

Towards the use of metaphors in the design and implementation of educational software

Maria Kordaki^{*}, George Strimpakos^{*}, Zisis Charalampidis^{*} and Thanasis Daradoumis^{**}

^{*} Dept of Computer Engineering and Informatics
Patras University, 26500, Rion Patras, Greece

^{**} Department of Computer Science, Open University of Catalonia
Rambla Poblenou 156, 08018 Barcelona, Spain
e-mail: kordaki@cti.gr, adaradoumis@uoc.edu

Abstract: This paper presents a case study that illustrates the significance of using metaphors in the design of educational software. We have chosen to apply our approach to the specific domain of Operating Systems (OS), showing how beginners can learn OS basic concepts more efficiently. Our software helps students study scheduling, using a simple and fundamental algorithm – the Priority Round-Robin (PRR) algorithm, used for scheduling in most OS, in a metaphorical context consisting of familiar and not-too-complex set of tasks. To this end, the rationale and the features of the proposed software are clearly presented; showing that, besides the use of metaphors, the design of the educational software was also based on modern social and constructivist learning theories. The architecture of the proposed software consists of two main parts: a) an information space where students are provided with opportunities to access appropriate information about the specific subject-matter they have to learn and b) an experimentation space that allows students to take control of their learning by forming their own set of tasks, assigning them priorities and durations. This architecture could be used for the learning of diverse scheduling algorithms. Finally, the system provides learners with feedback demonstrating the correct scheduling of the proposed tasks so that they can make self corrections.

Introduction

Computers have already had an enormous impact on the way we live, think and act. Indeed, it would not be an exaggeration to say that our lives depend upon computer systems. Yet, many believe that the true computer revolution will not happen until everyone can understand the technology well enough to use it in truly innovative ways (Association for Computing Machinery; ACM, 2003). Thus, understanding and advocating of Computer Science (CS) as an essential component of a well-rounded education is a key factor in ensuring that young students have the skills needed, not just to survive, but to thrive in this increasingly technological and global society. To this end, a four-level model curriculum of CS for K-12 has been proposed (ACM, 2003). One of the four goals of this curriculum is to introduce the fundamental concepts of CS to all students, beginning at elementary level. Indeed, Level I of the previously mentioned curriculum (recommended for grades K-8) is geared to provide primary and secondary school students with fundamental CS concepts by integrating basic technology skills with simple algorithmic thinking. Among these fundamental concepts, understanding how a computer works is pre-requisite. To this end, the introduction of young students to the concept of an OS is critical for the achievement of the goals of the CS curriculum at these education levels. It is also worth noting that both students in secondary education and adults have difficulties in understanding how a computer works (Soloway & Spohrer, 1989).

The concept of an OS in CS is complex as it is related to algorithms and operations which are realized in a hidden, not visible way. Consequently, they are difficult for young students to grasp. Several investigators have proposed the use of metaphors to help learners grasp concepts within complex domains of knowledge (Lakoff & Johnson, 1980; Ortony, 1979). A metaphor is a kind of linguistic figure with the form “X is Y” consisting of two major parts (Carroll and Mack, 1999). These major parts include novel concepts (X = target domain), previously acquired knowledge (Y = base domain), and the relationship between the two (Wozny, 1989). The significant role of metaphors is based on the fact that - when they are used - the critical role of students’ prior knowledge in acquiring new knowledge is acknowledged as well as the essential role of engaging learners in real life problems (Lakoff & Johnson, 1980; Ortony, 1979). However, the essential role of a metaphor in the learning process is much more than a structural description of two domains and the mapping relationships of one to another (Carroll and Mack, 1999). In fact, in the context of a metaphor, learners can be motivated

to be actively engaged in their learning. In addition, a close relationship seems to exist between the provision of metaphors and the formation of more comprehensive mental models, which results in better performance by learners in carrying out complex tasks (Hsu, 2006).

