A Constructivist Framework for Operating Systems Education: a Pedagogic Proposal Using the SOsim

Luiz Paulo Maia
Federal University of Rio de Janeiro, NCE/UFRJ, PoBox 2.324, Rio de Janeiro, RJ, Brazil, 20001-970
55-21-9806-9180
lpmaia@training.com.br

Francis Berenger Machado
PUC-Rio, Rua Marquês de São Vicente 225, Depto. de Informática, Rio de Janeiro, RJ, Brazil, 22453-900
55-21-8807-9641
berenger@pobox.com

Ageu C. Pacheco Jr.
Federal University of Rio de Janeiro, NCE/UFRJ, PoBox 2.324, Rio de Janeiro, RJ, Brazil, 20001-970
55-21-2598-3256
ageu@nce.ufrj.br

ABSTRACT
A conventional teaching approach, when applied specifically to the discipline of Operating Systems (OS), seems to fall short of attaining the overall objective, sometimes leaving the lecturer unsure about the students' actual understanding of the dynamic nature of OS concepts and mechanisms. This paper presents a pedagogical proposal, based on constructivist ideas, as a means of making the process of learning OS more efficient and interesting. The framework presented here uses the SOsim graphical simulator as a support tool, creating a teaching and learning environment in which practical experiments can be undertaken as each OS topic is introduced and explained.

Categories and Subject Descriptors
K.3.2 [Computers & Education]: Computer & Information Science Education—*Computer Science Education*;
D.4.m [Operating Systems]: *Miscellaneous*.

Keywords
Computer science education, operating systems, pedagogy, simulation, visualization, graphics.

1. INTRODUCTION
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 lecturer 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.

Unlike other disciplines in the computer area, OS is a subject that does not exhibit a linear structure that allows the lecturer to progress through the topics in a sequential order. Even experienced lecturers know that the way they approach these topics plays a major role in the final results achieved by the students in their classes. "One of the main characteristics of the OS discipline resides in the relative difficulty in defining a clear didactics sequencing for its different topics" [16].

The traditional course model, in which the lecturer follows a text book, prepares and exhibits slides, and presents some theoretical exercises, is not enough to assure a precise comprehension of what is being taught. The problem is due to both the teaching model and the lack of appropriate tools capable of translating the theory being presented into a more practical reality. And without a practical vision the student tends to lose touch and just "float" around the introduced concepts and mechanisms without gaining a realistic view of what is really going on.

In recent years, the problems associated with OS teaching have been the main theme of some important research works [7, 11]. Amongst them, one line of investigation that has produced some positive results makes use of the constructivist method of teaching. Although well established in other areas, e.g. mathematics, constructivism has only appeared relatively recently in computer science [2].

In this paper we present a constructivist framework to be used and evaluated in the everyday OS classrooms. In section 2 we briefly relate some proposals to modify the OS environment that has been adopted by some undergraduate courses, and that we consider to provide only partial solutions. Next, we present a general description of the SOsim graphical simulator. In section 4 we discuss the underlying concepts of the constructivist theory and its potential application to OS teaching. Finally we present the results from an experiment carried out during an Information Systems undergraduate course at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil.

2. OPERATING SYSTEMS TEACHING
A course on OS that consists only of theoretical lectures does not necessarily guarantee that the students will obtain a full comprehension and absorption of the many concepts introduced. It is essential to reserve part of the program for laboratory classes and practical exercises. This section presents and discusses the most common practices used in laboratories in many undergraduate courses: (1) small practical projects, (2) modifications at the code level of the operating systems and (3) the use of simulators.

ITiCSE'05, June 27–29, 2005, Monte de Caparica, Portugal.

2.1 Small Practice Projects

Some researchers propose "closed labs" supported by real system working; in most cases, some version of the Unix operating system [7, 8, 17, 19, 21]. 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 programs so developed may include some of their own system service routine or classic algorithms such as the ones for communication and synchronization among processes. Most proposed labs require good knowledge of programming in either C, or Pascal, or Java.

2.2 Modifications in the OS Code

Teaching methods based upon making alterations to the OS source code may use one of two forms of environment: real open source code or educational systems. Open source code systems such as Linux and FreeBSD, already offer their code for analysis and modification. The main goal for this code, however, is the attainment of high levels of performance, making it very difficult for beginners to understand. Also, apart from complexity, an open source system comprises thousands of lines of code. Another awkward aspect is the absence of proper documentation supporting the use of those systems as teaching tools.

