the opportunity to explore the algorithm execution in all these RS, to correct their own individual approaches to these algorithms and to link different approaches to a sorting algorithm, such as practical sorting of entities, flow-chart and pseudo-code. Consequently, students have the opportunity to make connections between practical knowledge using hands-on experience, visual knowledge presented in the visual flowchart and symbolic knowledge presented through the dynamic pseudo-code. In this way, students are provided with opportunities to acquire a broad view of sorting algorithms.

Technical information: *The environment is implemented using Java applets and Visual C++.*

4. The formative-evaluation pilot study

The formative-evaluation pilot study was realized taking into account a constructivist approach, stressing the investigation of student progress during interaction, within the context of the learning environment provided (Jonassen, Mayers, & McAleese, 1992). In terms of methodology, this is a qualitative research study Q1 (Cohen and Manion, 1989) focusing on: (a) the investigation of students' intuitive sorting approaches, (b) how students construct their knowledge of typical sorting algorithms - within the context of SORTING – and (c) the usability of the tools provided.

The study took place in two typical schools (in Kalamata and Patras, Greece), with the participation of 20 students from the 12th grade (all 18-years-old). Before interacting within the SORTING environment, students were asked to express their previous knowledge of sorting and sorting algorithms, a decision taken in order to investigate the impact of the SORTING environment on student knowledge. All students participating in this experiment had been taught at school about basic algorithmic structures, assignment statements, variables, counters and how to form flow-charts. They had also been taught the "Bubble sort" algorithm and its pseudo-code two months previously. Despite this fact, none of the participants successfully described this algorithm, either verbally or in pseudo-code. All students mentioned that they knew that sorting in general is about putting measurable entities in order. Moreover, all students had some knowledge regarding computers and their operations (5–6 years experience with computers) as the secondary school curricula in Greece consisted of relevant courses (1 course per year/1–2 h per week).

The experiment was performed in three phases. In the first phase, students' intuitive sorting approaches were investigated. In the second and third phase, student approaches to typical sorting algorithms such as "Bubble sort" and "Selection sort" were also explored. In each phase, students were asked to use the appropriate RS provided: (a) actively to sort the 7 coins using the sorting algorithm in focus, and (b) to describe the sorting procedure they performed by using both free text and pseudo-code.

Students worked individually within the context of SORTING. The researchers participated with minimum intervention in order not to affect student strategies. Any interventions were in the form of unstructured interviews (to clarify the meaning of students' actions and to encourage them to continue their learning attempts) and are reported in relation to the specific point in the pilot-study where they were performed. The duration of the experiment was commensurate with student needs; each student took about one hour to complete each phase of the experiment. The data resources were: the researchers' field notes, the log files automatically generated by the system capturing the students' actions and visual snapshots of student constructions taken during the experiment.

5. Results

The results, organized according to the data emerging from each phase of the experiment, are presented below:

5.1. Phase 1: Students' intuitive sorting approaches

5.1.1. Student sorting attempts using SROs

All students used the same method to sort the 7 coins: they used their visual perception to select the coin with the greatest value and dragged it to drop in the right position (see Fig. 1). Next, they focused on the remaining 6 coins and performed the same procedure, dragging the selected coin and dropping it in the last but one position. This procedure continued until all coins were sorted. It is worth mentioning that this intuitive algorithm is similar to the Selection sort algorithm, with the exception that, in this algorithm, the selection of the entity with the smallest value is emphasized.

5.1.2. Student attempts using TRS

Despite students having envisaged describing their intuitive procedures in their natural language as a trivial and easy task, this was clearly not the case when they started writing. Nobody wrote a clear text correctly describing the procedure used on their first attempt. The descriptions were usually vague and incomplete and sometimes concerned other sorting procedures which were not performed during this phase of the experiment, e.g. "Bubble sort". Although all students succeeded in sorting the coins provided one-in-three did not realize the exchange of coins during the dropping phase. After specific interventions by the researcher, asking students to reflect on their text, they returned and actively attempted to sort the coins provided. They tried a number of times (approximately three) to sort the coins until they realized what they were doing and wrote the sorting procedure correctly. In the words of one student, "*I compared the coins in order to find the one with the greatest value and then I exchanged it by drag-and-dropping the coin into the right position; I used this procedure repeatedly until all coins were sorted*".

