Development of Simulator for Operating System Processes

Louigi M. Arayata[1], Maria Carmela F. Francisco[2*]
and Ma. Ian P. Delos Trinos[2]
[1]*Department of Information Technology, Cavite State University - Cavite City Campus*
[2*]*College of Science, Technological University of the Philippines, Manila*
[2]*College of Industrial Technology, Technological University of the Philippines, Manila*

ABSTRACT

The study sought to develop a supplementary learning tool that intends to provide an innovative way of understanding the context of a uniprocessor type of operating system. Its core modules have the functionality to introduce the concepts of the operating system's boot process, process state, and interrupts through animated information graphics while the concepts of CPU scheduling schemes, memory management algorithms, and banker's algorithm are presented through simulation. The software tool was designed to run in a stand-alone computer system. The software development methodology involved prototyping which implementation required Visual Basic.Net programming language using Visual Studio Integrated Development Environment (IDE), Camtasia Studio and Camtasia Recorder for animation rendering, Adobe systems for graphics enhancement and SQLyog for the database management system. The application was subjected to different tests to ensure its functionality and accuracy. Further, it was evaluated based on the ISO 25010 quality factors such as functional suitability, reliability, portability, usability, performance efficiency, security, compatibility, and maintainability. It obtained an overall mean of 4.83 equivalent to a descriptive rating of excellent, which implies that the project attained its primary purpose to provide an innovative tool in understanding and learning the context of operating system.

Keywords: *Learning Aid , Operating System, OS Simulator, Simulator*

INTRODUCTION

An operating system (OS) is a system software and framework that oversees resources present on a computer system, thus administers the functions of a computer. OS can be considered as a vital software in a computer system that serves as an intermediary between the computer software and hardware. This serves as the focal point of all the activities and processes inside a computer system (Naghibzadeh, 2011) and as the machine's interface between the user and hardware (Tutorials Point, 2016). It manages computer resources through Central Processing Unit (CPU) allocation schemes, memory management, and other computer processes on the background. This context has made the OS to continuously progress but its foundation for resource management is firmly defined to one of the major subjects for computer-related courses.

Operating System as a course provides introduction to concepts, theories, system components, and computation for scheduling algorithms, process management and other computer resource management paradigms. Further, the Commission on Higher Education (CHED) Memorandum Order No. 25, Series of 2015 prescribes a curriculum map parallel to the subjects being offered to Information Technology and Computer Science – related courses so students can have an additional understanding on how the OS performs computations and other functions in the background (CHED, 2015).

The course is complex in nature for it deals with core concepts of OS like processes and threads, scheduling, memory management, and file systems, input and output device management. It also requires understanding in mathematics, logic and programming which makes it difficult for most students to deal with Operating System as a course. The topics being cited are taught by the instructor through board work and lecture-based instructions. Due to the complexity of the subject, teaching the algorithms implementation and solving it manually could be very hard for both instructor and students. One of the solutions for this dilemma is the implementation of an application with an integration of simulation as a classroom teaching tool. Different subject areas are being incorporated with this technology to aid teachers in delivering effective teaching.

Developing a simulator for OS algorithms in resource management could help educators improve teaching instrument, making the students adopt and learn the subject easier. Nowadays, modern methodologies in the field of education have been continuously on track to provide betterment to students and to aid the educators. Digital tools, like simulators in particular, have been incorporated to enhance the teaching strategies of modern educators in a classroom setup. There are existing software applications which provide simulation for operating system's scheduling algorithms. As such, developed simulator like OS Sim simulator has the capability to present the context and ideas behind OS process scheduling algorithm and memory management and disk scheduling (Macia, 2015). Likewise, a CPU scheduling simulator embedded with preemptive and non-preemptive CPU scheduling (Pinoy Computer Engineer, 2017), and simulator program for page replacement algorithm (Solanki, 2015). However, the cited simulator programs only deliver individual types of scheduling algorithm and static system interfacing. In addition, these simulators do not offer a comparative analysis on the performance of an algorithm. Existing software tools do not include functions to simulate memory allocation algorithms, memory swapping, and deadlock simulation using banker's algorithm for single and multiple resources, which make these simulators inadequate. Also, some concepts of operating system can be more precisely described through the use of animation, which existing simulator models lacked.

The development of the application focusing on the creation of a simulation software for OS context and other computer resource management can stimulate appreciation on the subject. This sought to effectively learn and teach OS subject. The study can be beneficial and constructive for both teachers and students since it can provide an innovative way for studying and understanding the context of OS that could lead to stimulating, exciting and interactive classroom discussions.