Educational operating systems such as Minix [20] and Xinu [6] are simplified versions of real open source systems, specifically developed to serve as a practical framework, for the study of the internal structure and working behavior of an operating system. A big advantage of these educational systems is the implicit availability of literature supporting teachers and students in the difficult task of understanding the system's source code.

Independent of the specific operating system being used, laboratories employing the source code analysis and modification approach face a series of restrictions. Firstly, they demand good prior knowledge of computer architecture, Unix and C/C++ programming from both instructors and students. The amount of time needed to install, alter and debug the system may compromise the whole process. Apart from that, many institutions may not have the necessary resources to undertake the laboratory implementation. As a consequence, many courses on operating systems do not have the resources to fulfill these requirements, and if they did so, the course would require an extended amount of teaching time.

2.3 Use of Simulators

A simulator attempts to create a dynamic and simplified model of reality. In educational applications we consider its potential far more efficient than other conventional tools. Within the sphere of computer science there are simulators supporting the teaching of various disciplines such as computer networks, programming techniques, computer architecture and operating systems. As examples of these we mention here: BASI [4], NACHOS [1], OSP [12] and RCOS.java [5, 10].

That simulators are just simpler versions of real systems does not necessarily mean that they are always easy to use. Each simulator has its own characteristics: some positive, some negative. Also, most of them demand some time to master its commands and working behavior, often requiring some previous basic programming and Unix knowledge.

3. THE SOsim SIMULATOR

The SOsim is a simulator with visual facilities that are mainly targeted at presenting the various concepts and techniques found within most modern multiprogramming operating systems in a more dynamic and clear way [13]. Some of the algorithms it implements can be found in commercial operating systems such as HP OpenVMS and Microsoft Windows NT/2000/XP. The simulator emulates the main subsystems of a multiprogramming operating system, as the process manager, the scheduler, and a paged virtual memory. The SOsim was developed to act as a support tool for OS teaching, allowing lecturers to introduce concepts and techniques in a clear, dynamic and animated environment, thus improving communication with the students and expanding their comprehension and understanding [14].

The program was devised for the Microsoft Windows platform and exhibits a simple and easy interface, making it possible for the user to visualize and monitor the various asynchronous events that occur in OS execution (Figure 1).

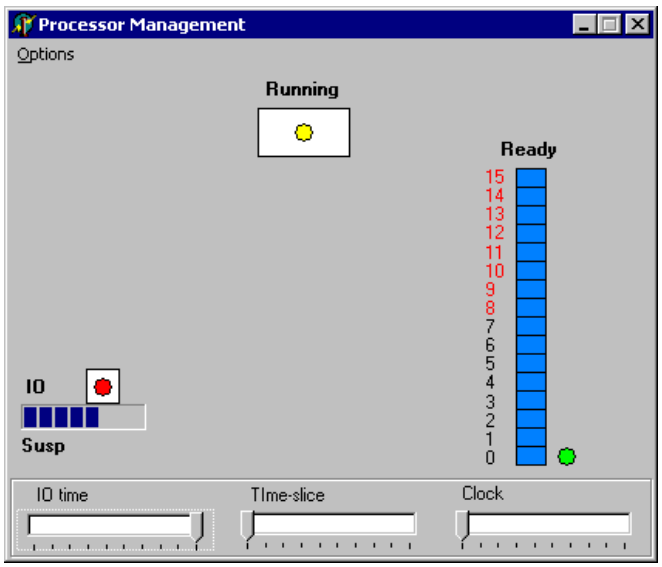

Figure 1. SOsim Simulator version 1.2

The simulator is available via the Internet for free downloading in Portuguese and English versions [15] and it was developed in such a way that translation into other languages can be done easily. There is already a group of international users applying the software as a supporting tool in the teaching of OS courses.

4. CONSTRUCTIVISM

The classic pedagogic model at all levels of education is based upon the instructive model, where instructional sequences tackle the task of transferring the maximum amount of information between an active teacher and a passive learner. In general, the instructive model tends to be standardized and homogenized in a sense that the teaching is mostly directed to the class as a whole, and not to individuals within the class.

One way to overcome the limitations imposed by the instructive model is to include concepts from the constructivist theory – the teacher/instructor plays not only the classic role of transmitting knowledge the best as he/she can, but also serving as a "facilitator" of the learning process. In the constructivist model

the student is the central focus of the whole process of knowledge construction. The development of his/her investigational/critical predicates and his ability to work cooperatively in group/teams are equally relevant tasks for the teacher.