5.1.3. Student attempts at using PCRS

Some students (2 students) skipped the text description of the intuitive sorting approach they used and tried to write it directly in pseudo-code. They argued that '*pseudo-code is stricter than natural language*' and consequently easier to use. Nonetheless, the majority of students (18 students) were of the opinion that it is hard to write the correspondent pseudo-code correctly, while one of them wrote the pseudo-code of "Bubble sort". To overcome their difficulties, these students went back and actively tried to sort the coins again and again (on average three times). After this experimentation, one-in-five students (4 students) succeeded, while the remaining students took advantage of the opportunity to put the specific expressions provided (*SEPCRS)* in correct order and receive feedback. The majority of these students (13 students) put these expressions in correct sequence at the same time, commenting that, '*these expressions helped me to understand how to write the procedure I performed in pseudo-code'* and '*these expressions helped me to organize my thinking'*. The one remaining student gave up and moved to "Bubble sort", commenting that he preferred to use this because it is a faster algorithm than the algorithm he had been using.

5.2. Phase 2: Student approaches to the 'Bubble sort' algorithm

5.2.1. Student sorting attempts using SROs

Despite the fact that students had been taught about "Bubble sort" at school two months previously, they did not succeed in sorting the coins using hands-on experience. To overcome this difficulty, students: (a) read the associated information in text again and again and (b) observed the automatic execution of sorting the coins using the Bubble algorithm. Each student tried (a) and (b) about 3 times. At this point, some students expressed the view that "*I needed to be informed about Bubble sort so I needed to read the text provided and also observe its automatic execution'*. After these attempts, all the students succeeded in sorting the coins using this algorithm. During this phase, all students also expressed the opinion that this kind of experimentation was very interesting for them and more attractive than traditional school chalk-and- blackboard or paper-and-pencil practices, exclusively focusing on pseudo-code representations.

5.2.2. Student attempts using TRS

Despite the fact that all students managed to sort the coins using the Bubble sort procedure and all of them had read the text describing it only a few minutes previously, a mere 6 of them formulated a correct description of this algorithm using natural language. In our view, this was mainly due to the following factors: students did not devote time to reflection on their sorting actions, they paid little attention to the text they read and their writing skills were limited. The remaining students returned to sort the Euro coins using hands-on experience. After trying on average three times they seemed to understand what they did and correctly managed to write the sorting procedure at hand.

5.2.3. Student attempts using PCRS

All students encountered difficulties in their attempts to describe Bubble sort in pseudo-code. As it emerged from students' logfiles, their most common mistakes were: (a) failure to acknowledge the necessity of nested loops. Instead, students used a simple loop. In fact, these students focused exclusively on the procedure consisting of the first search of coins and their respective exchanges. Consequently, students did not have full control over the iteration of this procedure; (b) double pointers being used inappropriately; students confused the ending values of I and J pointers, (c) failure to terminate the whole procedure. To surmount these difficulties, students returned to sort the 7 coins using hands-on experience. They also observed the animation on sorting the coins, using the multiple and linked RS (MLARS) provided. Each student tried these MLARS about 3 times. In particular, students paid attention to the animation of sorting the coins and at the same time they expressed that they observed the swaps performed, the changes in the values of the counters, as well as the changes in the parts of the flowchart and in pseudo-code highlighted and the unfolding of the nested loops use. After this experimentation, the majority of the students (15 students) succeeded in describing Bubble sort using pseudo-code. The remaining students (5 students) helped to correct their attempts by trying to put the specific pseudo-code expressions provided (*SEPCRS)* in correct sequence and exploiting the feedback given by the system. It is worth noting that each student tried the above activity on average three times before being able to write the Bubble sort pseudo-code correctly.