OBJECTIVES

In general, this research project sought to develop a Simulator for Operating System Processes that will serve as a supplementary learning aide for teaching and studying operating system processes.

Specifically, it was undertaken to:

1. Design a software tool with the following features:
   a. Animated representation of the following:
      a.1. Boot process;
      a.2. Process state diagram; and
      a.3. Interrupts
   b. Simulator function for:
      b.1. CPU scheduling algorithms for preemptive, non-preemptive algorithms, and multilevel queue;
      b.2. Memory management that includes page replacement algorithms, memory allocation algorithms and memory swapping; and
      b.3. Deadlock through Banker's algorithm
   c. Comparative analysis for CPU scheduling algorithms
   d. Comparative analysis for page replacement algorithms
2. Create the software using Visual Basic.Net as programming language through Visual Studio Integrated Development Environment, Camtasia Studio and Camtasia Recorder for animation rendering, Adobe systems for graphics enhancement and SQLyog as the database management system;

3. Test and improve the software tool in terms of functionality and accuracy; and
4. Evaluate the performance of the software tool according to the criteria of ISO 25010 for quality software.

Theoretical Framework

E-Learning, a future tool for globalization. Electronic learning (E-Learning) is an innovative tool that facilitates learning through the provision of interactive courses and methods of teaching stated that since technology continues to develop, e-learning is becoming more available to any platform. This gives opportunities in a vast multimedia training paradigm (Laskaris, 2014).

The creation of a software tool that will serve as a supplementary resource for learning the concepts of operating system is the consideration for development of an e-learning software tool. More so, the said tool is embedded with different functions that present and simulate the major topics of the subject.

Computer-aided instruction. A Computer – Aided Instruction (CAI) is a tool that utilizes a computer system as the tool assist the learners with their learning topics. On the part of the lecturer, the said tool can improve the teaching styles and methodologies. CAI learning material was embedded with tutorials, simulation, and problem-solving approaches to introduce lessons to improve students' knowledge (Mann, 2009).

The development of a computer-aided instruction sought to provide an innovative tool that will provide learners and teachers a software that can be used for discoursing operating system course subject. Through the use of the software tool, the teachers can have different sets of examples dealing with the implementation of the algorithms that the OS system uses.

Animation. Objects that are said to be metaphorically spatio-visual can most likely be best portrayed through graphics. As such, the use of moving graphics conveyed with the content and format of the concepts of something that a person wants to learn and understand tends to facilitate and improve comprehension, learning, retention, and inference (Tversky, Morrison, & Betrancourt, 2002).

Likewise, an animated instruction must be properly organized and integrated with thematically relevant information, incorporating knowledge structures for learners. Moreover, it involved a dynamic characteristic involving visuospatial, audio and graphics. This, in turn, promotes fundamental advantage of animations over still images as part of learning process enhancement (Lowe, 2003).

The implementation of animation to the software tool involves animated graphics. This conveys more information and promotes interactivity, making it more effective than static graphics in terms of representing sophisticated systems.

Operating System. An Operating System (OS) is a system software comprised of complex instructions to computer hardware. This forms layer of programming code in which computer functions and processes are built. Further, the system serves as an intermediary between hardware and software. OS is also described as the most vital part of computer system for it deals with all the assets and resources. This includes computer hardware such as memory, Central Processing Unit (CPU), and storage capacity and software applications. OS is developed by programmers in lieu to balance the differing needs of end users and system resources that contend for priority, CPU processing, and system services and applications running on the background (McHoes & Ballew, 2011).

The same idea was formulated by the researcher. The context of simulating the CPU scheduling algorithms focuses on computing the output averages in an automated manner. Also, the comparison between the algorithms can be done through running multiple CPU scheduling schemes simultaneously. This is to determine and compare the performance efficiency of one scheme to another.

OS Simulator. An OS Simulator is an educational aid for studying operating system concepts. Moreover, the purpose of OS Sim is to present the context and ideas behind OS in a graphical simulation, thus providing a tool for computer science students learn the subject [Macia].

The reference application includes simulation modules for process scheduling algorithm and memory management of the same functionalities to the development of the software tool. In addition, other OS topics such as boot process, process state and interrupt, page replacement algorithm, memory swapping and deadlock are also integrated to increase the complexity to the software tool, to which are the major topics that are not part of the reference simulator.

Similarly, a CPU scheduling simulator embedded with First Come First Serve, Shortest Remaining First (Preemptive), Shortest Process First (Non-preemptive), Round Robin, Priority (Preemptive) and Priority (Non-preemptive). The application also has the capability to generate random datasets. The simulator computes for limited outputs such as average waiting time (AWT) and turnaround time (ATAT) [Pinoy computer engr].

