XCRAM
Analysis Model

Submitted to:

Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:
Agluba, Gerry Jr. P.
Go, Sharleen Joy Y.
Silverio, Robelle C.

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY 2016-2017

Revision Control

History Revision:

Revision Date	***Person Responsible***	***Version Number***	***Modification***
10/14/16	Sharleen Joy Y. Go	1.0	Initial Document. Filled in the documents general information. Added the entites: Task, TimeBlock, Schedule and SavedSchedList, boundary: TaskInfoUI and controls: AddTask, EditTask, and ReuseSavedSchedule. Also, added a temporary class diagram.
10/18/16	Gerry P. Agluba	2.0	Added the boundary: DeleteTaskScheduleUI and controls: DeleteTask, ClearSchedule, and DeleteSavedSched
10/18/16	Sharleen Joy Y. Go	3.0	Added some methods in the class diagram. Added the description the control class CurrentScheduleStorage and boundary class SaveSchedUI
10/19/16	Robelle C. Silverio	4.0	Added the boundary: ViewTaskScheduleUI; controls: ViewCurrentSchedule,ViewSavedSchedule,ViewTaskInformation; entiry: priority queue
10/20/16	Gerry P. Agluba	5.0	Added methods in Deletion Classes, added attribute taskReferenceNumber to entity Task, redraw new class diagram
11/19/16	Sharleen Joy Y. Go	6.0	Modified the class diagram to agree with the prototype.

Purpose:

The purpose of this document is to present to the audience and programmers the components of the task scheduling system along with its attributes, methods and interaction with other components through an analysis model. For the programmers, this document will serve as a blueprint in the actual execution of the said software. For the audience, this document will give them a better idea on how the system works.

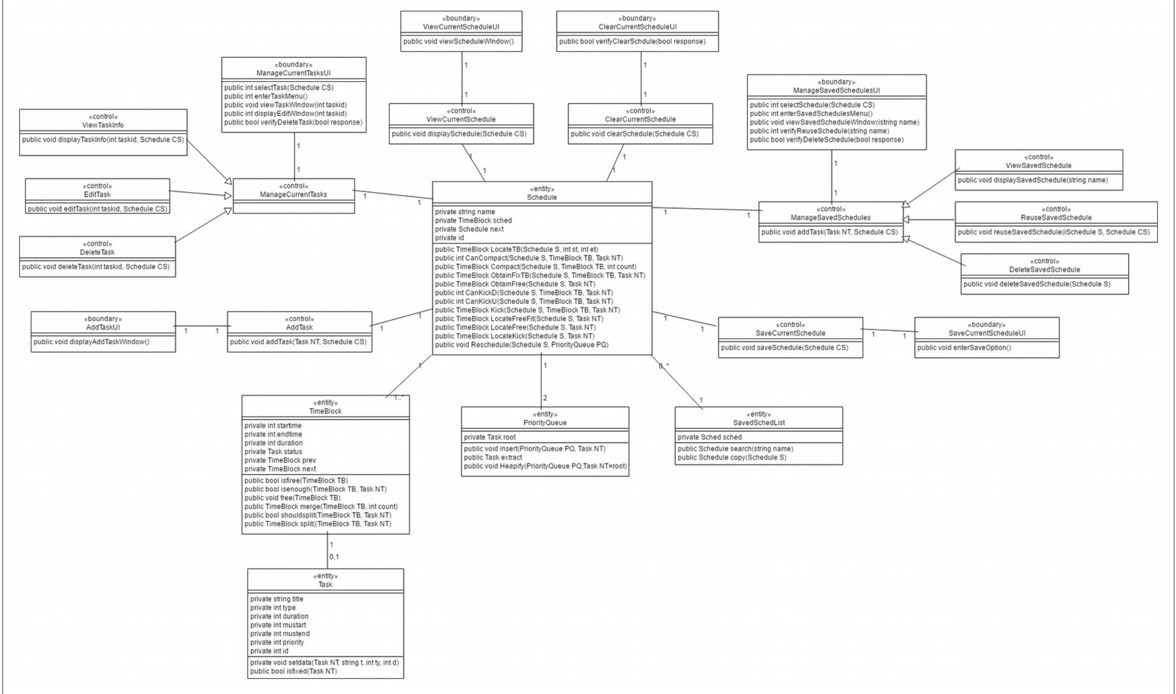
Audience:

Evaluators and Users

System Name: Task Scheduling System

Description: The system is basically an interconnection between the user and all the functionalies provided by the task scheduler. The user feeds this system with a series of tasks information which are all for the current day: the system does scheduling on a daily basis. In return, this system's functionalitites work together to provide the user with a one day schedule of tasks that satisfies all conditions inputted by the user for each task.