On the whole, it should be noted that the active sorting of coins using hands-on experience seemed to significantly help the students to successfully manage the process of sorting using the Bubble sort algorithm. In addition, the interactive animations demonstrating the visual history of sorting the Euro coins step by step using the Bubble algorithm in combination with the dynamic flow-chart and the dynamic pseudo-code appeared to play a crucial role in students'successful interpretations of the Bubble algorithm in relation to its pseudo-code. Finally, the possibility of receiving feedback by putting in order the specific expressions provided helped students to correct their sorting approaches using pseudo-code.

5.3. Phase 3: Student approaches to the 'Selection sort' algorithm

5.3.1. Student sorting attempts using SROs

As the sorting procedure that students used during the first phase of this experiment was similar to the Selection sort procedure, they did not encounter difficulties in actively sorting the 7 coins. In particular, each student read the text information provided on average 3 times and also observed the automatic sorting of coins on average 3 times before they could actively manage to sort them.

5.3.2. Student attempts using TRS

When students tried to express their approaches to Selection sort in natural language, they also encountered similar difficulties to those they faced using the previous algorithm. However, all these students realized that Selection sort is similar to their own sorting procedures used in the first phase. Consequently, they exploited their experience with the result that all of them correctly accomplished this task.

5.3.3. Student attempts at using PCRS

To avoid failure in formulating the Selection sort pseudo-code, each student practiced about 4 times with the MLARS provided. During this experimentation, students expressed that they also paid attention to the comparisons of coins performed and the relative changes realized, including swaps of coins, counters, parts of the flowchart and pseudo-code. After this experimentation, all students tried writing in PCRS, with the majority of them succeeding. Only five of these students failed, their most common mistakes being the same as those that came to light in performing Bubble sort. The rest of the students managed this task by experimenting, putting the specific expressions provided (*SEPCRS)* in correct order and receiving feedback.

5.3.4. Usability issues

All students experimented on their own with the operations provided by the SORTING environment and found them familiar and easy to use. The possibility of 'drag and drop' was not obvious as this was not displayed on the interface of the environment. At this point, researchers explained how to use "drag and drop" to put both the coins and the *SEPCRS* provided in correct order.

6. Discussion

In this paper, the design, specific features and the formative-evaluation pilot study of a multi-representational computer environment for the learning of sorting algorithms by secondary level education students has been presented. It should be noted that the area of sorting algorithms is an under-researched one in terms of students' learning and design of appropriate computer environments in secondary level education.

The design of this environment is based on a modeling process consisting of the construction of three models: (a) the learning model, based on modern constructivist and social views of learning, (b) the subject matter model, based on the information provided by the literature on sorting algorithms and (c) the student model, based on a field study of student needs and difficulties encountered while performing sorting procedures in order to grasp concepts related to sorting algorithms. In the construction of the aforementioned models, an attempt has been made to integrate theoretical principles into the design specifications. An attempt has also been made to exploit the advantages of computer technology in terms of the use of computational objects, immediate feedback, interactive animations and multiple and linked representation systems. In the context of SORTING, students can study three sorting algorithms: Bubble sort, Quick-sort and Selection sort. However, the system can be adapted to integrate more sorting algorithms than those previously mentioned.

The SORTING environment was tested in the field with actual students, in order to determine both its teaching ability and its usability. Despite the limitations of this field study, due to the small number of students participating, its findings provide some insights into student learning about sorting algorithms. The data analysis reveals that students used all representation systems provided in order to express their own intuitive sorting methods as well as their own approaches to the typical sorting algorithms in focus. More specifically, the possibility of sorting SRO using hands-on experience mostly helped students to express their own sorting methods and their approaches to typical sorting algorithms in focus in a way that was practical. In addition, students reflected on the hands-on experience they acquired from this kind of activity and interpreted their sorting approaches in both natural language and pseudo-code. Consequently, we can say that students' hands-on experience with SROs played the role of primary reference in students' learning about sorting algorithms. In addition, the text-based information provided, in combination with the step by step visual execution of sorting the SRO provided and the feedback given by the system for each student's sorting actions, played a significant scaffolding role for each student in each practical sorting activity.

