Basoper: a Web-based Open Educational Resource for the Teaching of Scheduling Algorithms for Batch Systems

Henrique Y. Shishido
Institute of Mathematics and Computer Sciences (ICMC)
University of São Paulo (USP)
São Carlos, SP, Brasil
shishido@usp.br

Leonildo J. M. de Azevdo
Institute of Mathematics and Computer Sciences (ICMC)
University of São Paulo (USP)
São Carlos, SP, Brasil
leonildo.azevedo@usp.br

Paulo Sergio L. de Souza
Institute of Mathematics and Computer Sciences (ICMC)
University of São Paulo (USP)
São Carlos, SP, Brasil
pssouza@icmc.usp.br

Júlio Cesar Estrella
Institute of Mathematics and Computer Sciences (ICMC)
University of São Paulo (USP)
São Carlos, SP, Brasil
jcezar@icmc.usp.br

Sarita Mazzini Bruschi
Institute of Mathematics and Computer Sciences (ICMC)
University of São Paulo (USP)
São Carlos, SP, Brasil
sarita@icmc.usp.br

ABSTRACT

Operating Systems are studied in several computing courses. They involve deep and broader topics, such as management of processes, memory, I/O and file systems. Educational tools have been proposed to assist in their transmission. However, no educational tool directed to scheduling process (batch or interactive systems), or capable of converging structural, functional and performance aspects in an integrated way has been found. This paper proposes a novel web-based open educational tool, titled Basoper, to assist in the teaching of scheduling algorithms for batch systems. Basoper is implemented through web technologies and it offers a clean visualization and interactive approach. Students and professors had contact with Basoper and evaluated it under different perspectives such as relevance, adherence to its objectives, interface, among others. The results show the tool is relevant to the teaching of scheduling algorithms of batch systems, attends its objectives in class and extra-class activities, and has an intuitive interface. Our work was rated as excellent by academic, getting the overall average of 9.1 students and 9.2 professors.

CCS Concepts

•**Software and its engineering → Operating systems; Scheduling;**

Keywords

Open Educational Resource; batch systems; scheduling algorithm

TISE '15 December 1-3, 2016, Santiago, Chile
© 2015 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

1. INTRODUCTION

Operating Systems (OS) is a basic subject studied by practically any computing undergraduate student. Although basic, OS cover many topics, e.g., management of process, memory, I/O, file systems and deadlocks [10]. To be taught in a deeper and broader way. Each topic has a vast repertoire of details related to structural, functional and performance aspects. The challenges regarding the understanding of OS grow when students realize distinct solutions must be applied to different systems. The processes and memory managements of OS for large batch systems are not appropriated for small systems, such as those applied to Internet of Things (IoT).

The learning of process management is a hard task, due to its theory's density. Subjects such as process/threads creation, inter-process communication (IPC), message passing and process scheduling impose a high level of abstraction, on students for their understanding of concepts tightly related to each other. Learning Objects (LO) can help this teaching process, mainly when the of specific and important algorithms is simulated [4, 1]. The teaching of scheduling algorithms of batch systems is a good example resources, as LO can make the difference, because they require higher abstraction from students for they understanding of the functionality and impact of such algorithms for the performance as a whole.

Open Educational Resources (OER) represent a forward-step in relation to LO, because they are open and can be used freely, changed and/or redistributed (under specific licenses). Indeed, OER are an already known approach to help professors in showing concepts and implementing aspects of operating systems [7, 13, 5, 11]. Another advantage is the potential to convey theory and enable same time to allow users to improve it.

The literature reports educational tools for scheduling processes teaching [2, 12, 3, 9, 6], however, they leave important gaps: has been (1) no educational tool focused on batch systems scheduling; (2) even considering the scheduling for interactive systems, neither one is an OER indeed, i.e., they do not make available source code, allow changes or the redistribution under specific licenses and

(3) the three related aspects - structural, functional and performance - are not simulated in an integrated way, which would strengthen learning.