Through this reference, the research project was designed with a more complex computation criteria such as averages for response time (RT), waiting time (WT), turnaround time (TAT), completion time (CT), and throughput (TP). Simulation function for multilevel queue scheduling is also added. More so, comparative analysis for multiple CPU scheduling algorithms, dataset management, printing of simulation data and system variables management are integrated to make the simulator more efficient.

Moreover, the simulator for page replacement algorithm includes functions to simulate the implementation logic of First In, First Out (FIFO), Optimal and Least Recently Used (LRU) algorithms. The simulator can accept manual user input and can also generate a random dataset. After the simulation, the application can generate the complete page table, total number of page hits, and page faults [Solanki].

The simulator became the basis for the design of the page replacement algorithm module of the overall system for the research project was integrated with additional functions such as simulating multiple page replacement algorithms at the same time to determine the efficiency ranking of the algorithms. Additional output computations such as hit ratio and fault ratio are also computed and can be printed. Further, the dataset used in simulation can be saved. The criteria for inputting necessary data can be modified and adjusted through the integration of system variables.

METHODS

The Simulator for Operating System Processes is intended to help the educators teach the fundamentals and context of Operating System (OS). The design of application focused on the inclusion of animated representation of the idea behind boot process, process states and interrupt. Simulation for OS process scheduling algorithms, memory management algorithms, memory allocation algorithms and memory swapping are also integrated as part of the design. The software tool design also included the management of system variables and a help module to provide an overview on system's usage and navigation. Figure 1 presents the functional diagram for the module definition of the Simulator for Operating System Processes.

Figure 1 shows the functional diagram of the system which includes the main form and the main modules as follows: (1) Boot Process, (2) Process State, (3) Scheduling Algorithm, (4) Memory Management, (5) Deadlock, and (6) Interrupt Module and two (2) complementary functions such (1) System Variables, and (2) Help Function will be displayed.

The system requires necessary inputs for simulator functions to execute. On CPU Scheduling, arrival time, burst time, priority level