Various studies have reported the significance of the use of metaphors in the learning of concepts within the complex domain of CS (Carrol & Tomas, 1982; Carrol and Mack, 1999; Duncker, 2002; Hundhausen, Douglas and Stasko, 2002; Hsu, 2006; Sajaniemi, Byckling, and Gerdt, 2007). In particular, metaphoric comparisons can structure the activity of learning to use a computer system, for example, the metaphor "a text editor is a typewriter" can be used in beginners' learning of text processors (Carrol & Tomas, 1982). Metaphors have also been used in the understanding of abstract concepts of CS, such as databases, where for example a desk has been used as a metaphor. Nowadays, metaphors are also an important part of a user interface design (Mayhew, 1992; Duncker, 2002). In fact, many technology-based learning products or environments as well as many WWW-pages are built upon a metaphor. Such metaphors are for example a book, a map, a village, a house or a classroom. These metaphors have been implemented through screen images/ backgrounds, icons/ buttons and interactive drag and drop functions (Cates, 2001). At this point, it is worth noting that system interface designs succeed or fail with respect to their learnability depending on what metaphors they suggest and how helpful these are to learners (Carrol and Thomas, 1982).

Appropriately designed computer-based learning environments can play a crucial role in student learning (Noss and Hoyles, 1996; Jonassen, 1999). In fact, computers are an ideal medium for providing unique opportunities for the design of learning situations within the framework of modern social and constructivist theories of learning (Vygotsky, 1974; Jonassen, 1991). In addition, computers can provide wide opportunities for the construction of various different, linked and dynamic representation systems, such as texts, images, equations, variables, tables, graphs, animations, simulations of a variety of situations, programming languages and computational objects (Kaput, 1994). These systems can be effectively used for the implementation of appropriately designed metaphors to be exploited for the learning of specific complex concepts. To this end, students could become actively engaged in their learning and receive appropriate feedback on their actions for them to make self corrections. At this point, it is worth noting that the role of designing appropriate feedback is crucial for student learning in the context of computer learning environments (Hummel, 2006) and especially those dedicated for the learning of such a complex topic as Operating Systems, indeed in any educational setting (Cohen, 1985).

The importance of using educational software in teaching Operating Systems courses is clearly depicted in surveys conducted by (Davoli & Goldweber, 2003; Anderson & Nguyen, 2005). Several attempts have been made to build general-purpose computer system simulators that allow undergraduate students to undergo an innovative and pedagogically different experience of learning OS (Morsiani & Davoli, 1999; Goldweber, et. al, 2005; Holland, et. al., 2002; Hovemeyer, et. al., 2004). More recent work has contributed in creating instructional simulators with a high degree of performance realism and simplicity (DeRosa, et. al., 2006; Dobrilovic & Stojanov, 2006; Liu, et. al., 2007; Laverell, et. al., 2008). Although various items of educational software for the learning of CS concepts have already been reported, educational software for the learning of basic aspects of Operating Systems by secondary level education students through the use of metaphors has not yet been reported. This is the main contribution of this work. In the design of this software, social and constructivist theories of learning were taken into consideration (Vygotsky, 1974; Jonassen, 1991; 1999) while at the same time the role of metaphors in student learning was acknowledged. In the following sections of this paper, the rationale for the proposed educational software is presented and subsequently its features are described using specific examples and then discussed in relation to their possible use with real students. Finally, conclusions and proposals for future plans are drawn.

The rationale behind the design of the proposed educational software

In the design of the proposed educational software, socio-cultural and constructivist perspectives on knowledge construction (Jonassen, 1991; 1999; Hannafin & Land, 1997; Vygotsky, 1974) were taken into account. According to these perspectives, learning is considered as an active, subjective and constructive activity placed within a rich and meaningful context for learners. The role of tools as mediators for the development of students' higher mental functions has been also acknowledged (Vygotsky, cited in Wertch, 1995). To this end, digital media 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 and Scardamalia, 1989; Noss & Hoyles, 1996).

In the context of constructivist learning, the role of exploiting a student's prior knowledge is very important in facilitating his or her learning of new concepts. In fact, during the learning process, new cognitive structures are developed from the existing structures. To this end, the role of metaphors as mediators of students' previous knowledge in the construction of a knowledge representation for some knowledge domain has been acknowledged (Carrol and Mack, 1999). However, the outcome of the comprehension process is not an understanding of some metaphorical description of the target domain per se but rather a representation of the domain itself - that is to say, a mental model (Collins and Gentner, 1987). Metaphors can have the form of kernel comparison statements whose primary function in learning is to stimulate active learner-initiated thought processes (Carrol and Mack, 1999). Metaphors are also open-ended because open-ended comparisons stimulate these processes more than explicit and comprehensive comparisons do.

