3 - Software Design Document

Team:
AJ Omartian - Project Manager
Danya Alrawi and Ikbel Amri - Architect
Elisabeth Fernandez - Business Analyst
Alex Hernandez - Lead Software Developer

Introduction:

This document serves as design guideline and a user manual to outline and explain design, features, and restrictions. This document will also outline the development team's plan, design, and strategy, in addition to providing high level and low level requirements. It will also serve as a manual for the user to understand what the product provides, intended platform, and restrictions of functionality.

Operating systems are a layer that allows for hardware to be utilised, and for programmers to build their applications for the general user. The most important task that operating systems take care of is organizing processes (jobs that need to be executed) and dedicating the necessary hardware resources for each process. This project will show a simulation of how multiple processes are scheduled upon their arrival. Multiple algorithms will be used to show performance difference, and pros and cons for each scheduling algorithm.

System Overview:

The development platform for this project will be Java. A simulation of how multiple processes may be scheduled according to different algorithms will be written to explain a fundamental function of operating systems. The user will be able to provide a list of processes with multiple running times and processor request times through a text file. The output will show the order at which the processes execute, the waiting queue of the processes to be run next, hardware usage per process if that information is provided, and total time taken for the processes to complete.

Initial Design Considerations:

At this stage we are considering implementing the First Come First Served as well as the Round Robin algorithms. We Through our implementation, we would like to be able to shed light into the following issues:
-Can performance difference be easily shown between these algorithms?
-What is the form of interaction with the user?

-What information are we expecting to be provided about each process?
-How are they stored?

-For the first phase of the project, we have implemented a simple method to randomly generate a number of processes, assign each a process ID, process arrival time, and the CPU cycles needed for the process to complete. In order to implement a first-come first-served scheduling algorithm, we sort the processes according to arrival time, then print out the order in which the processes were executed along with every PCB. For future implementation, we will add more user interaction in order to be able to modify the process data, insert new processes, customize the virtual CPU that the processes are run on (for example, multiple CPU's would allow for parallel processing).

-For the next phase of the project, we are looking at comparing FCFS algorithm with round-robin scheduling. The point is to show that round-robin is more fair to small jobs since every process is granted a constant number of clock cycles and is then pre-empted.

-For the third phase, the two algorithms have been implemented and are able to run on user defined input or randomly generated data.

Dependencies and Constraints:

Since this is a simulation and not software that directly communicates with the hardware, the machine that will run this program has to have the platform installed (be able to run a Java application).

-For the first phase of the project, we have example code as python script and a java file. In order to run the project in that phase, the user has to compile then execute the code through the terminal. The machine needs to have jdk installed, the code can be compiled using javac for the java code.

-For the second phase, we are converting all code to Java, putting together the data structures and the algorithm implementations.

-For phase 3, we have separated the code into several files such that modifying one aspect does not interfere with the rest. No new dependencies are added that the user needs to have.

Goals and Guidelines:

The purpose of this simulation is to show how different scheduling algorithms work, what processes are completed first according to different algorithms, and to show how multiple variable affect the total running time, total number of processes completed per unit time, and hardware utilisation. These variables may be changed such as allowing the user to decide the machine capacity (for example, the user may enter the number of processors that this virtual machine has, which would affect the scheduling of the processes). Another variable would be adding more information about each process; for instance, what if each process had a value that

determines priority of the process? How does that change the scheduling order and the overall run time?

-In the current progress of the project, we need to add multiple features to achieve the goal of a full simulation of the scheduling process. More specifically, we need to add the user interaction part, allow for dynamic input rather than a static randomly generated list.

Another important guideline is to validate the data used for each process. For example, no negative values for the number of processing cycles. These have been shown in the project plan for phase 1.

-For phase 2, we have worked on validating input, the next step is to put everything together so that the final product works without problems.

-For phase 3, we have achieved the goals outlined in the system overview. The program is capable of simulating two scheduling algorithms, reading and writing to file, take new input from the user in addition to modifying existing data, and take into account memory usage of the system.

Detailed architectural design:

This program allows the user to either provide data for processes directly through the command prompt, or provide them through a text file. When using a text file, the user has the ability to list all the processes currently written to the text file, modify them, and run either of the two scheduling algorithms implemented.

The program also takes into account a fixed system memory and does not allow the addition of processes that exceed the system memory limit. In addition, the system is designed such that we have a single core processor, and only one process can be running at a time, while the rest would be put on hold until the process is done (FCFS) or the quantum is reached or an I/O event occurs (Round-Robin).

-For phase 1, our design is a very simple list of processes along with their information, the processes are then sorted according to their arrival time since the algorithm implemented was FCFS.

-For phase 2, we are adding a second algorithm which has more restrictions on running processes, each process is given a time slice of a constant value, after the time slice for that process is done, the process is put in a waiting queue, and the time quantum is subtracted from the total processing time for that process, and the next process in the queue arrives next.

-For phase 3, we have further refined our algorithms, the time quantum is defined by the user and a dispatch penalty is added whenever a process is taken out of the processor. We also organized our code into several class files and a main class file that needs to be run, and added a better user interface that gives the user instructions on what to do.

We have also added methods for error checking, to make sure that the input the user provides does not break the logic of the program and is always valid. We have also implemented the

memory feature so that the user cannot add processes that exceed the memory limit, and when a process is done, the memory is freed and reused by the new process.

Strategy to achieve the goal design:

We constantly let the user know what is happening in the simulation, for instance, in order to demonstrate the memory aspect of the program, we constantly inform the user of how much memory is used and how much memory is freed to demonstrate that aspect of the simulation. The same is done for the scheduling algorithms where the processes are printed out each clock cycle in the correct order of processing.

User Manual:

In order to run the program, the user should be able to compile the code consisting of one main class and 4 class files that are needed for the program to function.
After downloading the .zip and extracting to a directory, open a command prompt and compile each of the class files before compiling the main class:
>> cd path/directoryName
example:

After changing directory to the correct folder, compile the class files:
>> javac pcb.java
>> javac FCFS.java
>> javac RR.java
>> javac generateData.java
finally, compile and run main class:
>> javac mainClass.java
>> java mainClass

From there, the user should follow the instructions that the program prompts.