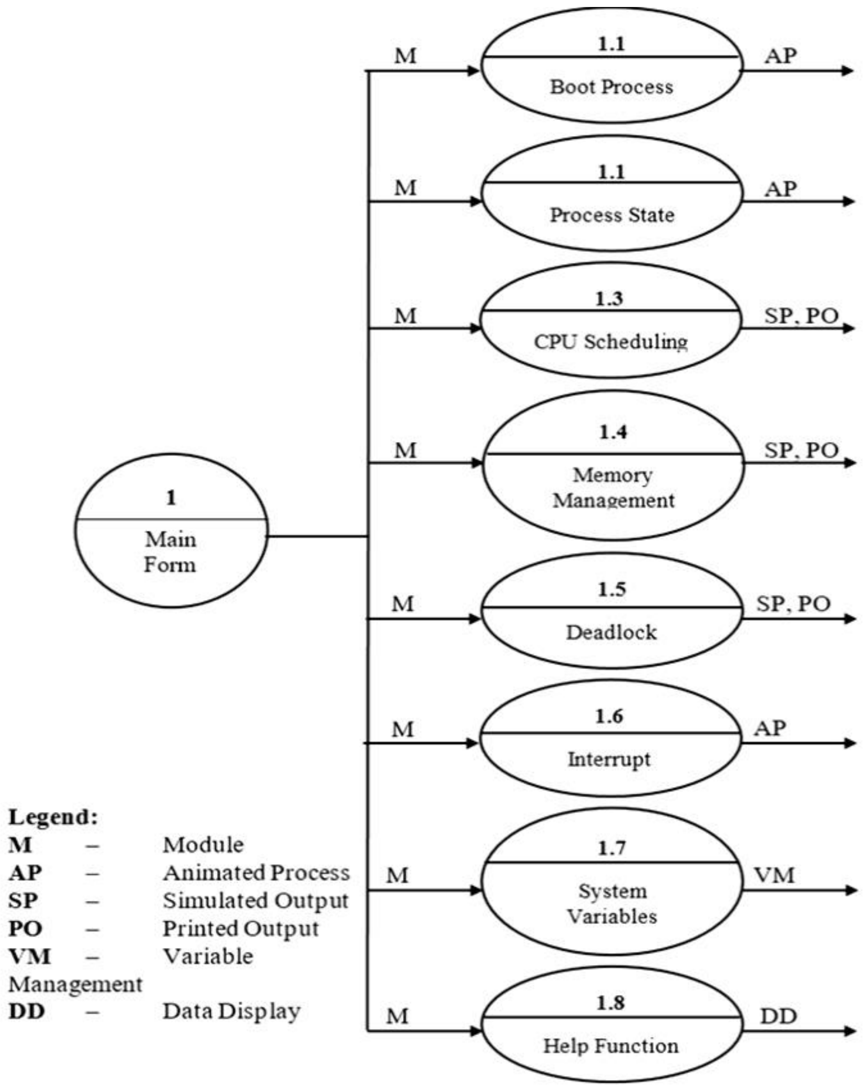


Figure 1. Functional diagram of Simulator for Operating System Processes

(for priority-based scheduling), time quantum (for round robin scheduling), and priority queue (for multilevel queue scheduling) are the inputs. On memory management, inputs to simulate page replacement algorithms are the reference string and the number of page frames. Further, the memory size, number of memory partitions, job ID, job size and execution time are the inputs needed to simulate the memory allocation algorithms. On memory swapping, size of the main and secondary memory are needed to simulate the swapping process. Deadlock simulation process, on the other hand requires dataset containing the number of resources, resource instance values, number of processes, and process allocation and maximum values.

All of the simulator functions are programmed in a way that it can accept range of inputs based from system variables, and thus the simulation can be processed "step-by-step" or "skip all steps" to show the final output such as Gantt chart and computation table for CPU scheduling, page replacement table for page replacement algorithm, process logs for memory allocation and memory swapping, and process safe sequence, and state for deadlock simulation. Further, the outputted simulation results of the said processes can be printed, while the animation-based modules such as Boot Process, Process State and Interrupt, the topics are presented through an animated information graphics.

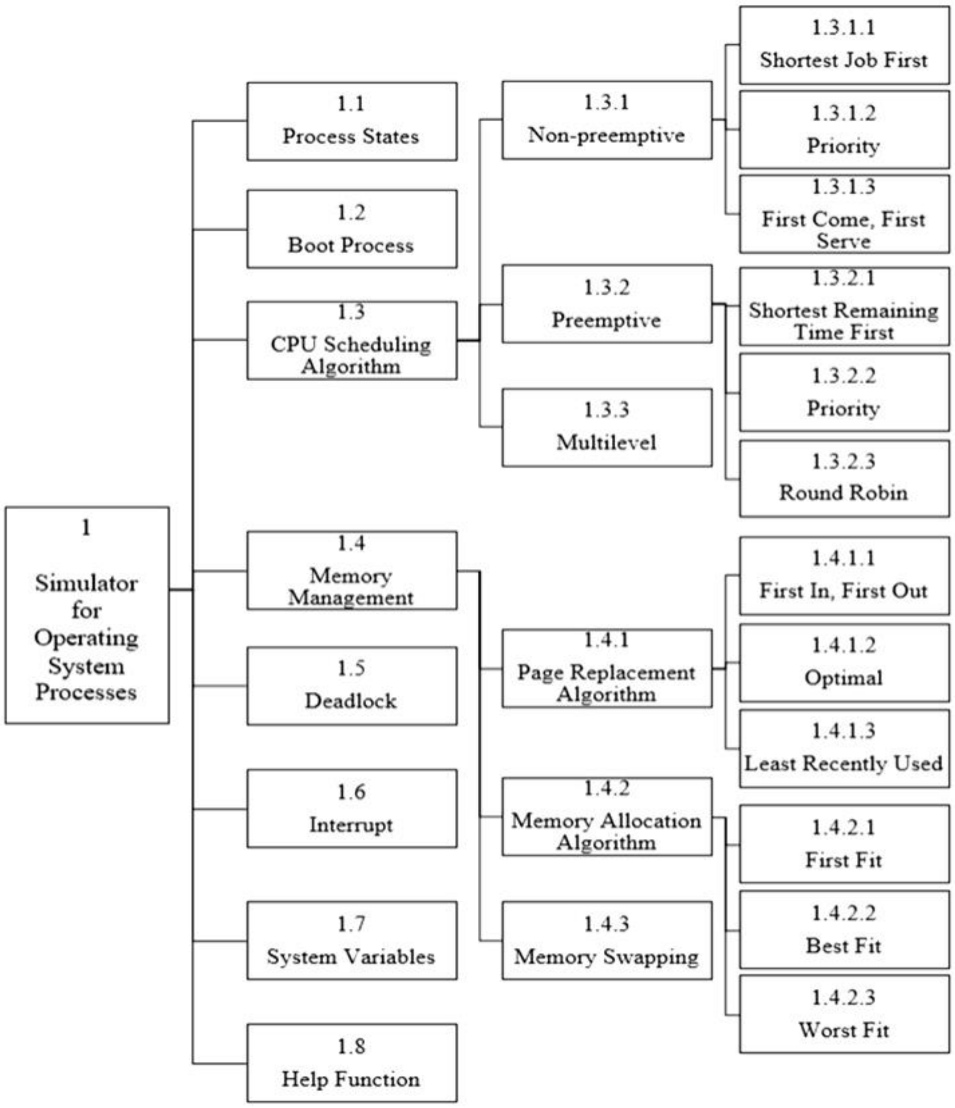


Figure 2. Functional decomposition diagram of Simulator for Operating System Processes