4.1 The Constructivist Theory

Constructivist theory was originally conceived by Jean Piaget as a result of research that began in the forties. His observations of how children construct their knowledge have, over the years, formed the basis for his work. From the first experiments, Piaget developed many theories, describing the stages of a child's cognitive development. Supported by his extensive research work, Piaget established an analysis methodology that set the basis for his learning theory, which is known as Genetic Epistemology [18].

Piaget proposes a hybrid model based on cognitive mechanisms from the species (epistemic being) and from the individual (psychological being), so that the knowledge is not just inherently related to the one individual, nor developed by its own conditioning. In the constructivist theory, knowledge is absorbed by progressive structuring of the experience, evolving by means of an interactive process of construction. According to Piaget's theories, knowledge, at any level, is generated by a radical interaction between the individual and their environment, departing from structures previously existent in the individual.

The development of the constructivist knowledge by Piaget is based on the mechanisms known as assimilation and accommodation, which are part of a process called equilibrium. Considering a maturated biologic stage the individual constructs a broad structure of knowledge through the association of ideas, interaction with objects, and the transmission of information received from the environment. If this structure is not consistent with live experiences, then a constructive error is characterized, and this makes the individual to react to the assimilation. In this case, the individual should begin to reconstruct his hypothesis to a point where the new data may be completely assimilated. This is the mechanism known as accommodation, where "the individual begins to change as a consequence of resistance imposed by the object" [16].

The constructive error applies in an unbalanced situation, which in turn generates a new intellectual action to reach a new equilibrium. According to Piaget, this dynamic process of knowledge construction based on the error is a necessary step towards cognitive development.

4.2 The Constructivist Pedagogic Model

In pedagogic models based on the constructivist theory, the student should construct their own knowledge instead of passively absorbing it in a classroom or by consulting text books. This way of learning demands that the student not only discovers the facts, but also creates mental models from them that may result in knowledge construction. The task of supervising and stimulating the students in achieving this goal is assigned to the teacher, who must be simultaneously aware of the individual cognitive structures of all the students, which in turns makes the method pedagogically more complex than the classical instruction.

Finemman and Bootz [9] outline that under the constructivist theory, the collaborative and support processes of the social negotiation of meanings are especially relevant, insofar as each student has their own perspective. The dialogue exposes the pupil to the multiple perspectives that share the environment, allowing a more precise understanding through the interaction with other classmates. Maziero [16] states that subject-object interaction is the basis of knowledge construction, as this is not in the subject nor in the object, but in the simultaneous interaction between them.

The constructivist model recognizes the benefit achieved when students participate in tasks that allow the active construction of their own knowledge domain. In order to do this the teacher requires a solid grounding in Piagetian fundamentals, as well as an ability to create the proper teaching platforms for the constructivist pedagogic models. One of the main aspects in the construction of this type of environment is the teacher's full perception of the master-pupil dyad that, as in all learning processes, involves a very strong interaction between subject and object.

Table 1 provides some comparisons between conventional and constructivist classrooms [3].

Table 1. Conventional versus Constructivist Classrooms

Conventional	Constructivist
Students fundamentally work alone.	Students fundamentally work in groups.
A high degree of importance is assigned to pre-established discipline curriculum sequence.	The answering for questions raised by the students is high valued.
The academic activity is fundamentally based in text and exercise books.	The activity is mainly based on primary data sources and practical hands-on devices.
The process of learning evaluation is dissociated from the teaching process and is normally accessed by means of tests and exams.	The evaluation is interlinked to the teaching process and it happens through teacher's close monitoring the students' work.

5. A CONSTRUCTIVIST FRAMEWORK

Many OS courses are based upon teacher presentation and explanation of concepts, rather than allowing the students to construct mental knowledge. This model may turn OS lectures into an extremely abstract and boring process. The constructivist theory provides an option for developing pedagogic proposals, possibly leading to better learning outcomes than those obtained with instructive models.

Our current work proposes a constructivist framework to support OS learning, over which pedagogic models can be developed for the discipline at undergraduate level. The main guidelines followed in developing this constructivist model are listed bellow:

- Teaching should be conducted in an individualized manner; the teacher paying close attention to each student's own absorption capability.
- The student-teacher interaction should have a strong emphasis on searching for practical and interesting questions.
- Group work should be proposed as a forum to achieve cooperative learning.

- The teacher should use the OS simulator in conjunction with theoretical lectures, so that complex concepts underlying the subject may be better illustrated.
- The students should use the OS simulator in the classroom and homework as a form of assembling situations difficult to generate in a real system.