We propose a novel web-based OER (titled Basoper) to assist the teaching of scheduling algorithms of batch systems [1]. It does not at to replacing the professor, but complementing the teaching of this topic, enabling the integration of the functional, structural and performance perspectives. It simulates the main data structures involved in the scheduling activity, the way algorithms use data structures to perform the scheduling and the impact such algorithms on the performance of the scheduled application.

Basoper's design is based mainly on studies of the concepts described in "Modern Operating Systems", book of Tanenbaum and Bos (2014) [10]. It is a web-based application that runs on the "client side", without any requirements of databases, system's special privileges or additional packages in the server. Implementation is based on HTML 5, Typescript language, jQuery and Bootstrap, which enable the design of simple interfaces with a clean visualization and interactivity.

Answers from professors and graduate and undergraduate students of computing courses were considered for its evaluation. Such users considered Basoper relevant to the teaching of scheduling of batch systems believed it reached its objectives. Basoper was evaluated as excellent and/or great by most users, regarding perspectives as interface, use in class, activities and extra-class, and installation process.

The main contributions of Basoper are it complements the teaching of batch scheduling algorithms and offers a dynamic, interactive and visually attractive simulation which enables the of tracking a step-by-step execution from a list of jobs scheduled by distinct algorithms. From another (practical) point of view, professors and students can access the OER through a web browser, without installing and configuring any libraries or virtual machines.

This paper is organized as follows: Section 2 provides an overview of concepts and policies about Batch Systems; Section 3 addresses the design and specifications of the Open Educational Resource; the prototype of the model is described in Section 4; Section 5 reports the results focused on users acceptance by students and professors; in Section 6, related works are listed and compared with our OER; finally, Section 7 summarizes the conclusions.

2. TEACHING SCHEDULING FOR BATCH SYSTEMS

The teaching of scheduling for batch systems involves context, motivation, objectives, scheduling algorithms and a performance evaluation resulting from the use of different scheduling algorithms.

Due to space restrictions, details about context, motivation and objectives will not be presented. According to Tanenbaum & Bos (2014) [10], batch computing systems have been widely used in the corporate world, for instance, to process daily banking transactions or issue insurance policies. In such systems, no user waits for a quick response to their requests. Consequently, non-preemptive and preemptive scheduling algorithms, with long-time CPU periods for each process, are acceptable. Three basic scheduling algorithms been addressed: **First-Come First-Served (FCFS)**, **Shortest Job First (SJF)** and **Shortest Remaining Time Next (SRTN)**. **FCFS** is the simplest non-preemptive algorithm, in which a queue of ready processes holds processes to be scheduled into CPU, at the same arrival sequence in the queue. **SJF** is another non-preemptive algorithm. In this algorithm, it is assumed to know the length of jobs in advance. It attends the shortest job quickly, however the duration time of the next CPU use is hard to be identified; **SRTN** is a preemptive version of SJF. It chooses the process of shortest remaining run time. When a new job arrives, its total time is compared to the current remaining time of the process. If the new job requires less time to finish than the current process, the latter one will be suspended and the new job will be started.

The core topic taught should be scheduling algorithms, considering the structures used, their functionality and the impact for the performance of the processes scheduled. Students that aim to learn such a topic must abstract the main aspects and understand how they work and interact.

An OER can transform those theoretical classes into a more attractive and dynamic activity to students by simulating the structures used, the behaviour and comparing the performance and costs of different algorithms. Basoper was designed to reach such aims.

3. DESIGN OF BASOPER

The Basoper is characterized by its technical portability (platform independent) and portability in teaching. At first, it refers to the ability to access the OER from any operating system, without the need for installation. The second is related to the different ways of teaching, both to deliver content and to assist in the setting of problem solving. The extensibility, the design allows adapt new scheduling algorithms, changes in the dynamics of the graphic interface and customization for other languages, as long as the the user to have technical knowledge of web programming.