Three kinds of approaches to metaphors have been used in CS education (Jiménez-Peris, et al., 1997): a) Operational metaphors focusing on their measurable effect on learning: for example, the use of 'advance organizer' as a metaphor for learning to use a text editor (Foss, Rosson and Smith, 1982). Operational approaches, b) Metaphors with structural approaches develop formal presentation of relations between the source and the target domain (Ortony, 1979; Gentner, 1983). A typical definition according to this structural approach to a metaphor would be: Given two domains A and B; taking A as a metaphor for B is equivalent to providing a formal mapping from the primitives defining A into the primitives defining B, at the same time pairing the nodes of each (it is implied that these domains are expressed in terms of graphs), and c) Pragmatic approaches acknowledging that metaphors are incomplete, but claiming to be powerful not only because of the similarities between the target and the source domain but also because of their differences (Carrol and Mack, 1999). In fact, the differences between the metaphor and what it describes can turn out to be as much of a problem as a benefit. Dubinsky & Hazzan (2003) suggest that the use of metaphors can create discussion among students about the similarities and the differences between the target and the source domain. Thus, the interaction with and comparison of differences and similarities of these domains can be very helpful for students in clarifying their primitives as well as developing their maturity so as to enable them to perform such comparisons (Cates, 2002). In this paper, the essential role of these pragmatic approaches has been taken into account.

Various studies have proved the positive effects of metaphors on learning subject domains or computer systems (Mayer, 1976; Gentner and Gentner, 1983; Streitz et al., 1989; Borgman, 1999; Cameron, 2002). Some of these studies illustrate the changes of the subjects' mental models, changes brought on by the intervention of metaphors. The studies show that metaphors may have positive effects on subjects' performance and construction of mental models in more complex learning situations. At this point, it is worth noting that various attempts have been made to identify the properties of a good metaphor (Gentner, 1983). However, metaphoric comparisons seem to be at least open-ended; in terms of this, the mappings implied in the structures of the target and base domains are by definition incomplete (Carrol and Mack, 1999) because there is an attempt to realize descriptions of real world phenomena. To this end, the success of a metaphor depends on having a familiar domain to analogize from and on recognizing enough in the new domain for some correspondence to be established (Carrol and Mack, 1999). In fact, a metaphor 'X is Y' (where Y has some known properties for the students) can facilitate active learning by providing them with opportunities to generate and test various hypotheses about the target novel domain (X) based on the similarities and dissimilarities it has with the base domain (Y). As Carrol and Mack (1999; p. 393) have pointed out, 'salient dissimilarities - in the context of salient similarities - stimulate thought and enhance the efficacy of the metaphor as a learning vehicle. To this end, metaphors can be used as orienting frameworks, at the same time leaving many operational details for the learner to discover. Thus, a metaphor can orient the learner to form and verify hypotheses about similarities between the target and the base domain and also to recognize and analyze their differences.

Metaphors have to be carefully crafted and presented in such a way as to help learners to form - in a comprehensible way - the appropriate connections between the metaphor and critical points of the new knowledge. These points cannot be readily understood in the student's own terms. Here, it is worth noting that the metaphor used cannot be too complex when a totally new and possibly difficult concept

is being taught. Probably, the use of more than one metaphor is appropriate for the learning of a novel concept. In addition, when students are encouraged to be involved in a process of generating their own metaphors regarding a novel learning subject, it can become a principal learning process (Carrol and Mack, 1999).

Taking into account all the above, we have constructed an interactive metaphor-based computer environment for beginners' learning of basic concepts related to an OS. A preliminary version of the design of this software has been presented in Kordaki, Strimpakos, Charampidis, and Daradoumis (2008). Specifically, the emphasis is put on the understanding by students of the most important missions of an OS, which are the management of all the resources that constitute the configuration of a modern computer system and which coordinate the access any user activity will have on them (Tanenbaum, 2001; http://en.wikipedia.org/wiki/Operating_system). In fact, a modern computer system consists of one or more processors, main memory, disc drives, printers, keyboard, monitor, network connections and many other I/O devices. An OS is a software layer that acts as a host for application programs that run on the machine. As a host, one of its main purposes is to handle the details of the operation of the hardware. This relieves application programs from having to manage these details and makes it easier to write applications. Almost all computers, including hand-held computers, desktop computers, supercomputers, and even modern video game consoles, use an operating system of some type.

