Design and Development of a Web-based Interactive Software Tool for Teaching Operating Systems

Article *in* Journal of Information Technology Education:Research · January 2011

DOI: 10.28945/1357

CITATIONS
16

READS
177

1 author:

Aristogiannis Garmpis
Technological Educational Institute of Western Greece
28 PUBLICATIONS **84** CITATIONS

SEE PROFILE

Design and Development of a Web-based Interactive Software Tool for Teaching Operating Systems

Aristogiannis Garmpis
Department of Applied Informatics in Management & Economy
Technological Educational Institute of Messolonghi,
Messolonghi, Greece

agarbis@teimes.gr

Execute Summary

Operating Systems (OS) is an important and mandatory discipline in many Computer Science, Information Systems and Computer Engineering curricula. Some of its topics require a careful and detailed explanation from the instructor as they often involve theoretical concepts and somewhat complex mechanisms, demanding a certain degree of abstraction from the students if they are to gain a full understanding. In this paper an overview of an interactive e-learning and web-based software tool is provided, which has been designed and developed for undergraduate university students of the Department of Applied Informatics in Management and Economy, Technological Educational Institute of Messolonghi, in Messolonghi, Greece. The aim of this software development was the self learning promotion related to memory management operations and especially the page replacement algorithms operation to be used in everyday OS classrooms. Thus undergraduate students can easily explore the operations of those algorithms through an interaction with the software. More specifically, students can explore each algorithm's mechanism separately and learn from their mistakes as shown automatically by the software in real time. All students' performances are stored in a database. This paper also proposes a study plan to examine the intention of students to use the software in their learning through a survey of a sample of undergraduates. The software does not intend to render obsolete or replace existing pedagogical approaches but instead will complement the existing teaching and learning methods of Operating Systems.

Keywords: Operating Systems, Page replacement algorithms, web based distance learning, e-Learning, Instructional Tool.

Introduction

Some educators propose short projects like shell programming exercises, source code modification exercises, and exercises using simulators for teaching a course on operating systems. Working on a real operating system is also prescribed. These proposals rely on supervised labs where students maintain direct contact with the shell command language and program development by the use of documented system calls. The

Editor: Thomas M. Connolly

programs so developed may include some of their own system service routines or implementations of classic algorithms such as the ones for communication and synchronization among processes.

Short projects propose shell interaction and programming using system calls. Programs can include developing inter-process communication (IPC) and synchronization programs, system utilities or some other classic OS algorithm (Downey, 1999; Pérez-Dávila, 1995; Ramakrishman & Lancaster, 1993; Wagner & Ressler, 1997). Short projects are easy to implement and there is no doubt that they really help students to better understand OS concepts (Maia, Machado, & Pacheco, 2005).

Source code modification can be done using instructional OS, e.g., Minix (Tanenbaum, 2001) and Xinu (Comer, 1984), or conventional open source OS, e.g., FreeBSD (http://www.freebsd.org) and Linux (http://www.linux.org). To take advantage of this kind of lab, students need a very good knowledge of computer architecture, the substrate operating system Unix, and C/C++ programming. Because of this, a course based on a code modification exercise is hard to implement.

On the other hand, simulators like BACI (Bynum & Camp, 1999), NACHOS (Anderson, Christopher, & Procter, 1999), OSP (Kifer & Smolka, 1992) and RCOS.java (Jones & Newman, 2001) are easier to implement than source code modification. Nevertheless, the process to learn, install, and run such tools frequently takes time to master its commands and requires some knowledge of Unix and good knowledge of programming in either C, Pascal, or Java. The Modern Operating Systems Simulator – MOSS (http://www.ontko.com/moss/memory/memory.exe) memory management simulator developed by Alex Reeder (Ontko, 2001) allows students to collect real page replacement statistics and, thus, compare the performance of various page replacement algorithms (Tanenbaum, 2001). Nevertheless, class projects based on MOSS require students to have Java programming skills to implement replacement algorithms that they want and integrate them into the MOSS simulator code files. This can be a complicated and time-consuming process. As a consequence, many courses on operating systems do not have the resources to fulfill these requirements and, if they do, the course requires an extended amount of teaching time.

Some other graphical simulators, such as SOsim (Maia & Pacheco, 2003) based on constructivist ideas, have been used as educational tools to support the teaching of operating systems. However, web-based experimental distance learning environments such as WebgeneOS (Buendía & Cano, 2006) or similar (Garcia, Rodriguez, Rosales, & Pedraza, 2005; Malmi, Korhonen, & Saikkonen, 2002; Tartaglia & Tresso, 2002), which complement traditional content-based learning platforms by including practical activities on operating systems subjects, have been used to lead students through a more active participation in the learning. They provide asynchronous communication between students and instructors allowing students to send their system programs using web forms, execute them in a native operating system, and receive their results and additional feedback information through web browsers.

The aim of this study was to design a software tool (Alg_OS V2.0) by using internet technology to serve the learning purposes of the memory management and especially the page replacement algorithms operation to be used in everyday OS classrooms. However, a study to investigate some of the factors that influence undergraduate students' intention to use the Alg_OS (V2.0) for learning purposes has been designed. The objectives of this future study are to: 1) investigate the influence of perceived usefulness and perceived ease-of-use on students' attitudes towards the use of the Alg_OS (V2.0) in their learning; 2) investigate the influence of perceived usefulness, attitude and subjective norm on students' intention to use the Alg_OS (V2.0) in their learning; and 3) examine the predictive validity of the Technology Acceptance Model (TAM) in predicting students' intention to use of the Alg_OS (V2.0) in their learning.