It defines structural, functional and performance models to simulate the behaviour of scheduling algorithms for batch systems. When viewed together, those models, contribute to the understanding of the scheduling algorithms. Each one is described in the following subsections.

3.1 Structural Model

The structural model represents the data structures used by the scheduling algorithms implemented. The main data structures represented are, PC (Program Counter), TCS (Time for Context Switch), execution steps, and blocked/ready/execution queues (Table 1).

Jobs are configured as a set of arrival time and instructions as P (Processing), and B (Blocked). Arrival time is the moment of arrival of in the batch system. P represents a process instruction spending a CPU logical time unit. B is an I/O instruction occurrence, e.g., a user's prompt. The job instructions structure is illustrated in Table 2, which shows a time line execution at the arrival time 0 (zero), four P (processing) instructions and two B (blocking) I/O instructions. When a job with a set of instructions has been defined, it is added to the Job List (Table 3).

The Ready Queue (Table 4(b)) represents the sequence of execution of each process (process on the left is the next to be executed). This queue is updated according to the scheduling algorithm used. Each process is represented by a

[1]http://lasdpc.icmc.usp.br/~ssc640/pos/basoper/index.html

Job name (process name), its arrival time in the scheduler, a Job Identification (JID) and the position of the Program Counter (PC). Users can track the process execution and its behaviour based on such values. In batch systems, when an I/O instruction occurs, the job is moved from Execution Queue to Blocked Queue (Table 4(a)). After the I/O request has been met, the job is placed in the Ready Queue again for execution.

3.2 Functional Model

Table 1: Summary of data structures for simulations.

Structure	Description
PC	*It a pointer that controls the job execution order, and is exclusive for each job.*
Job instructions	*It is an array that represents Processing (P) and Blocking (B) instructions*
TCS	*It is a logical interval time for switching CPU registers and cache*
Execution Steps	*It is a table that display the execution of the instructions step-by-step according to the scheduling algorithm.*
Execution	*Each job is inserted in this queue, which will provide tasks for the execution steps.*
Ready Queue	*It holds jobs able to be executed. Its content could be rearranged according to the scheduling algorithm*
Blocked Queue	*When a job executes a block instruction, it leaves the execution queue and goes to the blocked queue.*

Table 2: Example of a job. P represents a CPU instruction and B is a blocking request.

Arrival Time	Instructions						
	0	1	2	3	4	...	*N*
0	P	P	B	P	B	...	P

Table 3: Job List.

PID	Arrival time	Instructions List				
Job A	0	P	B	P	P	P
Job B	4	P	P	P	P	P
Job C	2	P	B	P	B	P

Table 4: Example of Blocked and Ready queues.

Blocked Queue		Ready Queue	
Job X JID: 5 PC: 5	Job Y JID: 7 PC: 2	Job A Arrival Time: 0 JID: 2 PC: 1	Job B Arrival Time: 2 JID: 1 PC: 0
(a)		(b)	

Basoper enable the to simulation of functional aspects for the following scheduling algorithms for batch systems: **FCFS**, **SJF** and **SRTN**, showing an step-by-step execution. Besides the tables used by the structural model, additional information is provided for the abstraction of representation CPU usage and I/O requests (Table 5). The first column (JOB) of the table represents the process identification. *P* denotes CPU processing instruction, *B* is a block request (I/O), " – " represents when the CPU is idle, and "↔" represents the time to context switch.

Table 5: Executions Steps.

Job	0	1	2	3	4	5	6	7	8	9	10
A	–	P	B	↔							P
C					P	P	↔				
B								P	P	↔	

3.3 Performance Model

Scheduling algorithms have different aims [8], and the choice by an specific algorithm may favor one type of process rather than other (in terms of performance). Basoper uses the following metrics to evaluate these different properties: (a) throughput: number of processes that are completed per time unit; (b) CPU utilization: percentage of CPU utilization (%); (c) waiting time: sum of the time spent by a process in the ready queue; (d) turnaround time: time taken by a process from its submission to the end of its execution; (e) response time: time elapsed between the submission of the command and obtaining of result.