In addition, the Functional Decomposition Diagram was also used to design the system. Figure 2 shows the hierarchy of the functional decomposition diagram of the study. The process starts on the module selection in the main form. The Boot Process module presents an animation with regards to stages and procedures of the operating system's booting process. The Process State module provides an animated graphical representation of process state diagram. The CPU Scheduling Algorithm module is integrated with functions to simulate the implementation logic of preemptive, non-preemptive and multilevel queue scheduling. The preemptive scheduling includes the shortest remaining time first (SRTF), priority, round robin (RR), while non-preemptive algorithms consist of shortest job first (SJF), first come first serve (FCFS) and priority, and multilevel queue scheduling. The cited CPU scheduling algorithms are embedded with a simulator function wherein the needed criteria for each algorithm, such as number of processes, burst time and arrival time should be inputted in order for a specific schema to be executed. Further, the simulation of the algorithms for preemptive and non-preemptive scheduling includes comparative analysis option to run and rank the efficiency of selected algorithms based on the inputted dataset. The output of the simulation process includes waiting time, response time, turnaround time, completion time, and throughput will be computed and a process Gantt chart that can be printed, in which the efficiency of the algorithms can be ranked based upon the dataset being used.

The Memory Management module contains three submodules such as page replacement algorithms, memory allocation algorithms, and memory swapping. The page replacement algorithm is integrated with simulator function to simulate the implementation of the optimal page replacement algorithm, first-in, first-out (FIFO) page replacement algorithm, and least recently used (LRU) page replacement algorithm. The function can be executed by providing the number of page frames and reference string input data. The output of the simulation process includes the page replacement table, the number of page hits, and the number of page faults with their averages that can be printed. This also has a comparative analysis function to determine the efficiency of each algorithm being used, presented through chart that shows the hit and fault percentages. Further, the memory allocation algorithm involves the replication of the processes for first fit algorithm, best fit algorithm and worst fit algorithm. This simulator needs specifications for memory size, partition data, and job data to execute. It has a real-time execution based on processes' execution time, and the simulator events are presented through logs. Memory swapping, on the other hand, includes a real-time simulation of the logical implementation of swapping of a job data in and out of the memory. The simulation process requires specifications for main and secondary memory sizes, job size, and CPU time. The execution of process swapping is seen through the event logs.

The Deadlock module was designed with the implementation of banker's algorithm with single resource and multiple resources. The simulation of the algorithms is done by providing the specifications and dataset for the resources, number of processes, allocation values, and maximum values. The simulator then generates a Gantt chart that contains the safe sequence and the state of the algorithm.

The Interrupt module, on the other hand, presents the context and ideas behind the interrupt process. This is shown and played through the integration of an animated information graphics on the form.

Also, the software tool has system variables that can be managed and updated. The variables are the bases for the minimum and maximum values for the user input and for the generation of random dataset. Lastly, a help tool is integrated for viewing the necessary topic data to aid the user with regards to the system's usage.

The simulator was designed and developed through the implementation of prototyping development methodology. Upon building the prototype, the design of the interface for each module and function based on system requirements and preparation of storyboard were made. Then, pseudo coding and initial prototype with core modules are defined. After the development of the first prototype, the functionalities of the simulator and synchronization of the animations based on the animation script and object transitions were tested. After the said phase, validation of the events performed during the functionality testing against the standard requirements being set. Also, design specifications and functionalities were refined and improved whenever possible. The researcher also consulted instructors of Operating System course subject with the output results generated by the prototype, and refined the system coding whenever inconsistencies and errors in computations are found.

Further, it was developed using Visual Basic.Net as programming language through Visual Studio Integrated Development Environment, Camtasia Studio and Camtasia Recorder for animation rendering, Adobe systems for graphics enhancement and SQLyog as the database management system to encapsulate default examples for the simulation of the system's scope.

RESULTS AND DISCUSSION

The system underwent a series of tests to guarantee the software quality in terms of functionality and accuracy. Functionality test was applied to evaluate the performance and efficiency of the software tool in connection to its requirements and specifications to assure that it has met and satisfied the requirements set. Table 1 discusses the percentage of all cases executed, passed, failed and not executed, if any.