This web-based software tool (http://algos.cislab.epdo.teimes.gr) was developed by using MS-Visual Basic computer language, MS-Silverlight, and ASP.NET, supported by the MS-SQL Server 2008 database. The software tool allows undergraduate students to explore easily the operation, the differences between and the performance of page replacement algorithms (Optimal, FIFO, LRU, and CLOCK) simulating the management operation of physical memory through an interaction with Alg_OS (V2.0) user interface. More specifically, students can develop and explore each algorithm's mechanism separately and learn from their mistakes that the software shows automatically in real time. One of the most important aspects of a paged virtual memory system is the replacement policies. These policies define the selection criterion applied to choose one or more pages for memory eviction when another page has to be loaded in memory and there is no space available. The main difficulty is to decide which page is the least important one, so that it can be evicted. The use of an efficient strategy is primordial because a good memory management is an essential feature of high performance systems (Midorikawa, Piantola, & Cassettari, 2008). There are several different algorithms in choosing such a page, but the present study dealt with those that are used widely (Stallings, 2003).

The Alg_OS (V2.0) as a teaching and learning computer information system, based on the principle of the curricula recommendations "*All computer science students must learn to integrate theory and practice*" (The Joint Task Force on Computing Curricula, 2001, p.12), clearly demonstrates the behavior of these page replacement algorithms through a part of training lessons and provides a convenient way to illustrate page faults and their corresponding page fault costs. Equipped with a dynamic user interface for page table visualization, it allows the student to experiment and helps them understand how page tables operate and which pages page replacement algorithms evict in case of a page fault. Moreover, students can ascertain the acquired knowledge through an automated examination process. The Alg_OS (V2.0) also helps instructors keep track of what their students are doing, providing them with utilities to guide and evaluate the students' progress.

Alg_OS (V2.0), being different than the existing software tools to teach courses on page replacement algorithms, provides an evaluation system for students' actions and the motivation to improve their cognitive background.

Design and Implementation of Alg_OS (V2.0)

The Alg_OS (V2.0) architecture is based on web technology, which enables any User to interact with Alg_OS (V2.0) components through a web browser. A database has been created by using SQL Server 2008, initially with two basic Tables: one for the students and one for the page replacement algorithms. The Student Table consists of the following fields: "name" labeled "AM" (Primary Key), "Name" for student's first name, "Lastname" for student's last name, and Password (Figure 1).

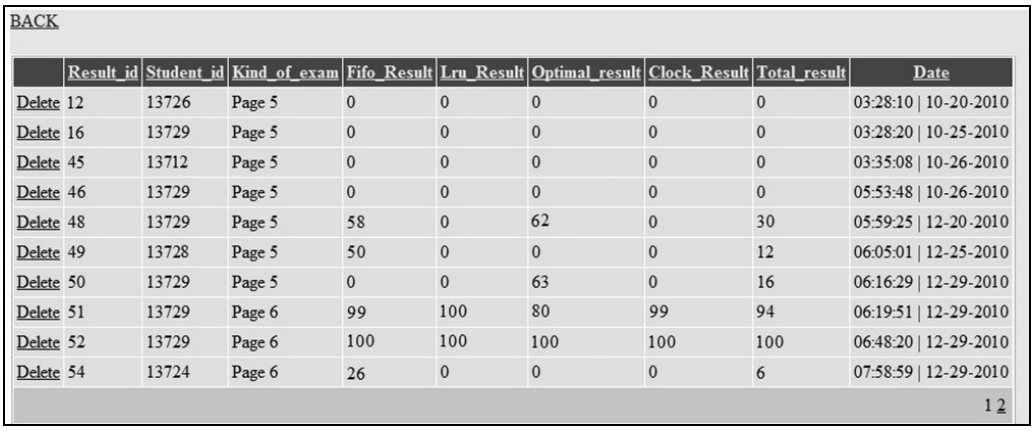
BACK

	AM	Name	Last_Name	password
Edit Delete	12345	aris	garmpis	12345
Edit Delete	13729	Nikolaos	Gouvatsos	212121
Edit Delete	13730	Nikolaos	gouvatsos	212121

Figure 1. The table "Student"

The Page Replacement Algorithms Table consists of the fields: "algorithm_id" (Primary Key), "algorithm_kind" (options: "Optimal", "FIFO", "LRU", "Clock") and "Training" (type: Yes/No). Both the Student and the Page replacement algorithms tables are connected to a new Table named Results for students' performance, which consists of the fields: "Result_id" (Primary Key),

"Kind_of_exam", "Fifo_Result", "Lru_Result", "Optimal_result, "Clock_Result", "Total_result" and "Date" (type: date and time) (Figure 2).



BACK

	Result_id	Student_id	Kind_of_exam	Fifo_Result	Lru_Result	Optimal_result	Clock_Result	Total_result	Date
Delete	12	13726	Page 5	0	0	0	0	0	03:28:10 | 10-20-2010
Delete	16	13729	Page 5	0	0	0	0	0	03:28:20 | 10-25-2010
Delete	45	13712	Page 5	0	0	0	0	0	03:35:08 | 10-26-2010
Delete	46	13729	Page 5	0	0	0	0	0	05:53:48 | 10-26-2010
Delete	48	13729	Page 5	58	0	62	0	30	05:59:25 | 12-20-2010
Delete	49	13728	Page 5	50	0	0	0	12	06:05:01 | 12-25-2010
Delete	50	13729	Page 5	0	0	63	0	16	06:16:29 | 12-29-2010
Delete	51	13729	Page 6	99	100	80	99	94	06:19:51 | 12-29-2010
Delete	52	13729	Page 6	100	100	100	100	100	06:48:20 | 12-29-2010
Delete	54	13724	Page 6	26	0	0	0	6	07:58:59 | 12-29-2010

1 2

Figure 2. The Results Table (Screen shot).

The corresponding entity relationships diagram (Connolly & Begg, 2005) is shown in Figure 3. Initially, both Student and Page_replacement_algorithms entities were related through a many-to-many relationship. Subsequently, this relationship was transformed into two different one-to-many relationships by the addition of the intermediate entity Results.