Three metrics are commonly used: CPU utilization, waiting time and turnaround time. Basoper adopts them to compare the performance of the algorithms. Our design includes the steps spent with CPU use/idle and the context switch must be computed an overview with for the obtaining of metrics. When CPU is either in use, or idle, it is considered a time unit. When a context switch occurs, it is considered 0.1 time unit, i.e. a small fraction spent on the CPU instruction. An exceptional case is when there is only one process executing in the system and a blocked instruction (B) occurs: in this case the B instruction is computed as idle time.

4. IMPLEMENTATION DETAILS

For a clean and interactive interface, Basoper has been developed for web by technologies, such as HTML 5, TypeScript language, jQuery and Bootstrap libraries. Even considering these qualities, it is important to highlight this first version of Basoper did not take all HCI (Human-Computer Interaction) metrics.

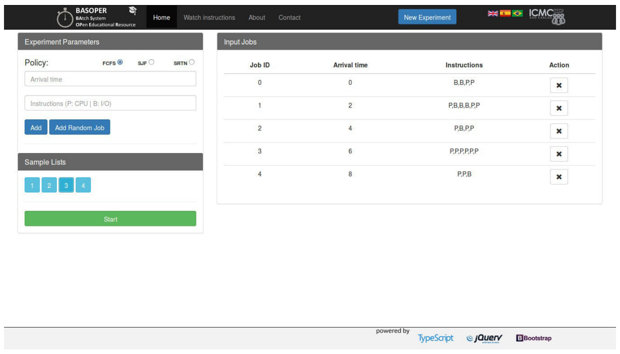
It is a web application that can be either hosted on web server, or directly executed on a desktop machine. The minimum server requirements for hosting our application are a CPU of 500Mhz, 256MB of RAM memory, 1Mbps bandwidth, Linux operational system, Windows or BSD-like, Apache web server, IIS, or compatible. The client must be any web browser that supports JavaScript and HTML5 (Firefox, Chrome, Safari or other).

The prototype was designed in two scenarios, so that the job input actions could be separated from the algorithm execution. Basoper site offers videos with tutorials on its use. Another aspect considered is the language translated to English, Spanish and Portuguese.

In the first scenario, the selection of scheduling algorithm and addition of a job list are performed, as shown in Figure 1. The process can be filled in three ways: a) manually adding the arrival time and instructions lists one per time; b) adding a random job in the list, which means the system generates a random arrival time and instruction sequence, and; c) choosing a static sample list suggested. The latter is recommended to users that aim at generating the same list to compare the performance among scheduling algorithms.

The second scenario contains the structural, functional and performance models described in Section 3 and shown in Figure 2. The structural interface is represented by the step-by-step execution (dark gray), execution queue (green), ready queue (orange), and blocked queue (red). The functional aspect is covered by the permutation among the queues, according to the scheduling algorithm. The performance metrics are computed considering the overall efficiency and CPU utilization, waiting time, and turnaround time of each finished job.

The Step-forward button executes one job instruction per step and plots it for the flow execution in the Table of Execution Steps (dark gray). Another option is to use the Play button for obtaining the behavior of consecutive Step-forward clicks.