Two cycles for testing the functionality of the software tool were conducted. In Cycle 1, all the test cases were executed. However, results showed that only 84.36 percent of the test cases passed the procedure, and 15.94 percent failed to satisfy its expected functionality, implying that the overall functionality of the system is not yet fully achieved. After evaluating and validating the results, the system was refined. Bugs and errors were corrected, minor adjustments to the objects were also done to make the application more interactive and suitably functional.

After applying the resolutions and revisions, the test cases which failed the test were executed again in Cycle 2 to validate if all the issues were resolved. Results showed that the eight test

Table 1. Test case execution summary

Test Execution	Expected Result	Actual Result	
		Cycle 1	Cycle 2
Test Cases Executed	100%	100 %	100 %
Results of Test Cases			
Passed	100%	84.36 %	100 %
Failed	0%	15.94 %	0 %
Test Case Not Executed	0	0 %	0 %

cases passed all the tests when executed in accordance with their functions. This implies that the software tool's functionality testing is successful.

The system was evaluated by ten (10) Information Technology instructors and 30 selected students taking computer-related courses. The study used ISO 25010 as the evaluation instrument. On the basis of the tests and evaluation conducted on the software tool, the project was found to be successfully developed in accordance with the defined requirements of the overall system functionality.

The functional suitability performance of the software tool obtained an excellent rating with a mean of 4.92, in which was the highest rating based on the evaluation results. It denotes that all functionalities were implemented based on the specified requirements, to the degree that the system produces correct and suitable outputs.

Further, the application got a mean of 4.83 on reliability with an equivalent qualitative interpretation of excellent, implying that the application is dependable under normal conditions.

The Portability criterion, on the other hand, obtained a weighted mean of 4.80 with a qualitative interpretation of excellent, indicating that the system can run and be supported on

Table 2. Summary of evaluation results

Criteria	Mean	Qualitative Interpretation
Functional Suitability	4.92	Excellent
Reliability	4.83	Excellent
Portability	4.81	Excellent
Usability	4.86	Excellent
Performance Efficiency	4.78	Excellent
Security	4.67	Excellent
Compatibility	4.84	Excellent
Maintainability	4.90	Excellent
Grand Mean	**4.83**	**Excellent**

Range of values: 4.20 – 5.00 = Excellent; 3.40 – 4.19 = Very Good; 2.60 – 3.39 = Good; 1.80 – 2.59 = Fair; 1.00 – 1.79 = Poor

different platform versions and can be easily installed.

More so, the usability criterion is marked as excellent, with a weighted mean of 4.86. Result showed that the software tool’s components can be used by specified users to achieve defined objectives with optimum effectiveness, efficiency, and satisfaction in a suggested context of use.

The Performance efficiency criterion got a computed mean of 4.78 that translates to excellent verbal interpretation. This denotes that the response time, processing time and throughput rates of the system is kept to a minimum.

Likewise, the criterion Security garnered a weighted mean of 4.67. The result translates to an excellent qualitative interpretation. This implies that the application can ensure the data are accessible only to the authorized system users.

The Compatibility criterion, likewise, obtained a weighted mean of 4.84, with an excellent qualitative interpretation. It shows that the software tool can perform its required function efficiently, thus the application components are integrated with one another without problems or issues.

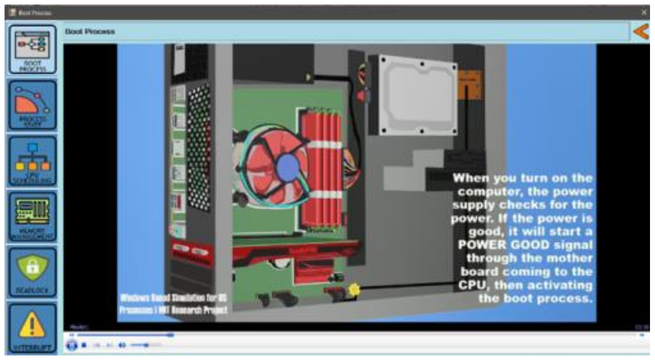
Lastly, the Maintainability criterion also got a high mean rating of 4.90, parallel to its qualitative interpretation of excellent. The result denotes that the said criterion met the prescribed standard software requirements, and specifications to be maintainable and flexible enough to cater the changing needs of the user.

On the basis of the tests and evaluation conducted on the software tool, the study is found to be successfully developed in accordance to the defined requirements of the overall system functionality. Results from accuracy and functionality tests proved that the system performs in conformance to its specifications, thus gaining an average mean of 4.83 with a verbal interpretation of excellent.

Below are the sample screenshots from the system, supporting that it successfully met all the specified requirements and specifications.