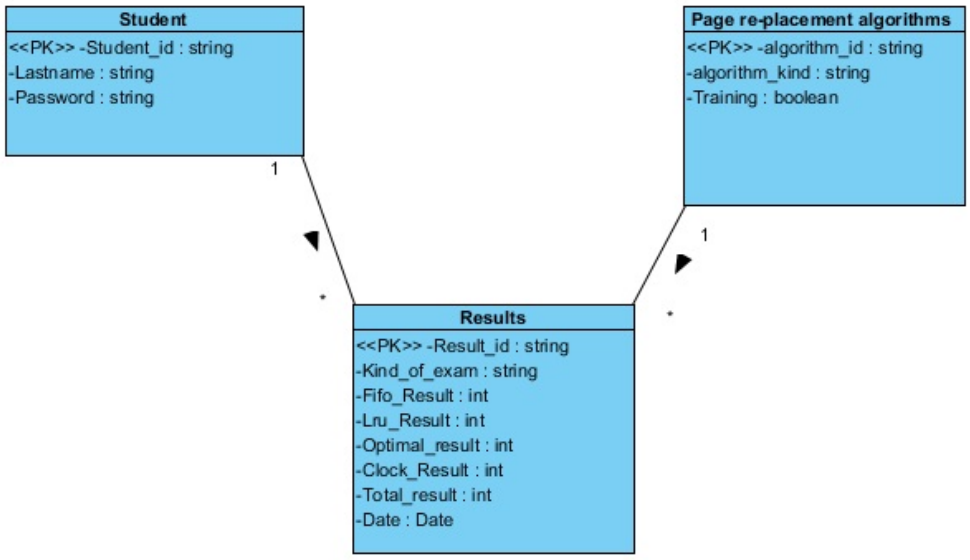


Figure 3. The class diagram showing the entities/classes and the attributes. Student to Results (One-to-many) and Page replacement algorithms to Results (One-to-many).

The interaction between the system and entities external to the system is usually represented by a *use case diagram*. These external entities are referred to as *actors*. Actors represent roles that may include human users, external hardware, or other systems. Figure 4 provides an overview of the use cases and actors (Administrator and Student) for the Alg_OS (V2.0) software. Thus, administrator (or a student) as an actor can log into the Alg_OS (V2.0) system by entering his/her name and password. Alg_OS (V2.0) validates the entered name and password and logs the actor into the system.

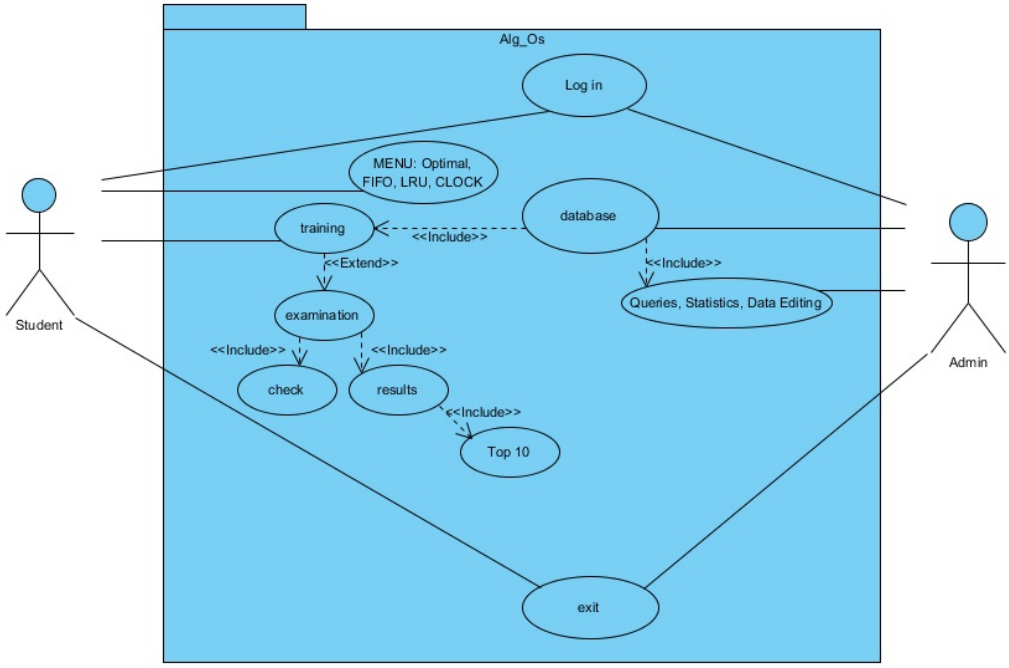


Figure 4. Use case diagram of Alg_OS (V2.0)

The actions of logged on students are listed below:

- choose the algorithm type (Optimal, FIFO, LRU, CKLOK)
- participate in a training course (it is mandatory for all students at least once)
- participate in an examination process
- logout.

Alg_OS (V2.0) automatically shows students' mistakes, the corrections, and a list of the best student performances ("top ten").

The actions of a logged on Administrator are:

- edit database entries (queries, data, statistics)
- edit students' performances (training or exams)
- logout.

The web pages that are integrated in the Alg_OS (V2.0) software have been created using MS Visual Basic 2008, an environment capable of easily creating, updating, and manipulating web pages.

Alg_OS (V2.0) uses a simple, visual, user friendly web-interface (Figure 5).