Despite the fact that students managed to perform practical sorting using hands-on experience, they faced difficulties in interpreting their practical sorting approaches in the form of free-text. However, the demand to interpret these sorting approaches in free-text played a significant role in students' thinking, since through this activity they were encouraged to reflect on their hands-on experience and to be aware of the sorting procedure at hand. It is worth noting that, at this level of sorting description, students needed some kind of scaffolding through performing practical sorting using hands-on experience.

In the level of expression of sorting algorithms in pseudo-code RS, students presented many difficulties. To overcome these, students used all the RS provided. Students actually succeed in writing the appropriate pseudo-code having experienced practical sorting using hands-on experience in combination with exploration of the co-present representations and the visualization of their explicit relations as well as using the facility for sequencing the specific expressions in pseudo-code and receiving immediate feedback. In particular, the exploration of interactive, interlinked RS seemed to help students to construct bridges between their practical sorting experience and the visual knowledge provided by the dynamic visual flowchart as well as the symbolic knowledge provided in the form of pseudo-code by the dynamic pseudo-code representation system. Moreover, these interlinked systems appeared mostly to help students to manage the specific sorting algorithms in terms of nested loops, counters, assignment statements, initialization and termination of the procedure. The possibility of sequencing the specific expressions in pseudo-code and receiving immediate feedback helped not only the students to correct their attempts at writing pseudo-code but also the students who could not manage to form the appropriate pseudo-code by themselves to accomplish the task at hand.

As regards the aforesaid typical sorting algorithms, translation between practical sorting representations, text and pseudo-code representations was difficult and time consuming for the students because the languages and the operations related to these representations were different. On the whole, in the context of SORTING, students can be encouraged to establish ‘spontaneous’ notions about sorting and then proceed to scaffold a ‘scientific’ synthesis using the variety of features of the environment.

Overall, student learning of typical sorting algorithms seemed to be a difficult task. In terms of factors significant in students’ learning, the findings of our study supported the role of: (a) sorting entities using hands-on experience and receive immediate feedback, (b) exploring analytic animations of the sorting algorithms in focus, automatically performed in a variety of representation systems, (c) expressing their own intuitive sorting approaches as well as their approaches to typical sorting algorithms, (d) experiencing practical sorting in combination with visual animations of sorting entities, dynamic visual flow-charts and the corresponding dynamic visual pseudo-code, and (e) sequencing the specific expressions in pseudo-code and receiving immediate feedback.

Finally, the findings emerging from this study revealed that the operations provided by the SORTING environment are friendly and easy to use.

7. Conclusions

This paper addresses the potential of a modelling, student-centred methodology for the design of a constructivist computer environment – the SORTING environment- for the learning of sorting algorithms, within a context consisting of interactive, multiple and linked representation systems. Concretely speaking, this methodology exploited the results of a field study regarding with students’ learning of sorting algorithms. Students’ experimentation within SORTING provided us with opportunities to clearly address that the learning of typical sorting algorithms by secondary level education students is a complex and difficult task. In fact, during this experimentation, students needed to be involved in practical sorting to express their sorting approaches and also their own approaches to typical sorting algorithms as well as to be engaged in the translation of this practical experience in various representation systems such as: text based RS, flow-chart RS and pseudo-code based RS. To be effectively involved in this translation process students needed to be provided with opportunities to reflect on appropriate feedback. In addition, students needed to make a coherent and integrated framework about sorting, so they needed to be supported to explore how a step by step practical sorting procedure – using a specific algorithm – could be expressed in all the previously mentioned RS. It is worth noting that, SORTING provides students with all the opportunities mentioned above as well as step by step scaffolding-feedback in the form of appropriate messages and also in the form of dynamic interlinked animations of the sorting algorithm at hand.