Class Diagram:



«boundary» ViewCurrentScheduleUI
public void viewScheduleWindow()
«boundary» ClearCurrentScheduleUI
public bool verifyClearSchdule(bool response)
«boundary» ManageCurrentTasksUI
public int selectTask(Schedule CS)
public int enterTaskMenu()
public void viewTaskWindow(int taskid)
public int displayEditWindow(int taskid)
public bool verifyDeleteTask(bool response)
«control» ViewTaskInfo
public void displayTaskInfo(int taskid, Schedule CS)
«control» EditTask
public void editTask(int taskid, Schedule CS)
«control» DeleteTask
public void deleteTask(int taskid, Schedule CS)
«control» ManageCurrentTasks
«control» ViewCurrentSchedule
public void displaySchedule(Schedule CS)
«control» ClearCurrentSchedule
public void clearSchedule(Schedule CS)
«boundary» ManageSavedSchedulesUI
public int selectSchedule(Schedule CS)
public int enterSavedSchedulesMenu()
public void viewSavedScheduleWindow(istring name)
public int verifyReuseSchedule(string name)
public bool verifyDeleteSchedule(bool response)
«control» ManageSavedSchedules
public void addTask(Task NT, Schedule CS)
«control» ViewSavedSchedule
public void displaySavedSchedule(string name)
«control» ReuseSavedSchedule
public void reuseSavedSchedule(iSchedule S, Schedule CS)
«control» DeleteSavedSchedule
public void deleteSavedSchedule(Schedule S)
«entity» Schedule
private string name
private TimeBlock sched
private Schedule next
private id
public TimeBlock LocateTB(Schedule S, int st, int et)
public int CanCompact(Schedule S, TimeBlock TB, Task NT)
public TimeBlock Compact(Schedule S, TimeBlock TB, int count)
public TimeBlock ObtainFixTB(Schedule S, TimeBlock TB, Task NT)
public TimeBlock ObtainFree(Schedule S, Task NT)
public int CanKickD(Schedule S, TimeBlock TB, Task NT)
public int CanKickU(Schedule S, TimeBlock TB, Task NT)
public TimeBlock Kick(Schedule S, TimeBlock TB, Task NT)
public TimeBlock LocateFreeFit(Schedule S, Task NT)
public TimeBlock LocateFree(Schedule S, Task NT)
public TimeBlock LocateKick(Schedule S, Task NT)
public void Reschedule(Schedule S, PriorityQueue PQ)
«boundary» AddTaskUI
public void displayAddTaskWindow()
«control» AddTask
public void addTask(Task NT, Schedule CS)
«control» SaveCurrentSchedule
public void saveSchedule(Schedule CS)
«boundary» SaveCurrentScheduleUI
public void enterSaveOption()
«entity» TimeBlock
private int startime
private int endtime
private int duration
private Task status
private TimeBlock prev
private TimeBlock next
public bool isfiree(TimeBlock TB)
public bool isenough(TimeBlock TB, Task NT)
public void free(TimeBlock TB)
public TimeBlock merge(TimeBlock TB, int count)
public bool shouldsplit(TimeBlock TB, Task NT)
public TimeBlock split((TimeBlock TB, Task NT)
«entity» PriorityQueue
private Task root
public void insert(PriorityQueue PQ, Task NT)
public Task extract
public void Heapify(PriorityQueue PQ,Task NT=root)
«entity» SavedSchedList
private Sched sched
public Schedule search(string name)
public Schedule copy(Schedule S)
«entity» Task
private string title
private int type
private int duration
private int mustart
private int mustend
private int priority
private int id
private void setdata(Task NT, string t, int ty, int d)
public bool isfixed(Task NT)
1
1..*
0..*
2
0,1

Boundary Classes:

Class Name	**Description**
AddTaskUI	This user interface has the responsibility of allowing the user to add a new task in the current schedule. (1) displayAddTaskWindow opens a window which allows the user to enter the required information regarding the new task that needs to be added in the current schedule.
ManageCurrentTasksUI	This user interface has the responsibility of allowing the user to view, edit and delete any task that is contained in the current schedule. (1)selectTask enables the user to choose which among the existing tasks in the current schedule he wishes to view, edit or delete. (2)enterTaskMenu opens a window that display the 3 options that the user has regarding the chosen task: View, Edit or Delete. (3)viewTaskWindow is used for displaying all the available information regarding the current state of the chosen task. (4)displayEditWindow opens a window that the user may use to edit information regarding the chosen task. (4)verifyDeleteTask is a method that prompts the user if he really wants to delete selected task.
ViewCurrentScheduleUI	This user interface allows the user to view the state of the current schedule. Through this user interface, the user is able to see which TimeBlocks are free and which are occupied by which tasks. (1) viewScheduleWindow displays all the TimeBlocks that make up the current schedule including the tasks that some may contain.
ClearCurrentScheduleUI	This user interface enables the user to completely empty the current schedule if he wishes to do so. (1)verifyClearSchedule is a method that prompts the user if he really wants to clear the current schedule.
ManageSavedSchedulesUI	This user inteface enables the user to view, reuse and delete any schedule that was previously saved (1)selectSchedule enables the user to choose which among the existing Schedules in the list of saved schedules he wishes to view, edit or delete. (2)enterSavedSchedulesMenu opens a window that display the 3 options that the user has regarding the chosen schedule: View, Edit or Delete. (3)viewSavedScheduleWindow displays the contents (TimeBlocks+Tasks) of the chosen schedule. (4)verifyReuseSchedule is a method that prompts the user if he really wants to load the chosen schedule as the current schedule. (5)verifyDeleteSchedule is a method that prompts the user if he really wants to remove the chosen schedule from the list of saved schedules.
SaveCurrentScheduleUI	This user interface has the responsibility of allowing the user to save the current schedule. (1)enterSaveOption prompts the user if he wants to add the current schedule in the list of saved schedules.

Control Classes:

Class Name	**Description**
AddTask	This control class will add the new task to the current schedule if nothing hinders its addition: space, priority and task type plays an important role in determining whether the new task will be added and/or some other tasks would be rescheduled or removed.
ManageCurrentTasks	This control class fetches the correct control class in charge of performing the operation that the user chooses regarding some specific task in the current schedule. Calls either one of these 3 control classes: ViewTaskInfo, EditTask or DeleteTask.
ViewTaskInfo	Retrieves all the relevant information regarding a chosen task.
EditTask	This control class will modify the current schedule in order to fit the new information of a preexisting task. Editing a task may result in rescheduling or removal of any task in the current schedule including the edited task. Similar to the addition of a new task, the 3 mentioned properties also play an important role in determining the resulting schedule after an editing is done.
DeleteTask	This control class will delete the selected task from the current schedule. Deletion only happens if it is verified by the user and selected task is present in the current schedule.
ViewCurrentSchedule	This control class will retrieve all the TimeBlocks and Tasks that make up the current schedule.
ClearCurrentSchedule	This control class will iteratively delete all the task in the current schedule.
ManageSavedSchedules	This control class fetches the correct control class in charge of performing the operation that the user chooses regarding some specific schedule in the list of saved schedules. Calls either one of these 3 control classes: ViewSavedSchedule, ReuseSavedSchedule or DeleteSavedSchedule
ViewSavedSchedule	This control class will retrieve all the TimeBlocks and Tasks that makes up the chosen saved schedule.
ReuseSavedSchedule	This control class will make the current schedule exactly like the chosen schedule to be reused.
DeleteSavedSchedule	This control class will delete the schedule that the user had chosen from the list of saved schedules.
SaveCurrentSchedule	This control class will first make a copy of the current schedule. Afterwards, the copy will be added to the list of saved schedules for future use.