Scheduling is a key concept in computer multitasking and multiprocessing operating system design, as well as in real-time OS design (Tanenbaum, 2001; [http://en.wikipedia.org/wiki/Scheduling_\(computing\)](http://en.wikipedia.org/wiki/Scheduling_(computing))). It refers to the way processes are assigned priorities in a priority queue. This assignment is carried out by software known as a scheduler. Among the plethora of common scheduling practices and disciplines, the '*Priority Round-Robin Scheduling*' algorithm is one of the oldest, simplest, fairest and most widely used scheduling algorithms designed especially for time-sharing systems (Faisstnauer, Schmalstieg, Purgathofer, 2000). A small unit of time, called a time slice or quantum, is defined. All runnable processes are kept in a circular queue. The CPU scheduler goes around this queue, allocating the CPU to each process for a time interval of one quantum. New processes are added to the tail of the queue. The CPU scheduler picks the first process from the queue, sets a timer to interrupt after one quantum and dispatches the process. If the process is still running at the end of the quantum, the CPU is pre-empted and the process is added to the tail of the queue. If the process finishes before the end of the quantum, the process itself releases the CPU voluntarily. In either case, the CPU scheduler assigns the CPU to the next process in the ready queue. Each process has a priority and all processes with the same priority go into a separate queue. The Round Robin algorithm is applied to the queue where processes have the highest priority. As soon as all these processes have been successfully executed, the same algorithm is applied to the queue where processes have the second-highest priority.

As mentioned above, one purpose of an OS is to handle the details of the hardware actions without letting the high level processes or users see those details. This feature was a great boost in Application Programming and development but at the same time left amateur users completely ignorant of how Operating Systems work and led to many misconceptions! What we tried to create is an emulation of *Priority Round-Robin* (PRR) algorithm (Faisstnauer, Schmalstieg, Purgathofer, 2000) in order for students to understand how scheduling works. The PRR algorithm was chosen because of its simplicity but it was preferred to the classic RR algorithm because the latter was too simple. However, it is stated clearly that, if a student sets the same value for every priority, he will eventually have the same results as he would using the classic RR algorithm. The architecture and the features of the proposed educational software for the understanding of task management and scheduling using PRR are described in the following section.

Description of the proposed educational software

The proposed software is organized in HTML pages, with its features being presented on its main page, and classified into two parts: i) information, and ii) experimentation space. A diagrammatic presentation of the general structure of this software is reflected in the menu bar on its 'Homepage', (see Figure 1a).

Information

Three type of information are provided: a) general information - in the form of text - about basic aspects related to OS, such as task management, scheduling and the notion of time quantum as well as a simple description of the PRR algorithm, b) appropriate URLs for further study, and c) metaphoric ready examples of task scheduling using a specific time quantum and the PRR algorithm.




Figure 1a. A metaphor based example using PRR




Figure 1b. Selecting a set of tasks as well as their priorities and duration

A metaphoric example is presented in Figure 1a, where a set of tasks which have to be performed by a secretary are automatically managed by the system using the PRR algorithm and taking a time quantum of 10 minutes into account. This set consists of the following tasks: Bring manager’s coffee, photocopy documents, check mail, break and meeting. The correspondent priorities of these tasks are: 2nd, 3rd, 3rd, 4th, 1st. The assigned duration for each task is quantum 1, 2, 4, 1 and 4 correspondingly. The end of each task is also demonstrated. The metaphor examples used were selected in order to be simple and familiar to the students. To this end, the set of tasks selected to be scheduled stem from students’ everyday life; thus, they have the opportunity to easily grasp the concept of task scheduling using priorities. In addition, students have the opportunity to give meaning to the concept of time quantum, as it is far greater than those actually used by a real OS. By exploring these metaphor examples, students also have the chance to estimate similarities and dissimilarities between these metaphoric examples and real OS. In this way, students can acquire a deeper understanding of the basic concepts of OS and of PRR algorithm.

Experimentation Space

Within the experimentation space, students can learn about the previously mentioned concepts by taking an active role designing their own metaphor examples.