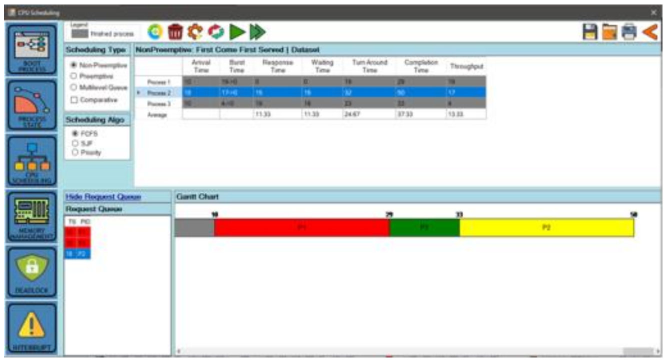
Figure 3 exhibits the information graphics of the Boot Process animation. An animated video pertaining to the context of Operating System Boot Process can be played in this window. It is integrated with a media player that has controls to play, pause, stop and adjust the playback volume.

Figure 4 illustrates the CPU Scheduling Algorithm Window. This contains the scheduling type selection for Preemptive, Non-preemptive, Multilevel Queue and the option to simulate a comparative analysis among algorithms. The window also has a table for inputting necessary



When you turn on the computer, the power supply checks for the power. If the power is good, it will start a POWER GOOD signal through the mother board coming to the CPU, then activating the boot process.

Figure 3. Boot Process



Scheduling Type: [✓ Non-Preemptive] Preemptive Multilevel Queue Comparative

Scheduling Algo: [✓ FCFS] SJF Priority

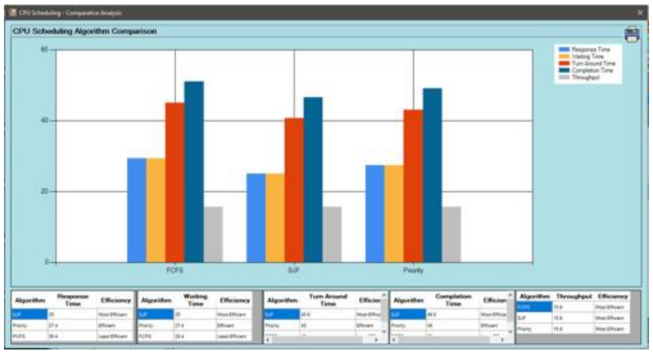
NonPreemptive: First Come First Served | Dataset

	Arrival Time	Burst Time	Response Time	Waiting Time	Turn Around Time	Completion Time	Throughput
Process 1	10	19/0	0	0	19	29	19
Process 2	18	17/0	15	15	32	50	17
Process 3	10	4/0	19	19	23	33	4
Average			11.33	11.33	24.67	37.33	13.33

Figure 4. CPU Scheduling Algorith

dataset. Buttons for removing of an inputted process, generating a random dataset, resetting the simulation process, printing of the simulation outputs, saving the dataset, and loading the saved datasets are also integrated in the interface. Moreover, the window contains functions to simulate the algorithm in step by step procedure to skip all steps, and are presented through Gantt chart and in populated data table.

Figure 5 denotes the graphical representation of the comparative analysis for multiple CPU Scheduling algorithms with the same dataset. The ranking for the efficiency of selected algorithms through chart and tables are also displayed on the said form. The window also includes the functionality to print the efficiency ranking.



Algorithm	Response Time	Efficiency
SJF	25	Most Efficient
Priority	27.4	Efficient
FCFS	29.4	Least Efficient

Algorithm	Waiting Time	Efficiency
SJF	25	Most Efficient
Priority	27.4	Efficient
FCFS	29.4	Least Efficient

Algorithm	Turn Around Time	Efficien
SJF	40.6	Most Efficie
Priority	43	Efficient

Algorithm	Completion Time	Efficien
SJF	46.6	Most Efficie
Priority	49	Efficient

Algorithm	Throughput	Efficiency
FCFS	15.6	Most Efficient
SJF	15.6	Most Efficient
Priority	15.6	Most Efficient

Figure 5. CPU Scheduling Algorith Comparative Analysis

CONCLUSIONS

The software tool was developed to provide a supplementary learning tool that will aid the user in understanding or learning the course topics in a more innovative method. The software tool is comprised of modules to present the context of boot process, process state, and interrupt through animated information graphics. It also includes the simulation of the different processes performed by the operating system in resource management such as CPU scheduling schema, algorithms implied on memory management, and Banker's algorithm for single and multiple resources. In addition to its simulator function, the system has the capability to run multiple algorithms at the same time for comparative performance analysis. The software tool can support runtime on Windows 10 operating system.