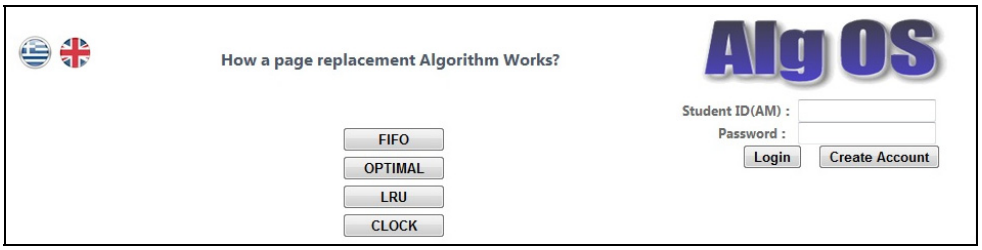


Figure 5. The main menu of Alg_OS (V2.0)

Initially, students have to create a user account and enter their personal data. Alg_OS (V2.0) automatically stores these in a database system as a logged in user with the current time and date. Afterwards, a specific training course comes up to assist students to understand the meaning of the page replacement algorithms, such as Optimal (Most optimal replacement), FIFO (First In First Out), LRU (Least Recently Used) and CLOCK, and to improve their knowledge. During the training, the Alg_OS (V2.0) system automatically shows a references string of memory pages (a sequence of 20 different values randomly) and also a table that simulates the computer's physical memory frames. Students can enter values in the cells of that table according to the concept of the selected algorithm (Figure 6). For any wrongly entered value, the Alg_OS (V2.0) system displays an appropriate message helping the students to understand the importance of the specific algorithm in the memory operation. This training course is mandatory for all students at least once while the Alg_OS (V2.0) software automatically stores the personal data of those students who have completed the training course in the database.

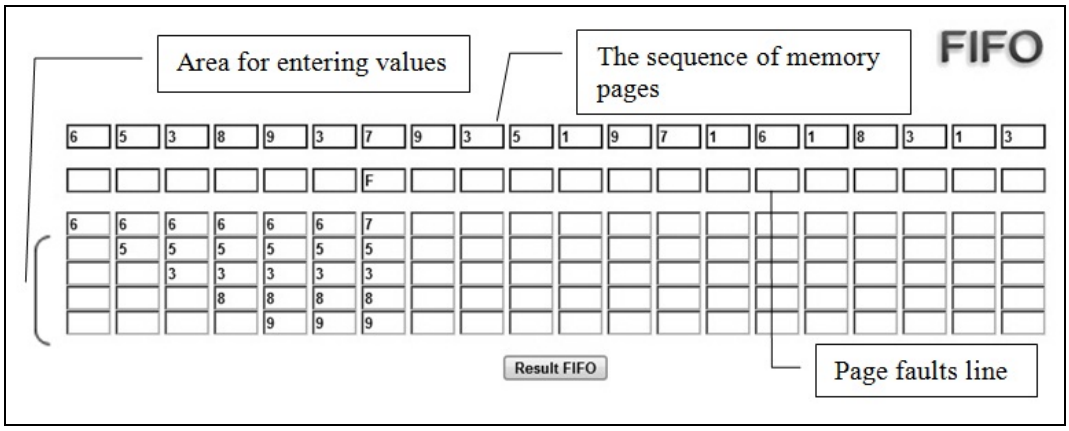


Figure 6. The simulation of memory frames

The Optimal Page Replacement Algorithm

The best possible page replacement algorithm is easy to describe but impossible to implement; it is unrealizable. The optimal page algorithm simply states that the page with the highest label should be removed. If one page will not be used for 8 million instructions and another page will not be used for 6 million instructions, removing the former pushes the page fault that will fetch it back as far into the future as possible. Computers, like people, try to put off unpleasant events for as long as they can.

At the time of the page fault, the operating system has no way of knowing when each of the pages will be referenced next. Still, by running a program on a simulator and keeping track of all page references, it is possible to implement optimal page replacement on the second run by using the page reference information collected during the first run. In this way it is possible to compare the performance of realizable algorithms with the best possible one (Tanenbaum, 2001). An optimal algorithm is useful for evaluating page replacement algorithms; it is of no use in practical systems as the OS cannot look into the future to know how long it will take to reference every page again.

Students' actions for the Optimal algorithm operation and learning achieved through interaction with the Alg_OS (V2.0) software is as follows:

1. Pages are placed in succession in the available memory frames, according to arrival time.
2. When there are not available memory frames, the page replacement procedure begins.
3. Each page replacement produces a fault (F).

4. At the moment of a page fault:
   a) Label each page in memory with the number of instructions that will be executed before that page is first referenced.
   b) Replace the page that is least likely to be referenced or for which the time to the next reference is the longest (Replace page needed at the farthest point in future).
5. If a page that will come back is already registered in a memory frame the process continues normally and without fault production (F).

Students can enter in cells both the pages' address values and the character "F" (to declare a fault), according to the Optimal algorithm concept. When the Optimal algorithm implementation is completed, students can check their mistakes by pressing the button "Check", at which point the Alg_OS (V2.0) automatically evaluates students' choices (Figure 7). However, students can repeat this procedure in the effort to understand the Optimal algorithm concept. The algorithms presented below are useful on real systems.

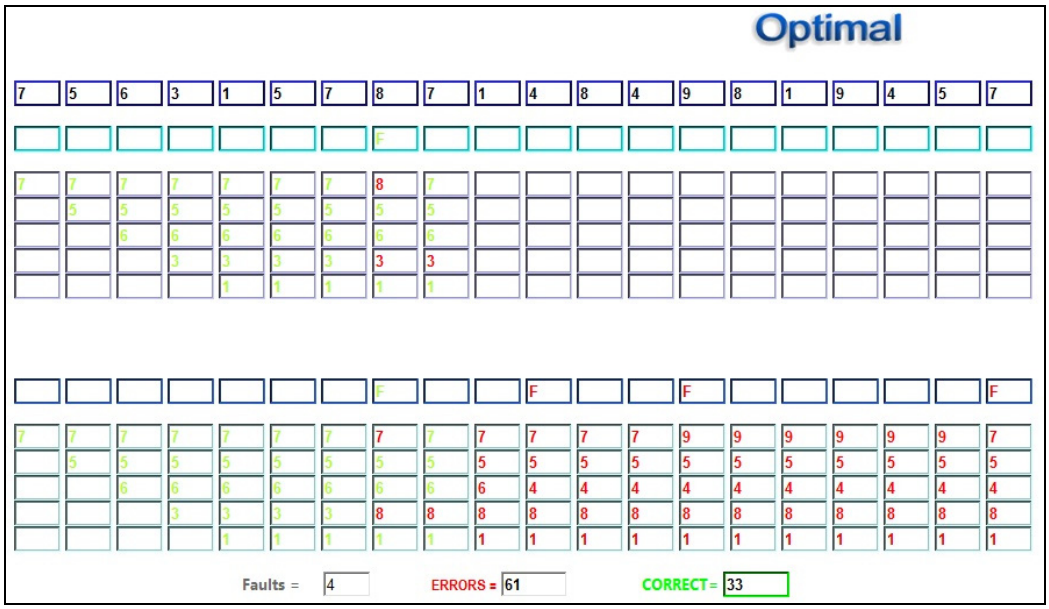


Figure 7. Screen shot of Alg_OS (V2.0) evaluation process (correct values with Green color and the mistakes with Red color).