Experiment Parameters

Policy: FCFS [✓] SJF SRTN

Arrival time

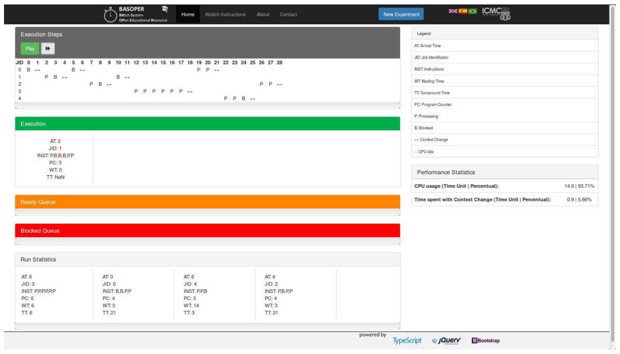
Instructions (P: CPU | B: I/O)

Sample Lists: 1 2 3 4

Input Jobs

Job ID	Arrival time	Instructions	Action
0	0	B,B,P,P	✖
1	2	P,B,B,B,P,P	✖
2	4	P,B,P,P	✖
3	6	P,P,P,P,P,P	✖
4	8	P,P,B	✖

Figure 1: Interface for input job list



Execution Steps

```
JID 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
0   B ↔       B ↔                                      P  P  ↔
1       P B ↔             B  ↔
2               P B ↔                                                     P  P  ↔
3                            P  P  P  P  P  P  ↔
4                                                         P  P  B  ↔
```

Execution

AT: 2
JID: 1
INST: P,B,B,B,P,P
PC: 3
WT: 0
TT: NaN

Ready Queue

Blocked Queue

Run Statistics

AT: 6	AT: 0	AT: 8	AT: 4
JID: 3	JID: 0	JID: 4	JID: 2
INST: P,P,P,P,P,P	INST: B,B,P,P	INST: P,P,B	INST: P,B,P,P
PC: 6	PC: 4	PC: 3	PC: 4
WT: 6	WT: 0	WT: 14	WT: 3
TT: 6	TT: 21	TT: 3	TT: 21

Legend

AT: Arrival Time
JID: Job Identificator
INST: Instructions
WT: Waiting Time
TT: Turnaround Time
PC: Program Counter
P: Processing
B: Blocked
↔: Context Change
-: CPU idle

Performance Statistics

CPU usage (Time Unit | Percentual):	14.9 | 93.71%
Time spent with Context Change (Time Unit | Percentual):	0.9 | 5.66%

Figure 2: Job Execution Interface

Basoper is available for downloading, including codes, from the Open Educational Resource web page[2] at the LASDPC/ ICMC/USP.

[2]http://lasdpc.icmc.usp.br/~ssc640/pos/basoper.zip

5. EVALUATIONS

The Basoper's acceptance was evaluated, regarding its contributions to the teaching of process scheduling in batch systems, under the perspective of students and professors. We make available an electronic form with some specific questions to be answered by professors and students was made available.

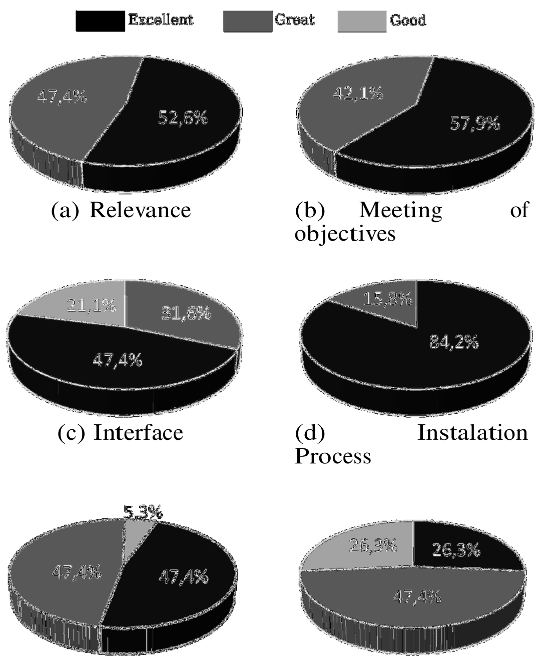
The electronic form contains eight questions, of which six are closed (multiple choice) and two are open. The answers to the closed questions have five levels (Excellent, Great, Good, Fair and Poor), for the classification the user's opinions regarding: teaching relevance (Q1), adherence to the objectives (Q2), evaluation of the GUI (Q3), installation process (Q4), use in the classroom (Q5) and extra class use (Q6). The open questions are devoted to correctness (Q7) and suggestions, approval or criticism to our tool (Q8). Values were assigned for each level Excellent-10; Great-8.5; Good-7; Regular-5; Bad-3. Therefore a weighted average of each closed question could be established.