Throughout the development cycle, the specified methodologies and procedures were considered and followed. The logical design of the software tool, in particular to the input-process-output model, was conceptualized through the use of functional diagrams. The functional decomposition diagram was further used to define and organize the system's components and modules. Pertaining to the simulator's physical design, storyboarding for each window was effective in visualizing the graphical user interface design. Further, the software tool was primarily developed through the framework of a prototyping model where its iterative approach was found to be an effective way in monitoring the improvement of the software development. During the development stage, series of tests on functionality using the predefined test cases played significantly in capturing and fixing the issues based on the observed severity and priority levels of the failed test cases. The evaluation system which includes the ISO 25010 evaluation instrument, weighted mean, Likert scale and purposive sampling technique proved to be appropriate in assessing the performance of the software tool.

On the basis of the tests and evaluation conducted on the software tool, the study is found to be successfully developed in accordance to the defined requirements of the overall system functionality. Results from accuracy and functionality tests proved that the system performs in conformance to its specifications. Further, the application was evaluated to be excellent in terms of verbal interpretation, with its overall weighted mean of

4.83. This result denotes that the software tool has attained its objectives and intended purpose to provide a supplementary learning tool to aid instructors teach operating system course subject, and for the students to grasp the lessons through a computerized learning management system.

RECOMMENDATIONS

The following recommendations are made to further enhance the software tool: (1) assessment and examination for users and learners, in a network-based environment be included; (2) the algorithm implementation for multiprocessor be considered; (3) I/O burst on CPU scheduling algorithms be included; and (4) compatibility for dynamic memory partition size and memory addressing in memory allocation be included.

ACKNOWLEDGMENT

The success and completion of this research project would not be possible without the valuable help and support of numerous people. Likewise, the authors would like to express sincere indebtedness and gratitude to the following for extending their help and the best of their knowledge throughout the conduct of the study:

Technological University of the Philippines – College of Industrial Technology, for colluding to pursue the conduct of this study;

Mr. Dionard Vale Cruz, Mr. Arjon C. Veluz, for their valuable contributions for test proofing and validating the accurateness of the computation outputs of the research project. Their participation in preparing and solving volume of test cases helped the authors attain the outstanding evaluation results;

Respondents and evaluators who tested and evaluated the efficiency of the project;

Prof. Teresita R. Tolentino, for her selfless contributions and valuable pieces of advice during the conduct of the study;

Prof. King David J. Agreda, for sharing his profound ideas about Operating System which served as the base concept of the study;

Mr. Dionard Vale Cruz, Mr. Arjon C. Veluz and Ms. Jocelyn Ramos, colleagues, for their valuable contributions for test proofing and validating the accurateness of the computation outputs of the research project. Their participation in preparing and solving volume of test cases helped the authors attain the outstanding evaluation results; and lastly

Deepest gratitude and appreciation is given to God, for His showers and blessings of knowledge, talents, and for making the authors surrounded with the people who helped them accomplish and finish this graduate study.

The Authors

REFERENCES

Commission on Higher Education. (2015). *CHED Memorandum Order (CMO) No. 25, Series of 2015.* Retrieved from Commision on Higher Education.

Laskaris, J. (2014). *E-Learning, A Future Tool for Globalization*. Retrieved from https://www.talentlms.com/blog/where-do-i-start-creating-first-e-learning-course/

Lowe, R.K (2003). Animation and Learning: Selective Processing of Information in Dynamic Graphics. *Learning and Instruction*. Retrieved from https://www.sciencedirect.com/science/article/pii/S095947520200018X?via%3Dihub

Macia, A. (2015). OS Sim (OS Concepts Simulator). Retrieved from https://sourceforge.net/projects/oscsimulator/

Mann, B.L. (2009). Computer-Aided Instruction. doi:10.1002/9780470050118.ecse935.

McHoes, A.M. & Ballew, J. (2011). *OS.* McGraw -Hill Education.

Naghibzadeh, M. (2011). *Operating System: Concepts and Techniques.* iUniverse.

Pinoy Computer Engineer. (2017). CPU Scheduling (Simulator Application). Retrieved from https://play.google.com/store/apps/details?id=com.pinoycomputerengineer.cpuscheduling&hl=en

Tutorials Point. (2016). *Operating System: Funadamenal OS Concepts*. Retrieved from https://www.tutorialspoint.com/operating_system/

Tversky, B., Morrison, J.B., & Betrancourt, M. (2002). Animation: Can it Facilitate? *International Journal of Human-Computer Studies*, 57(4), 247-262. doi:https://doi.org/10.1006/ijhc.2002.1017