The First-In First-Out (FIFO) Page Replacement Algorithm

The operating system maintains a list of all pages currently in memory, with the page at the head of the list the oldest one and the page at the tail the most recent arrival. On a page fault, the page at the head is removed and the new page added to the tail of the list. There is a chance that the oldest page may be used heavily (thrashing - page moves back and forth between memory and disk). When the form labeled FIFO appears, students can implement the First-In First-Out page replacement algorithm through an interaction with the Alg_OS (V2.0) software as follows:

1. Pages are placed in succession in the available memory frames, according to arrival time.
2. When there are no available memory frames, the page replacement procedure begins.
3. Each page replacement produces a fault (F).
4. Replace the page that has been in memory for the longest time
5. If a page that will come back is already registered in a memory frame the process continues normally and without fault production (F).

First-In First-Out page replacement algorithm (FIFO) in its pure form is rarely used. It is clearly not particularly optimal because it might end up removing a page that is still referenced, since FIFO only looks at the page's age.

The Least Recently Used (LRU) Page Replacement Algorithm

A good approximation to the optimal algorithm is based on the observation that pages that have been heavily used in the last few instructions will probably be heavily used again in the next few. Conversely, pages that have not been used for some time will probably remain unused for a long time. This idea suggests a realizable algorithm: when a page fault occurs, throw out the page that has been unused for the longest time. This strategy is theoretically realizable and it is called LRU (Least Recently Used) paging (Tanenbaum, 2001). Students can implement the LRU page replacement algorithm through an interaction with the Alg_OS (V2.0) software and according to the following actions:

1. Pages are placed in succession in the available memory frames, according to arrival time.
2. When there are not available memory frames, the page replacement procedure begins.
3. Each page replacement produces a fault (F).
4. The page that has been unused for the longest time is replaced.
5. If a page which will come back is already registered in a memory frame the process continues normally and without fault production (F).

The Clock Page Replacement Algorithm

The CLOCK algorithm keeps a circular list of pages in memory, with the 'hand' (iterator) pointing to the oldest page in the list. When a page fault occurs and no empty frames exist, then the R (set when the page is referenced, read or written) bit is inspected at the hand's location.

a) If its R bit is 0, the page is evicted, the new page is inserted into the clock in its place and the hand is advanced one position.
b) If R is 1, it is cleared and the hand is advanced to the next page.

This process is repeated until a page is found with R = 0.

Students can implement the LRU page replacement algorithm through an interaction with the Alg_OS (V2.0) software as follows:

1. Pages are placed in succession in the available memory frames, according to arrival time.
2. When there are not available memory frames, the page replacement procedure begins.
3. Each page replacement produces a fault (F)
4. When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:
   i) R=0, Evict the page.
   ii) R=1, Clear R and advance hand (Figure 8).

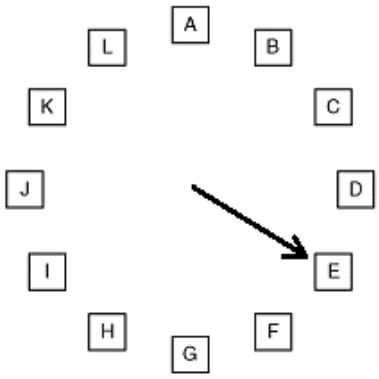


Figure 8. The circular list of pages in memory. The oldest page in this list is the page E.

The Alg_OS (V2.0) form for the CLOCK algorithm implementation provides two specific buttons (colored yellow and light blue) allowing the students to indicate both the 'hand' (iterator, '>') and the asterisk symbol '*' (if R=1) (Figure 9).

Figure 9. The Alg_OS form for CLOCK (screen shot).

The Examination Process

Students can participate in an examination process provided by Alg_OS (V2.0) so that they can check their understanding for the operation of the algorithms. The participants of this process can choose both the type of algorithm and/or the level of its difficulty. For that reason two different levels are available. In Level 1, Alg_OS (V2.0) shows six places and nine pages of a simulated memory system, while in Level 2 shows five places and nine pages. When the examination process starts, Alg_OS (V2.0) automatically shows 20 different values of memory pages randomly and an empty table that simulates the computer memory frames. These values remain precisely the same for each algorithm selected by the students, so the comparison and the finding of the most efficient algorithm are achieved. Students can enter the required values in the cells of that table according to the logic of the selected algorithm without any software help. Students have to be examined in all available algorithms provided by the software. At the end of the examination process Alg_OS (V2.0) automatically displays the appropriate algorithm's implementation and the student's mistakes and stores the student's score separately in a database. Additionally a classification list ('top-ten') of all students' IDs and the best scores is displayed, which motivates

each student to improve his/her position in the list. The instructor, as an Alg_OS (V2.0) administrator, can observe the students' cognitive progress exploring stored data for every type of algorithm.

Future Research

The Technology Acceptance Model (TAM) was developed by Davis, Bagozzi, & Warshaw (1989) to explain computer-usage behavior. The theoretical basis of the model was Fishbein and Ajzen's Theory of Reasoned Action (1975). The goal of TAM is "*to provide an explanation of the determinants of computer acceptance that is general, capable of explaining user behavior across a broad range of the end-user computing technologies and user populations, while at the same time being both parsimonious and theoretically justified*" (Davis et al. 1989, p. 985). According to TAM, shown in Figure 10, an individual's technology acceptance decision is determined by his or her behavioral intention which is underpinned by his or her attitude towards the use of technology. Attitude toward the behavior (AB) is the person's positive or negative evaluation of performing the behavior. Attitude towards use is determined by beliefs towards a technology's usefulness and ease of use, as perceived by an individual.