The form was filled out by professors and students of computing courses. A total of 24 volunteers graduate and undergraduate participated, students of OS and 5 professors from different teaching institutions. Figures 3 and 4 show our preliminary results.

All students (100%) consider the relevance of Basoper to teach batch systems great and excellent. The same result can be observed when evaluating if Basoper reached its objectives as an OER and its installation process. Interface was evaluated as great/excellent by 79.0% of students and its lowest score was good - for 21.1% of the students. The use of Basoper in-class and extra-class activities was evaluated as great/excellent by, respectively, 94.8% and 73.7% of students. According to Table 6, the comments made by students highlighted the positive aspects of Basoper, mainly those ones related to interface, functionality and videos on "how-to-use" this OER. We determined the overall score of 9.1 considering a simple arithmetic average from those values associated with each level.

Under professors' point of view, Basoper was also positively the evaluated (Figure 4 and Table 6). 80% considered the tool excellent or great in relation to relevance (the lowest score for this topic was good (with 20%)). It was also considered great or excellent by all professors (100%) regarding the adherence to the objectives, interface used, installation processes and use in extra-class activities. The use in-class was considered great or excellent by 80% of professors; only 20% considered it good (the lowest score). Therefore, the overall score assigned by professors was 9.2.

Our preliminary results show Basoper has had an excellent acceptance from its users (professors and students of computing courses). The users considered in our evaluation pointed out Basoper is relevant for the teaching of scheduling of batch systems, shows tight adherence to its objectives, has a great user's interface, and a simple installation process and is useful in both classroom and extra-classroom activities. The comments were positive and highlight, among other aspects, through Basoper, the behaviour of different scheduling algorithms. The use of web contributed significantly technology to this acceptance, because this technology offers resources for the development of clean and interactive interfaces. Another important point is Basoper can be deployed on a web server and accessed by any user with an Internet connection, which reduce efforts for installation. This is ongoing work and the results presented here are preliminary. In future work, experiments will be performed evaluating the learning content by the students.

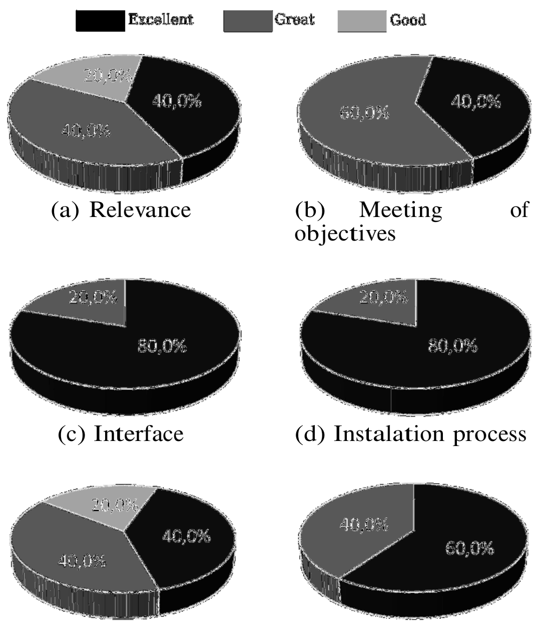


(a) Relevance (b) Meeting of objectives

(c) Interface (d) Instalation Process

(e) Usefulness in the classroom (f) Usefulness in after-school activities

Figure 3: Opinion of the interviewed students. Classification range: excellent, great, good, fair and poor.



(a) Relevance (b) Meeting of objectives

(c) Interface (d) Instalation process

(e) Usefulness in classroom (f) Usefulness in after-school activities

Figure 4: Opinion of the teachers interviewed. Classification range: excellent, great, good, fair and poor.