As a final point, it is worth noting that, despite the limitations of this study, due to the fact that the number of students participating in the pilot-formative evaluation of SORTING was limited, these results can be exploited for the teaching and learning of typical sorting algorithms in secondary level education. Our future plans include the introduction of SORTING into a variety of real classroom settings and using the findings of this study as a basis for the investigation of student learning while interacting in this computer environment.

8. Uncited references

Q3 Thompson (1992).

Acknowledgement

Many thanks to V. Kekatos, M. Misedakis, E. Houstis and G. Sipsa, who contributed to the construction of the early version of the software and to the collection of field data.

References

ACM. (1991). Curricula Recommendations, Volume 1: Computing Curricula. *Report of the ACM/IEEE-CS Joint Curriculum Task Force*. [www document] URL http://www.acm.org/education/curr91/homepage.html.
ACM. (2003). A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee. Retrieved March 12, 2006 from http://www1.acm.org/education/k12/k1final1022.pdf.
Ainsworth, S. E., Wood, D. J., & Bibby, P. A. (1996a). Co-ordinating multiple representations in computer-based learning environments. In P. Brna, A. Paiva, & J. Self (Eds.), *Proceedings of the European conference of artificial intelligence in education* (pp. 336–342). Lisbon: Edicoes Colibri.
Ainsworth, S. E., Wood, D. J., & Bibby, P. A. (1996b). Analysing the costs and benefits of multi-representational learning environments. In H. P. A. Boshuizen, M. van Someren, P. Reimann, & T. de Jong (Eds.), *Learning with multiple representations* (pp. 120–134). Oxford: Elsevier Science.
Ainsworth, S. E. (1999). The functions of multiple representations. *Computers and Education, 33*(2–3), 131–152.
Baecker, R. M. (1981). *Sorting out Sorting*. Film, SF, CA, USA: University of Toronto. M. Kaufmann, Publishers.
Bereiter, C., & Scardamalia, M. (1989). Intentional learning as a goal of instruction. In L. B. Resnick's (Ed.). *Knowing, learning and Instruction* (pp. 361–391). Hillsdale, NJ: Lawrence Erlbaum Associates.
Bishop, A. J. (1988). *Mathematical Enculturation*. Dordrecht: Kluwer Academic Publishers.
Borba, M., & Confrey, G. (1996). A student's construction of transformations of functions in a multi-representational environment. *Educational Studies in Mathematics, 31*, 319–337.
Brown, M. H. (1988). Exploring algorithms using BALSA-II. *IEEE Computer, 21*(5), 14–36.
Davidovitch, L., Parush, A., & Shtub, A. (2006). Simulation-based learning in engineering education: Performance and transfer in learning project management. *Journal of Engineering Education, 95*(4), 289–300.
Dershem, H. L., & Brummund, P. (1998). Tools for Web-based Sorting Animation. *ACM Special Interest Group for Computer Science Education Bulletin Inroads, 30*(1), 222–226.
Du Feau, C. (1999). A Sort for Statistics Lesson. *Teaching Statistics, 21*(1), 8–10.
Duffy, T., & Cunningham, D. (1996). Constructivism: Implications for the design and delivery of instruction. *Handbook of Research for Educational Communications and Technology*, 170–198.
Dyfour-Janvier, B., Bednarz, N., & Belanger, M. (1987). Pedagogical considerations concerning the problem of representation. In C. Janvier (Ed.), *Problems of Representation in Teaching and Learning of Mathematics* (pp. 109–122). London: Lawrence Erlbaum Associates.
Gal-Ezer, J., & Zur, E. (2004). The efficiency of algorithms – misconceptions. *Computers & Education, 42*, 215–226.
Geller, J., & Dios, R. (1998). A low-tech, hands-on approach to teaching sorting algorithms to working students. *Computers & Education, 31*, 89–103.
Hofstetter, F. T. (1997). *Multimedia Literacy* (2nd ed.). Boston: Irwin/McGraw-Hill.
Jonassen, D. H. (1991). Objectivism versus constructivism: Do we need a new philosophical paradigm?. *Educational Technology Research and Development 39*(3), 5–14.
Jonassen, D. H., Mayers, T., & McAleese, R. (1992). A Manifesto for a constructivist approach to technology in higher education. In T. Duffy, D. Jonassen, & J. Lowyck (Eds.), *Designing Constructivist Learning Environments*. Heidelberg, FRG: Springer-Verlag.
Jonassen, D. H., & Reeves, T. C. (1996). Learning with technology: Using computers as Cognitive Tools. In D. H. Jonassen (Ed.), *Handbook of Research on Educational Communications and Technology*. New York: Sholastic Press with the Association for Educational Communications and Technology.
Jonassen, D. H. (1999). Designing constructivist learning environments. *Instructional design theories and models, 2*, 215–239.
Jonassen, D. H. (2000a). *Mindtools for Schools: Engaging Critical Thinking with Technology* (2nd ed.). Colombus, OH: Merrill/Prentice-Hall.
Jonassen, D. H. (2000b). Revisiting activity theory as a framework for designing student-centered learning environments. In D. H. Jonassen & S. M. Land (Eds.), *Theoretical Foundations of Learning Environments* (pp. 89–121). London: Lawrence Erlbaum Associates.
Hannafin, M. J., & Land, S. (1997). The foundations and assumptions of technology-enhanced, student-centered learning environments. *Instructional Science, 25*, 167–202.
Hein, G. E. (1999). Is meaning making constructivism? Is constructivism meaning making? *The Exhibitionist, 18*(2), 15–18.
Kann, C., Linderman, R. W., & Heller, R. (1997). Integrating algorithm animation into a learning environment. *Computers & Education, 4*, 223–228.