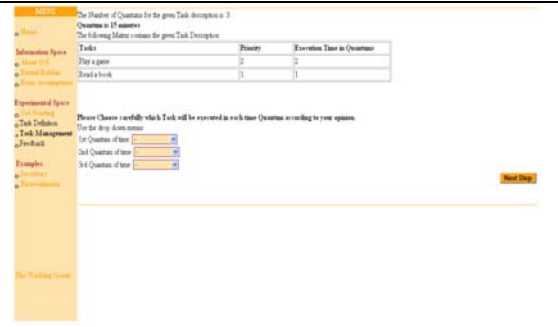


Figure 2a. Student scheduling of tasks using PRR

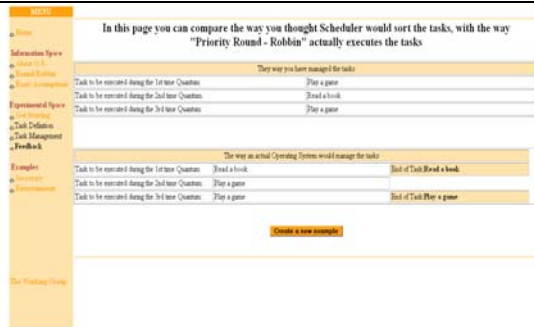


Figure 2b. System scheduling of the tasks using PRR

In fact, students are provided with the following possibilities: a) to define a time quantum that they can select from a list of such. The available options are: 10, 15 and 30 minutes and b) to define their own scenario of tasks (maximum 10 tasks) and to assign them with the priorities they prefer. Students also have the chance to define the duration of each task in terms of the time quantum they selected in the previously mentioned step (see Figure 1b), c) to schedule the set of tasks according to their own conception of PRR algorithm (see Figure 2a), and d) to receive feedback from the system in terms of

automatically scheduling the said set of tasks using the PRR algorithm (see Figure 2b). Appropriate help is also provided for the students to easily perform all the related operations provided by the software.

By forming their own metaphors, students can be active and also take control of their learning. Students can also focus on the previously mentioned critical concepts related to OS and the PRR algorithm. Students can also generate more than one metaphorical context of tasks so that they can clarify these critical concepts. Then, students can further discuss and hypothesize about similarities and dissimilarities of the task scheduling in this metaphorical open-ended context and the context of the real OS and the PRR algorithm. In this sense, the metaphorical contexts formed can act as orienting frameworks leaving many other details for the students to explore with their teachers in the discussion sections. Moreover, by forming their own familiar domain to analogize from and on recognizing enough in the new domain of OS using PRR, students are expected to be able to establish some correspondences. It is worth noting that the metaphorical context - that supported by the proposed software to be formed by the students - is approached from a pragmatic point of view (Carol and Mack, 1999). Consequently, the metaphors used/formed are incomplete and open-ended but can become powerful not only because of the similarities between the two domains but also because of their differences.

Technical characteristics: This Project was created using PHP, HTML and JavaScript; an Apache server and PHP v5.0 are required.

Epilogue and future plans

The Interactive metaphor-based educational software for the learning of basic concepts of an Operating System by beginners has been presented in this paper. In the context of this software, students have the chance to learn about some critical concepts related to OS, such as task scheduling, task priorities and time quantum, using a simulation of the *Priority Round-Robin algorithm*. In fact, students have the chance to receive not only appropriate information about basic aspects of OS and the PRR algorithm and to explore the previously mentioned concepts using familiar pragmatic metaphor-based ready examples but also to set their own task scenarios and to assign them the priorities they prefer as well as to define their duration using a time quantum they can select from a list. Students can schedule the tasks they have defined using their own conceptions about the PRR algorithm and also receive feedback from the system, providing the correct scheduling of the said tasks using this algorithm. In this way, students can verify their attempts and also evaluate their mistakes and so proceed through questioning and discussions to clarify the related topics of difficulty. In fact, the system is structured in such a way as to provide students with the opportunity to exploit their previous knowledge of familiar tasks and their scheduling in order for them to make connections with basic aspects of real OS based on analogical reasoning. In the context of this proposed educational software, students can experience enjoyment and pleasure by forming their own task scenarios and assigning them both priorities and durations. It is also expected that students will try interacting with the proposed software a number of times to reflect and make clarifications and abstractions about basic aspects of OS and the PRR algorithm. To verify our expectations, field studies to assess the proposed educational software using real students are necessary. Finally, the enrichment of the software through simulation of a greater number of scheduling algorithms is among our future plans.

References

- Association for Computing Machinery, (2003). *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee*. Last retrieved on 20/05/08 from http://www.acm.org/education/curric_vols/k12final1022.pdf?searchterm=K-12
- Bereiter, C. and 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, USA: Lawrence Erlbaum Associates.
- Borgman, C.L. (1999). The user's mental model of an information retrieval system: an experiment on a prototype online catalog. *In International Journal of Human-Computer Studies*, 51, pp. 435-452.