Table 6: Qualitative opinion from both students and professors. "P" and "S" denote professors and student, respectively.

Users' opinions
P1: "I enjoyed this tool."
P2: "I did not detect errors and its interface is very good."
S5: "The first time I used Basoper I was a little confused about the execution steps; but after watching the video with instructions, the execution sequence was much clearer. I found it a great idea to provide that video."
S10: "The interface is very good. It can illustrate the concepts seen in classroom and enables users to add their own examples, which facilitates personal or classroom use. The inclusion of videos explaining how to use this tool is also a plus ..."
S16: "Excellent OER! It is great to exercise different scheduling algorithms and check their behavior. It would be very interesting to use it in the classroom..."

6. RELATED WORK

This section reports the main simulators available in the literature and used as LO to complement the teaching of scheduling algorithms. We could not find simulators focused on the teaching of the scheduling of batch systems, but only on scheduling in a general sense. The software tools presented are CSCI 152 CPU Scheduling Algorithm Simulator [2], Tran's Scheduling Algorithm Simulator [12], MLFQ Scheduling Algorithm Simulator [3], CPU Scheduling Simulator [9] and MOSS Scheduling Simulator [6].

CSCI 152 is a web-based application to simulate the scheduling of interactive systems available on the Internet. It offers a web interface to show how FCFS, SJF and Round-Robin algorithms work over a unique set of processes. Unfortunately, we can not run this tool to test it, preventing the realization of performing a scheduling simulation.

Tran's Scheduling Simulator is a Java applet that supports a variety of algorithms and creates a custom list of processes. However, each process has a simple process model, by means of it is possible add only a CPU-burst instructions per process. It does not deal with I/O instructions and requires web browser support for Java applets.

MLFQ offers only one scheduling algorithm and more flexibility than the previous ones. Although it enable the setting of a mixed of CPU and I/O instructions, the burst length is fixed to only one value. It has a Gantt chart to illustrate when the process is using the CPU.

CPU Scheduling Simulator provides a Java Desktop program with a graphical animation to cover scheduling algorithms. It uses a realistic process model that can be configured by the user and a graphical tracking to represent what occurs the in scheduler during an execution. It also enables users to test and increase their comprehension of theoretical concepts by making their own scheduling decisions through the GUI of the simulator.

Moss Scheduling Simulator is another program that displays the behavior of scheduling algorithms against a variety mix of process loads. The user can specify the number of processes, the mean and standard deviation for CPU and I/O time for each process, and period of simulation. A statistical summary is provided at the end of simulation. An interesting resource available by this simulator is its flexibility. Users are able writing and testing his own scheduling algorithms to be tested with different process loads.

Analysing these related work, we observe they can be considered learning objects; but not REAs. They are not available as an open-source resource (indeed, they are just available to download). Considering the content taught by Basoper, we can see our tool exceeds the characteristics of the related work.Although CSCI 152 is a web simulator and offers different algorithms, it does not supply an interface for its proper execution. In this sense, Basoper offers a simpler and friendly user interface, able to run on different browsers. We concerned our effort to overcome a limitation of Tran's Scheduling Algorithm Simulator, in order to offer a detailed process model. With this model it is possible to add CPU and I/O bursts, without extra configuration of the web browser to open Java applets as needed by Tran's Simulator.

Although some of the work related allow to perform bursts of CPU and I/O as the fixed length MLFQ, our OER supports a flexible number of instructions per burst. CPU and Moss are two programs with an interesting coverage of details, however they can be used as learning tools only with students of a greater knowledge base.

7. CONCLUSIONS

This paper presented Basoper, a novel OER that helps professors and students to teach and learn process scheduling for batch systems. Basoper explores the integration of three perspectives: functional, structural and performance.