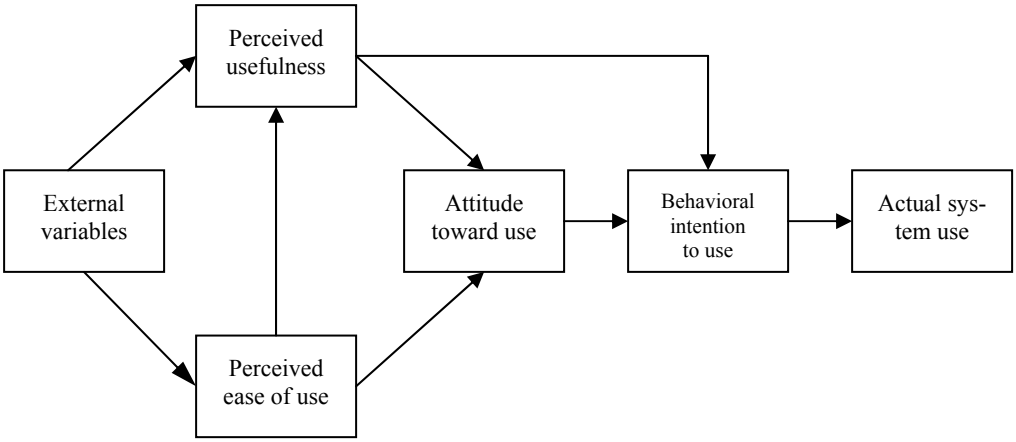


Figure 10. The Technology Acceptance Model.

Perceived usefulness is defined as "*the degree to which a person believes that using a particular system would enhance his or her job performance*" (Davis, 1989, p. 320). Perceived ease of use is defined as the "*the degree to which a person believes that using a particular system would be free from effort*" (Davis, 1989, p. 320). Furthermore, TAM theorizes that external factors, such as the task, user characteristics, and organizational contexts, are expected to influence technology acceptance behavior indirectly by affecting attitudes or intention.

The TAM has been successfully applied in numerous settings of Information Technology and has been shown to be a very good predictor of ICT use. For example, TAM is used in very different settings, in order to test the acceptance of Internet utilization behavior (Lai & Honglei, 2005), online shopping (Vijayasarathy, 2004), online learning (Saadé, & Bahli, 2004) and Internet banking (Shih, 2004).

While being powerful in helping predict user acceptance, TAM does not include social factors that may have important influences on attitude, intention and behavior (for a review, see Saadé & Bahli, 2004). In the literature review there are many studies that have expanded the TAM, and some researchers suggested that it should be modified to include other components in order to explain users' intention and behavior better (for example: Agarwal & Prasad (1997); Lucas &

Spitler (1999); Venkatesh & Davis (2000)). More recently, Venkatesh and Davis (2000) extended TAM to include variables relevant to social influence and cognitive instrumental processes. This extended model, called TAM2, includes subjective norms and was tested with longitudinal research studies.

Subjective norm is a component of the Theory of Planned Behavior (TPB) (Ajzen, 2006). Subjective norm (SN) is the person's perception of social normative pressures, that is, that relevant others believe he or she should or should not perform the behavior. It is assumed to have two components that work in interaction: beliefs about how other people, who may be in some way important to the person in question, would like them to behave (i.e. normative belief strength) (bi) and the positive or negative judgments about each belief (i.e. motivation to comply) (mi). The following equation (see Equation 1) represents this relationship:

$$SN = \sum_{i=1}^{n} b_i \, m_i$$

Equation 1

- **Participants and procedure**

The survey participants will be a set of a final-year undergraduate university students of the Department of Applied Informatics in Management and Economy, Technological Educational Institute of Messolonghi, Messolonghi, Greece. This set of students will be already familiarized with the use of Alg_OS (V2.0) software before they answer a suitable questionnaire (see the Appendix). The kind of questions that will be asked will be about students' intentions, attitudes, perceived usefulness, and perceived ease of use of Alg_OS (V2.0).

- **Questionnaire**

The questionnaire contains items designed to measure the variables of the Technology Acceptance Model (TAM). The questions have been phrased and scaled exactly as recommended by the authors of the Theory of Reasoned Action, Technology Acceptance Model and Theory of Planned Behavior (Ajzen & Fishbein, 1980). Each question is scored from 1 (negative end) to 7 (positive end). The questionnaire was adapted from a previous study (Ajzen, 1991).

- **Intention**

According to TAM, intention can be measured using 7-point scales for these three items: 1) "I intend to use the Alg_OS (V2.0) during the next academic year: extremely unlikely/extremely likely"; 2) "I will try to use the Alg_OS (V2.0) during the next academic year: definitely true/definitely false"; 3) "I plan to use the Alg_OS (V2.0) during the next academic year: strongly disagree/strongly agree)". The average responses to these three items will yield a measure of intention to use the Alg_OS (V2.0).

- **Attitudes towards use**

Students' attitude will be measured by five items using a 7-point scale: "For me to use the Alg_OS (V2.0) would be: harmful/beneficial, pleasant/unpleasant, good/bad, worthless/valuable, enjoyable/unenjoyable". The average responses to the five items will provide an attitude towards the use measure.

- **Perceived usefulness**

The perceived usefulness component will be measured in the questionnaire with the following four items: 1) "Using the Alg_OS (V2.0) in my learning enables me to accomplish tasks more

quickly"; 2) "Using the Alg_OS (V2.0) enhances the quality of my learning"; 3) "Using the Alg_OS (V2.0) would make it easier to do my learning activities"; 4) "I find the Alg_OS (V2.0) useful in my learning". Each item is rated on a scale of 1 to 7 (Strongly Disagree to Neutral to Strongly Agree). The average response to the four items will serve as a measure of perceived usefulness.