- Cameron, L. (2002). Metaphors in the learning of science: A discourse focus. *In British Educational Research Journal*, 28 (5), pp. 673–688.
- Carroll, J.M., Thomas, J.C. (1982). Metaphor and the cognitive representation of computing systems. *In IEEE Transactions on Systems, Man, and Cybernetics*, SMC 12 (2), pp. 107–116.
- Carroll, J.M. and Mack, R.L. (1999). Metaphor, computing systems, and active learning. *In International Journal of Human-Computer Studies*, 51, pp. 385–403.
- Cates, W.M. (2002). Systematic selection and implementation of graphical user interface metaphors. *In Computers & Education*, (38), pp. 385-397.
- Anderson, C. & Nguyen, M. (2005). A survey of contemporary instructional operating systems for use in undergraduate courses, *Journal of Computing Sciences in Colleges*, v.21 n.1, p.183-190.
- Cohen, V.B. (1985). A reexamination of feedback in computer based instruction: Implications for instructional design. *In Educational Technology*, pp. 33-57.
- Collins, A., Gentner, D. (1987). How people construct mental models. In: Holland, D., Quinn, N. (Eds.), *Cultural Models in Language and Thought*, pp. 243–268, New York, USA: Cambridge University Press.
- Davoli, R. & Goldweber, M. (2003). New Directions in Operating Systems Courses Using Hardware Simulators, *ICSEE'03*, SCS 2003.
- DeRosa, P., Shen, K., Stewart, C., and Pearson, P. (2006). Realism and simplicity: disk simulation for instructional OS performance evaluation, *ACM SIGCSE Bulletin*, v.38 n.1, March 2006.
- Dobrilovic, D & Stojanov, Z. (2006). Using Virtualization Software in Operating Systems Course. In *Proceedings of the International Conference on Information Technology: Research and Education*, 16-19 Oct. 2006, pp. 222 – 226.
- Dubinsky, Y. & Hazzan, O. (2003). Using Metaphors in eXtreme Programming Projects. In M. Marchesi & G. Succi (eds.) *XP 2003*, pp. 420-421, Berlin Heidelberg: Springer-Verlag.
- Duncker, E. (2002). Cross-Cultural Usability of the Library Metaphor. *Proceedings of JCDL 2002*, pp. 223-230. Accessed from: <http://csdl.computer.org/comp/proceedings/jcdl/2003/1939/00/19390415.pdf>
- Faisstnauer C. , Schmalstieg D. , Purgathofer W. (2000). "Priority round-robin scheduling for very large virtual environments". *Proceedings of the IEEE Virtual Reality 2000 Conference*, Washington, DC, USA, pp. 135.
- Foss, D.J., Rosson, M.B. and Smith, P.L. (1982). Reducing Manual Labor: An Experimental Analysis of Learning Aids for a Text Editor. In: Nichols, J.A. and Schneider, M.L. (eds.) *Proceedings of the SIGCHI conference on Human factors in computing systems*, March 15-17, 1982, Gaithersburg, Maryland, USA, pp. 332-336.
- Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *In Cognitive Science*. 7 (2), pp.155-170.
- Gentner, D., Gentner, D.R. (1983). Flowing waters or teeming crowds: mental models of electricity. In: Gentner, D., Stevens, A.L. (Eds.), *Mental Models*, pp. 99–130NJ, USA: Lawrence Erlbaum, Hillsdale, .
- Goldweber, M., Davoli, R., and Morsiani, M. (2005). The Kaya OS project and the μ MPS hardware emulator. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, Caparica, Portugal, pp. 49-53.
- Jiménez-Peris, J., Patiño-Martínez, M., Velázquez-Iturbide, J. & Pareja-Flores, C. (1997). The Locker Metaphor to Teach Dynamic Memory. *Proceedings of the twenty-eight SIGCSE technical symposium on Computer Science education*, pp. 169-173.
- Jonassen, D. H. (1999). Designing constructivist learning environments. *In Instructional design theories and models*, 2, pp. 215-239.
- Jonassen, D. H. (1991). Objectivism versus constructivism: Do we need a new philosophical paradigm?. *In Educational Technology Research and Development*, 39(3), pp. 5-14.
- Hannafin, M.J & Land, S. (1997). The foundations and assumptions of technology-enhanced, student-centered learning environments. *In Instructional Science*, 25, pp. 167-202.