Its main contribution is it makes the teaching of process scheduling for batch systems a more interactive, dynamic and attractive activity. Differently from other related tools, Basoper complements the teaching of this subject through a step-by-step execution of scheduling algorithms and enables their comparison under three distinct points-of-view (functional, structural and performance). Professors can use Basoper to demonstrate the impact of each scheduling algorithm and require students to do extra-class activities, to consolidate the subject taught.

Our results show Basoper has had excellent acceptance from professors and students. According to interviews with users, we note that the Basoper is (has a): (1) relevant for the teaching of process scheduling for batch systems, (2) closely related to its objectives, (3) friendly graphical interface, (4) simple installation process and (5) useful for activities in the classroom and extra-class. Comments made by those users are very positive and highlight, among other aspects, Basoper enables the comparison of the behaviour of different scheduling algorithms. The use of web technology significantly contributed to the acceptance, because it offers resources for the development of clean and interactive interfaces. Basoper can be deployed on a web server and accessed by any user with an Internet connection, which reduces efforts for its installation.

Our next challenges include (i) enabling users to customize some execution parameters, as cost with context switch; (ii) considering the time spent on with (un)load of program into memory; (iii) offering exercise lists for professors to verify if students have understood the topic; (iv) providing charts for comparisons among the different algorithms implemented; and (v) implementing a queue-animation when a job exchange occurs.

8. ACKNOWLEDGMENTS

The authors acknowledge FAPESP, CNPq, CAPES and USP for their support. In addition, acknowledgements to Angela Cristina Giampietro by review of English.

References

[1] P. Bohrer, J. Peterson, M. Elnozahy, R. Rajamony, A. Gheith, R. Rockhold, C. Lefurgy, H. Shafi, T. Nakra, R. Simpson, E. Speight, K. Sudeep, E. Van Hensbergen, and L. Zhang. Mambo: A full system simulator for the powerpc architecture. *SIGMETRICS Perform. Eval. Rev.*, 31(4):8–12, Mar. 2004.

[2] Capricorn. Csci 152 cpu scheduling algorithm simulator, July 2015.

[3] S. Khuri and H.-C. Hsu. Visualizing the cpu scheduler and page replacement algorithms. In *ACM SIGCSE Bulletin*, volume 31, pages 227–231. ACM, 1999.

[4] L. Maia and A. Pacheco. A simulator supporting lectures on operating systems. In *33rd Annual Frontiers in Education, 2003. FIE 2003.*, volume 2, pages F2C_13–F2C_17. IEEE, 2003.

[5] A. Noon, A. Kalakech, and S. Kadry. A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average. *arXiv preprint arXiv:1111.5348*, 2011.

[6] A. Reeder. Moss Scheduling Simulator. www.ontko.com/moss/sched/install_windows.html, 2001. [Online; accessed July, 03,2008].

[7] S. Robbins. Experimentation with bounded buffer synchronization. In *ACM SIGCSE Bulletin*, volume 32, pages 330–334. ACM, 2000.

[8] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. Wiley Publishing, 8th edition, 2008.

[9] S. Suranauwarat. A cpu scheduling algorithm simulator. In *Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE'07. 37th Annual*, pages F2H–19. IEEE, 2007.

[10] A. S. Tanenbaum and H. Bos. *Modern Operating Systems*. Pearson Education Inc, Upper Saddle River, NJ, USA, 4th edition, 2014.

[11] F. Tiosso, S. M. Bruschi, P. S. L. de Souza, and E. F. Barbosa. Amnesia: um objeto de aprendizagem para o ensino de hierarquia de memória. In *Anais do Simpósio Brasileiro de Informática na Educação*, volume 25, pages 80–89, 2014.

[12] Q. T. Tran. Algorithm animation project: Java applet cpu scheduling algorithm animations, July 2015.

[13] R. Urunuela, A. Deplanche, and Y. Trinquet. Storm a simulation tool for real-time multiprocessor scheduling evaluation. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–8. IEEE, 2010.