Kaput, J. J. (1989). Linking representations in the symbol systems of algebra. In S. Wagner & C. Kieran (Eds.), *Research issues in the learning and teaching of algebra* (pp. 167–194). New Jersey: Erlbaum, Hillsdale.

Kaput, J. J. (1992). Technology and Mathematics Education. In D. A. Grouws (Ed.), *Handbook of Research on Mathematics Teaching and Learning* (pp. 515–556). New York: Macmillan.

Kaput, J. J. (1994). The Representational Roles of Technology in Connecting Mathematics with Authentic Experience. In R. Biehler, R. W. Scholz, R. Strasser, & B. Winkelman (Eds.), *Didactics of Mathematics as a Scientific Discipline: The state of the art* (pp. 379–397). Dordrecht: Kluwer Academic Publishers.

Khalife, J. T. (2006). Threshold for the Introduction of Programming: Providing Learners with a Simple Computer Model. 18th Workshop of the Psychology of Programming Interest Group, University of Sussex, September 2006. Retrieved on 15/03/07 from 18th Workshop of the Psychology of Programming Interest Group, University of Sussex, September 2006.

Knuth, D. E. (1973). *The Art of Computer Programming. Sorting and Search* (Vol. 3). Mass: Addison-Wesley.

Kordaki, M., & Potari, D. (1998). A learning environment for the conservation of area and its measurement: A computer microworld. *Computers and Education, 31*, 405–422.

Kordaki, M. (2005a). The role of multiple representation systems in the enhancement of the student model. *Proceedings of 3rd International Conference on Multimedia and Information and Communication Technologies in Education*. Caceres, Spain, pp. 253–258.

Kordaki, M. (2005b). A special purpose e-learning environment: Background, design and evaluation. In Zongmin Ma (Ed.). *Web-Based Intelligent e-Learning Systems: Technologies and Applications* (pp. 348–375). Idea Group Publishing.