- Holland, D., Lim, A., and Seltzer M. (2002). A new instructional operating system. In Proceedings of the *ACM 33rd Technical Symposium on Computer Science Education SIGCSE 2002*, pp. 111-115.
- Hovemeyer, D., Hollingsworth, J. and Bobby Bhattacharjee, B. (2004). Running on the bare metal with GeekOS, *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, March 03-07, 2004, Norfolk, Virginia, USA.
- Hsu, Y.-C. (2006). The effects of metaphors on novice and expert learners' performance and mental-model development, *In Interacting with Computers*, 18 (2006), pp. 770–792.
- Hundhausen, C. D., S. A. Douglas and J. T. Stasko (2002). A meta-study of algorithm visualization effectiveness. *In Journal of Visual Languages and Computing* 13, (2002), pp. 259–290.
- Hummel, H., G-K. (2006). Feedback Model to Support Designers of Blended-Learning Courses. *In International Review of Research in Open and Distance Learning*, 7(3), pp. 1-16.
- Kaput, J. J. (1994). The Representational Roles of Technology in Connecting Mathematics with Authentic Experience. In R. Biehler, R. W. Scholz, R. Strasser, B., Winkelmann (Eds), *Didactics of Mathematics as a Scientific Discipline: The state of the art* (pp. 379- 397) , Dordrecht: Kluwer Academic Publishers.
- Kordaki, M., Strimpakos, G., Charampidis, Z and Daradoumis, T. (2008). Metaphor based educational software for beginners' learning of operating systems. Proceedings of IADIS International Conference E-Learn 2008, October, 2008, Freiburg, Germany.
- Lakoff, and Johnson, M. 1980. *Metaphors we live by*, Chicago, USA: Chicago University Press.
- Laverell, W., Fei, Z., and Griffioen , J., 2008. Isn't it time you had an emulab? In Proceedings of the *39th SIGCSE technical symposium on Computer Science Education*, pp. 246-250, Portland, OR, USA.
- Liu, H., Chen, X. and Gong, Y. (2007). BabyOS: a fresh start, *ACM SIGCSE Bulletin*, v.39 n.1, March 2007.
- Mayer, R.E. (1976). Some conditions of meaningful learning for computer programming: advanced organizers and subject control of frame order. *Journal of Educational Psychology*, 68, pp. 143–150.
- Mayhew, D. (1992). *Principles and Guidelines in Software User Interface Design*, NJ, USA: Prentice-Hall, Englewood Cliffs.
- Morsiani, M. & Davoli R. (1999). Learning operating systems structure and implementation through the MPS computer system simulator. *ACM SIGCSE Bulletin archive*, Vol. 31, Issue 1, pp. 63 – 67.
- Noss, R. & Hoyles, C. (1996). *Windows on mathematical meanings: Learning Cultures and Computers*, Dordrecht: Kluwer Academic Publishers.
- Ortony, A. (1979). *Metaphor and Thought*, Cambridge: Cambridge University Press.
- Sajaniemi, J., Byckling, P. and Gerdt, P. (2007). Animation Metaphors for Object-Oriented Concepts. *In Electronic Notes in Theoretical Computer Science*, 178, pp. 15–22, doi:10.1016/j.entcs.2007.01.037
- Soloway, E. & Spohrer, J. C. (1989). *Studying the novice programmer*, N.J. USA: Erlbaum, Hillsdale.
- Streitz, N., Lieser, A., Wolters, A., 1989. The combined effects of metaphor worlds and dialogue modes in human-computer-interaction. In: Klix, F., Streitz, N., Waern, Y., Wandke, H. (Eds.), *Man-Computer Interaction Research*, pp. 75–88, Amsterdam, NL: MACINTER-II. Elsevier .
- Tanenbaum, A.S. (2001). *Modern Operating Systems (2nd Edition)*, NJ, USA: Prentice Hall.
- Vygotsky, L. (1974). *Mind in society*. Cambridge, MA: Harvard University Press.
- Wertch, J. V. (1995). Introduction. In J. V. Wertsch (Eds), *Culture, communication, and cognition : Vygotskian perspectives*,(pp. 1-20). Cambridge, UK: Cambridge University Press.
- Wozny, L. A. (1989). The application of metaphor, analogy and conceptual models in computer systems. *Interacting with Computers*, 1 (3), pp. 273–283.