In this proposal, work in the classroom should stimulate the student's ability to construct the new knowledge contained in the course syllabus, not forgetting of course, his previous knowledge level and assimilation rate. And it is in this context that the use of an OS simulator establishes a rich student-tool interaction, so that problems that simulate real situations can be presented. The facility to develop and test hypothesis, to create alternative solution proposals and discuss them with the other students and the teacher, makes the simulator an essential tool in the learning process.

The simulator as a constructivist tool emphasizes knowledge construction, as it makes multiple displays of the reality possible, allowing students to test their own hypotheses and learn from their successes and mistakes. In this way, as the students become used to the simulation environment, the software improves their reflexive thinking. It also allows them to control the experiment, enabling a more natural evolution of the complexity as the simulation evolves. Once faced with a specific problem, the students find real support in the simulator that helps them to actively search for a solution, improving their ability to identify, define, and solve problems.

6. AN EXPERIMENTAL MODEL

An experimental pedagogic model, based upon the constructivist framework, has been implemented and used in OS teaching within the Information System undergraduate course at PUC-Rio, Brazil. The model employs the SOsim simulator in the laboratory and includes group work to be developed in both practical classes and at home.

The SOsim simulator was adopted because it is a simple tool to master. It can also illustrate theoretical concepts in an intuitive and easy way, allowing the student to construct their mental model of knowledge. Currently, three laboratory classes using the simulator are reserved in the course syllabus. The first one explains the concepts related to process management. The second works out the various scheduling policies and the last approaches the paged-virtual memory mapping mechanism.

The laboratory classes take place as soon as the corresponding theory is given in the classroom. Students work in pairs in the lab. They work on tasks including practical drills, specific situations to be simulated and analyzed, and some theoretical questions to be answered with the help of the simulator. As the class evolves, the students and the teacher discuss and exchange comments on their simulation results.

During 2003, at the end of each class, we solicited both quantitative and qualitative feedback from the students. The goal was to assess the benefits of the simulator as a valid tool in a constructivist teaching environment. It consisted of seven questions and was submitted to thirty students. With the exception of questions six and seven, which could be answered freeform, the questions were in the form of a Likert scale: "I totally disagree", "I partly disagree", "Don't agree nor disagree", "I partly agree", "I totally agree". The questions are listed below:

1. The simulator makes the understanding of theoretical concepts more satisfying.
2. The simulator helps motivate the student to the subject.
3. The simulator bears an easy and clear interface.
4. The simulator helps the comprehension and absorption of the theoretical concepts introduced.
5. The simulator is adequate for simulating real situations in an operating system.
6. Which simulator features do you find more appropriate to OS learning?
7. What is your overall opinion about the simulator's use?

The results for the first five objective questions can be seen in Table 2. Since there were not any students who answered "Totally disagree", the column was omitted. In summary, the feedback shows that the majority of the students felt that learning with the simulator was enjoyable, it sparked their interest in the subject, yielded a better comprehension of the concepts, and made it possible to configure real situations.

Table 2. Research results

Question	Partially disagree	No opinion	Partially agree	Totally agree
Q1			15,8%	84,2%
Q2		15,8%	42,1%	42,1%
Q3	10,5%	21,1%	52,6%	15,8%
Q4		5,3%	15,8%	78,9%
Q5		5,3%	52,6%	42,1%

For question six, most students answered that the software helped them to better visualize OS concepts and problems, and also that it had narrowed the gap between theory and practice. For question seven, they praised and declared their support for the simulator initiative. Some asked for more lab classes and some suggested improvements in the software. A sample of these answers is listed below.

- "The simulator helps the student move beyond theory."
- "The visualization of concepts introduced in classroom."
- "Very good! I have learned a lot"!
- "More lab classes! The simulator helps a lot in understanding the concepts."
- "Excellent, though other tools should be introduced."
- "The idea is good, but the software should be improved."

7. CONCLUSIONS

The adoption of a constructivist pedagogic model opens excellent perspectives for improvements in Operating Systems' teaching-learning. From an experiment which has been in use at Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, the problems due to a non-linear structured subject, exhibiting a significant gap between theory and practice, were minimized. Apart from that, a qualitative improvement of the whole learning process is expected. This should be most noticeable in intensive

or short period courses, or in those courses for which some of the attendants do not comply with the necessary pre-requisites.