Kordaki, M. Miatidis, M., & Kapsampelis, G. (2005). Multi representation systems in the design of a environment for the learning of sorting algorithms. In *Proceedings of IADIS international conference cognition and exploratory learning in digital age (CELDA 2005)* (pp. 363–366). Porto, Portugal, 14–16 December 2005.

Land, S. M. & Hannafin. Student-Centered Learning Environments. In D. H. Jonassen, & S. M. Land (Eds.), *Theoretical foundations of learning environments* (pp. 1–23). New Jersey: Lawrence Erlbaum Associates.

Laurillard, D. (1993). *Rethinking University Teaching, a Framework for the Effective Use of Educational Technology*. London: Routledge.

Linderstrom, E. E., & Vitter, J. S. (1985). The design and analysis of bucket sort for bubble memory secondary storage. *IEEE Transactions on Computers, C-34*, 218–233.

Matsaggouras, E. (1997). *Teaching Strategies*. Athens: Gutenberg.

Mellar, H., & Bliss, J. (1994). Modelling and Education. In H. Mellar, J. Bliss, R. Booham, J. Ogborn, & C. Tompsett (Eds.), *Learning with Artificial Words: Computer Based Modelling in the Curriculum*. London: The Falmer Press.

Naps, T. L. (1990). Algorithm visualization in Computer Science Laboratories. Proceedings of 21st SIGSE Technical Symposium on Computer Science Education. *Special Interest Group for Computer Science Education Bulletin Inroads, 22*(1), 105–110.

Nardi, B. A. (1996). Studying context: A comparison of activity theory, situated action models, and distributed cognition. In B. A. Nardi (Ed.), *Context and consciousness: Activity theory and human-computer interaction*. Cambridge, MA: MIT Press.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning Cultures and Computers*. Dordrecht: Kluwer Academic Publishers.

Piaget, J. (1976). *The grasp of consciousness*. Cambridge, MA: Harvard University Press.

Salomon, G., Globerson, T., & Guterman, E. (1989). The computer as a zone of proximal development: Internalizing reading-related metacognitions from a reading partner. *Journal of Educational Psychology, 81*(4), 620–627.

Stasko, J. T. (1990). Tango: A framework and system for algorithm animation. *IEEE Computer, 23*(9), 39–44.

Schwartz, D. L. (1995). The emergence of abstract representations in dyad problem-solving. *Journal of the Learning Sciences, 4*(3), 321–354.

~~Thompson, P. W. (1992). Notations, conventions and constraints: Contributions to effective users of concrete materials in elementary mathematics. *Journal of Research in Mathematics Education, 23*(2), 123–147.~~

Vlachogiannis, G., Kekatos, V., Miatides, M., Kordaki, M., & Houstis, E. (2001). A multi-representational environment for the learning of Bubble sort. In *Proceedings of Panhellenic Conference with International Participation 'New Technologies in Education and in Distance Learning'* (pp. 481–495). Greece: Rethymnon.

Vygotsky, L. (1978). *Mind in Society*. Cambridge: Harvard University Press.

www.ncrel.org/sdrs/areas/issues/content/cntareas/science/sc500.htm. Critical Issue: Providing Hands - On, Minds-On and Authentic Learning Experiences in Science.

Wertch, J. V. (1995). Introduction. In J. V. Wertsch (Ed.). *Culture, Communication, and Cognition: Vygotskian Perspectives* (pp. 1–20). Cambridge: Cambridge University Press.

Wilson, B. (1997). Thought on theory in educational technology. In B. Seels (Ed.), *Educational Technology*, Special Issue on Theory, Jan/Feb. 22–27.

Zikouli, K., Kordaki, M. & Houstis, E. (2003). A Multi-representational Environment for the Learning of Programming and C. *Proceedings of 3rd international conference on advanced learning technologies 2003*. Athens, Greece, p. 459.