- **Perceived ease of use**

The perceived ease of use will be measured using the following five items: 1) "It is easy for me to become skillful at using Alg_OS (V2.0)"; 2) "My interaction with Alg_OS (V2.0) is clear and understandable"; 3) "I find Alg_OS V2.0 easy to use"; 4) "Learning to use Alg_OS (V2.0) is easy for me"; 5) "I find Alg_OS (V2.0) to be flexible to interact with". Each item is rated on a scale of 1 to 7 (Strongly Disagree to Neutral to Strongly Agree). The average response to the five items will serve as a measure of perceived ease of use.

- **Subjective norm**

Subjective norm will be measured by the following three items on a 7-point scale that ranges from 1 to 5: 1) "It is expected of me that I will use the Alg_OS (V2.0) during the next academic year: extremely likely/extremely unlikely"; 2) "Most people who are important to me think that I should/I should not use the Alg_OS (V2.0) during the next academic year"; 3) "The people in my life, whose opinions I value would approve/disapprove of my using the Alg_OS (V2.0) during the next academic year". The average responses will yield a measure of subjective norm.

Discussion and Conclusions

In this paper, a working prototype of an interactive web-based software tool (Alg_OS V2.0) has been presented to address the learning weaknesses of undergraduate students studying Operating Systems. More specifically, the aim of the software was promoting self learning related to memory management operations and especially the page replacement algorithms operation to be used in the OS classrooms, complementing existing teaching and learning methods. The software tool requires some upgrades:

i) technical improvements,
ii) extension and in other algorithms types, and
iii) improvement of English version software,

which will be completed in the near future.

Additionally, the development of an evaluation framework connected with the perceived usefulness and ease of use should improve the overall quality of Alg_OS (V2.0), while an evaluation survey of students' attitudes towards using this software in their learning is judged essential.

References

Agarwal, R., & Prasad, J. (1997). The role of innovation characteristics and perceived voluntariness in the acceptance of information technologies. *Decision Sciences, 28*(3), 557-582.

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes, 50*(2), 179-211.

Ajzen, I. (2006). *TPB questionnaire construction: Constructing a theory of planned behavior questionnaire*. Retrieved September, 30, 2010 from http://www-unix.oit.umass.edu/~aizen/pdf/tpb.measurement.pdf

Ajzen, I., & Fishbein, M. (1980). *Understanding attitudes and predicting social behavior.* Englewood Cliffs, NJ: Prentice-Hall.

Anderson, T. E., Christopher, W. A., & Procter, S. J. (1999). *The Nachos instructional operating system*. Retrieved June 22, 2010 from: http://www.cs.washington.edu/homes/tom/nachos/

Buendía, F., & Cano, J-C. (2006). bgeneOS: A generative and web-based learning, architecture to teach operating systems in undergraduate courses. *IEEE Transaction on Education, 49*(4), 464-473.

Bynum, B., & Camp, T. (1999). After you, Alfonse: A mutual exclusion toolkit - An introduction to BACI. Retrieved June 22, 2010, from: http://www.mines.edu/fs_home/tcamp/baci/

Comer, D. (1984). *Operating system design – The XINU approach*. Englewood Cliffs, NJ: Prentice-Hall.

Connolly T. M., & Begg C. E. (2005). *Database systems: A practical approach to design, implementation, and management* (pp.836-846). Reading, MA: Addison - Wesley.

Davis, F. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly, 13*(3), 319-340.

Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). User acceptance of computer technology: A comparison of two theoretical models. *Management Science, 35*(8), 982-1003.

Downey, A. B. (1999). Teaching experimental design in an operating systems class. *Proceedings of the 30th ACM SIGCSE, Bulletin, 31*(1), 316-320.

Fishbein, M., & Ajzen, I. (1975). *Belief, attitude, intention and behavior: An introduction to theory and research*. Reading, MA: Addison-Wesley.

Garcia, A., Rodriguez, S., Rosales, F., & Pedraza, J. (2005). Automatic management of laboratory work in mass computer engineering courses. *IEEE Transaction on Education, 48*(1), 89–98.

The Joint Task Force on Computing Curricula. (2001). *IEEE Computer Society and Association for Computing Machinery. Computing Curricula,* (p.12). Retrieved June 22, 2010 from: http://www.acm.org/education/curric_vols/cc2001.pdf

Jones, D., & Newman, A. (2001). RCOS.java: A simulated operating system with animations. *Proceedings of CBLIS'2001, Brno, The Czech Republic.* Retrieved July 22, 2010 from: http://cblis.utc.sk/cblis-cd-old/2001/text/pdf/c4.pdf

Kifer, M., & Smolka, S. (1992). OSP: An environment for operating systems. *Operating Systems Review, 26*(4), 98-99.

Lai, V. S., & Honglei, L. (2005). Technology acceptance model for internet banking: An invariance analysis. *Information and Management, 42*(2), 373-386.

Lucas, H. C., & Spitler, V. K. (1999). Technology use and performance: A field study of broker workstations. *Decision Sciences, 30*(2), 291-311.

Maia L. P., & Pacheco, A. C., Jr. (2003). A simulator supporting lectures on operating systems. *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference, Boulder, CO, USA, 2,* F2C-17.

Maia, L. P., Machado, F. B., & Pacheco, A. C., Jr. (2005). A constructivist framework for operating systems education: A pedagogic proposal using the SOsim. *Proceedings of the 10th annual SIGCSE Conference on Innovation and technology in computer science education (ITiCSE'0), Monte de Caparica, Portugal, 37*(3), 218-222.

Malmi, L., Korhonen, A., & Saikkonen, R. (2002). Experiences in automatic assessment on mass courses and issues for designing virtual courses. *Proceedings of the 7th Annual SIGCUE Conference on Innovation and Technology in Computer Science Education, ACM,* 55-59.