In our experience, the SOsim established itself as an important tool in supporting knowledge construction, by allowing a closer interaction among the students and the object of study. A big pedagogic advantage in using such tool is the construction of a hybrid teaching-learning environment where conventional expositive lectures and simulations may be combined. This way, the cooperative approach is applied in classroom and the constructivist thinking can be introduced to support knowledge construction, making it possible to experiment with the newly introduced theories. The use of the simulator might also contribute to reducing the total time needed for theory presentation and explanation, perhaps extending the practical sessions, and possibly creating new laboratories.

It is important to highlight some problems when using a constructivist model. A common criticism of Piaget's theories resides precisely in the absence of a clear and explicit pedagogy line, as this is not its main purpose. The reasoning behind that is that the theoretical principle supporting the model is epistemological and not pedagogical. Another source of problems is that teachers can be unfamiliar with pedagogic issues. The constructivist instructor should be acquainted with the principles proposed by Piaget and master the pedagogic model proposed.

Future expansion of this work will be targeted towards refining the proposed constructivist model as a form of structuring a systematic pedagogical practice for teaching OS. We aim to establish a pedagogic method that might be used by instructors without previous knowledge of Piaget's theories. Investigations into new potential features for the simulator are expected in due course. Also, in-depth comparative analyses between the effects of traditional and constructivist methods of OS teaching should be tackled.

8. REFERENCES

[1] Anderson, T. E., Christopher, W.A. and Procter, S. J. The Nachos instructional operating system. Available in http://www.cs.washington.edu/homes/tom/nachos/, 1999.

[2] Ben-Ari, M. Constructivism in computer science education. In *Proceedings of the 29th ACM SIGCSE*, 1998.

[3] Brooks, J. G. and Brooks, M. G. Structuring learning around primary concepts: The quest for essence. In search of understanding: The case for constructivist classrooms. Alexandria, VA: Association for Supervision and Curriculum Development, 1993.

[4] Bynum B. and Camp, T. After you, Alfonse: a mutual exclusion toolkit - an introdution to BASI. Available in http://www.mines.edu/fs_home/ tcamp/baci/, 1999.

[5] Chernich, R., Jamieson, B. and Jones, D. RCOS: Yet another teaching operating system. In *Proceedings of the 1st Australian Conference on Computer Science Education*. 1996.

[6] Comer, D. *Operating System Design – The XINU Approach*. Prentice-Hall, 1984.

[7] Downey, A. B. Teaching experimental design in an operating systems class. In *Proceedings of the 30th ACM SIGCSE*, 1999.

[8] Fekete, A. and Greening, A. Designing closed laboratories for a computer science course. In *Proceedings of the 27th ACM SIGCSE*, 1996.

[9] Finemman, E. and Bootz, S. An Introduction to constructivism in Instructional Design. Technology and Teacher Education Annual University of Texas, 1995.

[10] Jones, D. and Newman, A. RCOS.java: a simulated operating system with animations. *Proceedings of the Computer-Based Learning in Science Conference*. Rep. Tcheca, July 2001.

[11] Jones, D. and Newman, A. A Constructivist-based tool for operating systems education. *Proceedings of EdMedia'2002*. Denver, Colorado, June 2002..

[12] Kifer M. and Smolka, S. *OSP: An Environment for Operating Systems (Instructor Version)*. Addison-Wesley, 1991.

[13] Machado F. B. and Maia, L. P. *Arquitetura de Sistemas Operacionais*. 3 ed., LTC, Brazil, 2002.

[14] Maia, L. P. SOsim: A Simulator Supporting Lectures on Operating Systems. M.S. Thesis, IM/NCE, Federal University of Rio de Janeiro, Brazil, 2001.

[15] Maia, L. P. SOsim website. Available in http://www.training.com.br/sosim, 2004.

[16] Maziero, C. A. Reflexões sobre o ensino prático de Sistemas Operacionais. *Anais do X Workshop sobre Educação em Computação (WEI2002)*, XXII Congresso da SBC, 2002.

[17] Pérez-Dávila, A. OS bridge between academia and reality. In *Proceedings of the 26th ACM SIGCSE*, 1995.

[18] Piaget, Jean. *Epistemologia Genética*. Ed. Martins Fontes, 2002.

[19] Ramakrishman, S. and Lancaster, A. M. Operating system projects: linking theory, practice, and use. In *Proceedings of the 24th ACM SIGCSE*, 1993.

[20] Tanenbaum, A. S. and Woodhull, A. S. *Operating Systems: Design and Implementation*, 2 ed., Prentice-Hall, 1997.

[21] Wagner, T. D. and Ressler, E. K. A practical approach to reinforcing concepts in introductory operating systems. In *Proceedings of the 28th ACM SIGCSE*, 1997.