Midorikawa, E. T., Piantola, R., L., & Cassettari, H. H. (2008). On adaptive replacement based on LRU with working area restriction algorithm. *ACM SIGOPS Operating Systems Review, 42*(6), 81-92.

Ontko, P. (2001). *MOSS memory management simulator user guide.* Retrieved June 22, 2010 from: http://www.ontko.com/moss/memory/user_guide.html

Pérez-Dávila, A. (1995). OS bridge between academia and reality. *Proceedings of the 26th ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE Bulletin, 27*(1), 146-1481.

Ramakrishman, S., & Lancaster, A. M. (1993). Operating system projects: Linking theory, practice, and use. *Proceedings of the 24th ACM SIGCSE Bulletin, 25*(1), 256-260.

Saadé, R., & Bahli, B. (2004). The impact of cognitive absorption on perceived usefulness and perceived ease of use in on-line learning: An extension of the Technology Acceptance Model. *Information and Management Journal, 42*(2), 317-327.

Shih, H. P. (2004). Extended technology acceptance model of Internet utilization behavior. *Information and Management Journal, 41*(6), 719-729.

Stallings, W. (2003). *Operating systems: Internal and design principles* (4th ed., p.473). Thessaloniki, Greece: Tziolas Publications.

Tanenbaum, A. (2001). *Modern operating systems* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

Tartaglia, A., & Tresso, E. (2002). An automatic evaluation system for technical education at the university level. *IEEE Transaction on Education, 45*(3), 268–275.

Venkatesh, V., & Davis, F. D. (2000). A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management Science, 46*(2), 186-204.

Vijayasarathy, L. R. (2004). Predicting consumer intentions to use online shopping: The case for an augmented technology acceptance model. *Information and Management, 41*(6), 747-762.

Wagner, T. D., & Ressler, E. K. (1997). A practical approach to reinforcing concepts in introductory operating systems. *ACM SIGCSE Bulletin*. *Proceedings of the 28th SIGCSE Technical Symposium on Computer science education, 29*(1), 44-47.

Appendix

Questionnaire for Alg_OS V2.0 Evaluation

The basic aim of this questionnaire is to record students' perceptions while they interact with the e-learning software Alg_OS V2.0, to learn the operating mechanism of page replacement algorithms in operating systems, the software friendliness and usability, the intentions of continuous use, and also the subjective judgments for the knowledge acquisition.

The recorded data of this questionnaire will be used for research purposes. However, the findings will be used for Alg_OS V2.0 software upgrading and maybe for the design and development some other e-learning software tools on the same subject in the future.

1. *Personal data*

1.1. Gender: Man ☐ Woman ☐

1.2 Semester: _____

1.3 How many times have you failed on the final exam, while you were prepared? 1 time ☐ 2-4 times ☐ more than 5 times ☐

1.4 Is the examination process difficult? Yes ☐ No ☐

1.5 Please, circle your answer that is close to your opinion (for each line below).

	Nothing	A bit	Some	Enough	Too much
I believe that I know the operation of page replacement algorithms… (Optimal, FIFO, LRU, CLOCK);	1	2	3	4	5
	Nothing	A bit	Medium	Enough	Too much
I believe that the e-learning software usage be ease …	1	2	3	4	5

1. *Intention*

1.1 I intend to use Alg_OS V2.0 software during next academic year.

unlikely	1	2	3	4	5	6	7	likely

1.2 I will try to use Alg_OS V2.0 software during next academic year.

It is true	7	6	5	4	3	2	1	It is untrue

1.3. I plan to use Alg_OS V2.0 software during next academic year.

I disagree	1	2	3	4	5	6	7	I agree

2. *Students' attitude towards the system usage*

Alg_OS V2.0 usage during the next academic year, for me is:

harmful	1	2	3	4	5	6	7	beneficial
pleasant	7	6	5	4	3	2	1	nasty
good	7	6	5	4	3	2	1	bad
worthless	1	2	3	4	5	6	7	valuable
enjoyable	7	6	5	4	3	2	1	unenjoyable

3. *Perceived usefulness*

1	Using the **Alg_OS (V2.0)** in my learning enables me to accomplish tasks more quickly.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree
2	Using the **Alg_OS (V2.0)** enhances the quality of my learning.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree
3	Using the **Alg_OS (V2.0)** would make it easier to do my learning activities.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree
4	I find the **Alg_OS (V2.0)** useful in my learning.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree

4. *Perceived ease of use*

1	It is easy for me to become skillful at using **Alg_OS (V2.0)**.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree
2	My interaction with **Alg_OS (V2.0)** is clear and understandable.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree
3	I find **Alg_OS V2.0** easy to use.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree
4	Learning to use **Alg_OS (V2.0)** is easy for me.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree
5	I find **Alg_OS (V2.0)** to be flexible to interact with.	absolutely agree 7 6 5 4 3 2 1 absolutely disagree

5. *Subjective standards*

5.1. I think that the most important people of TEI of Messolonghi, (e.g. my colleagues and my Teachers) believe that I:

must	7	6	5	4	3	2	1	should not

use Alg_OS V2.0 during the next academic year.

5.2. People of TEI of Messolonghi, (e.g. my colleagues and my Teachers) whose their mind opinion estimate, will

endorse	7	6	5	4	3	2	1	condemn

Alg_OS V2.0 usage during the next academic year.

5.3. The most important people for me use software for their learning.

It is true	7	6	5	4	3	2	1	It is untrue

Biography

Aristogiannis Garmpis is an Assistant Professor at the Department of Applied Informatics in Administration and Economy in the Technological Educational Institute of Messolonghi (Greece). He holds a PhD in Applied Informatics (London South Bank University, London, UK); an MSc in Computing and Information Systems (University of Greenwich, London, UK) and a BSc in Mathematics (University of Athens, Greece). He is the Director of the "Data Bases and Computer Information Systems - CisLab" Lab in the same department. His research interests include, Computer Operating Systems, Web Mining, e-Learning, Computer Information Systems and Internet programming.