Entity Classes:

Class Name	**Description**
Task	This entity represents any task that the user inputs. It contains all the important information of a task such as its: name, type, duration, priority (optional for flexible tasks), specified start time (for fixed task), computed end time(for fixed task), and an id which uniquely identifies a task. (1) The setTaskdata method is used to set a task's three basic attributes: name, type, and duration.(2) The isfixed method returns true if the task object represents a fixed task.
TimeBlock	This entity represents a block of time in a day. Its startime and endtime attributes specifies the range of time included in the block. It may or may not contain a task object. If it contains a task, the said task is kept in its status attribute; otherwise, the attribute's value is NULL. Also, its prev and next attributes are used to reference the TimeBlocks that come before and after it in a day. (1) The isfree method returns true if the TimeBlock TB does not contain a task. (2) The isenough method returns true if the TimeBlock TB is large enough to contain a Task object based on the TimeBlock's and task's duration. (3) The free method is used to empty the TimeBlock. (4) The merge method takes in as parameters TimeBlock TB along with an integer which is the number of TimeBlocks that follows TB which are to be merged with TB. The resulting, merged TimeBlock is then returned. (5) The shouldsplit method returns true if a task NT's duration is less than a given TimeBlock TB's duration. (6) The split method takes in as parameters a TimeBlock TB and a Task NT. It splits TB into either two or three TimeBlocks such that one of the TimeBlocks exactly fits NT. The said TimeBlock is then returned.

Schedule	This entity represents a single day's schedule. By default, it doesn't have a name and contains a single TimeBlock with starttime=0, endtime=2359 and status=NULL. After constant addition of Tasks by the user, the Schedule will contain a list of smaller, contiguous TimeBlocks. A schedule will only be named if it is to be saved. The attribute next refers to another Schedule which is stored in the list of saved schedules. The attribute id is used to uniquely identify a schedule. (1) The LocateTB method traverses the list of TimeBlocks in Schedule S to find the TimeBlock which is or is a part of the block of time containing st and et as start time and end times respectively. (2) The CanCompact method takes in as parameters a free TimeBlock TB and a Task NT. It returns an integer n which is the number of free TimeBlocks that follow TB in Schedule S. If compacting TB with abs(n) free TimeBlocks that follow it in the schedule will yield a large enough TimeBlock to fit NT, n is positive. A negative n means that the n free blocks plus TB won't yield a sufficient TimeBlock to fit NT; on the other hand, n=0 means that no free TimeBlocks follow TB. (3) The Compact method takes in a free TimeBlock TB along with a positive integer returned by the CanCompact method and performs the needed compaction. It returns a free TimeBlock having the duration of TB plus those of the n free TimeBlocks that follow TB. (4) The ObtainFixTB method takes in as parameter the TimeBlock TB that was returned by the LocateTB method. If TB is both free and enough to contain NT, it is simply returned by ObtainFixTB. Otherwise, ObtainFixTB will find a way to produce the needed TB either by temporarily or permanently removing some flexible tasks or compacting free TimeBlocks when possible. This method will return either NULL or a TimeBlock whose duration is greater than or equal to the duration of the task and contains the start and end time specified by the user for the fixed task. (5) The ObtainFreeTB takes in as parameter a new flexible task NT that the user wants to add to the current schedule. With the information contained in NT, this method will try to obtain a large enough TimeBlock that can fit NT either by temporarily or permanently removing other flexible tasks whose priorities are less than NT or compacting free TimeBlocks when possible. The method will return either NULL or a TimeBlock whose duration is greater than or equal to the duration of NT. (6) The CanKickU receives as input a new flexible task NT and a TimeBlock TB containing a flexible task whose priority is less than NT's. This method returns a positive integer n if temporarily removing n flexible tasks (TB's task+flexible tasks that precede TB whose priority is less than TB's) will result in a large enough, free TimeBlock to contain NT. The return value of this method is either negative or positive; it will never return 0 because TB is counted as 1. (7) The CanKickD method is similar to the CanKickU method, its only difference is that it considers flexible tasks that follow TB rather than those that precede it. (8) The Kick method receives as input a TimeBlock TB and an integer count: abs(count) is the output of either CanKickU or CanKickD. This method pushes the flexible tasks contained in TimeBlocks TB and the n TimeBlocks that either precede (if count is -) or follow (if count is +) TB to a PriorityQueue for later readdition in the current schedule. This method returns a free TimeBlock resulting from the merging of TB, the free TimeBlocks and the n TimeBlocks that were freed. (9) The LocateFreeFit method searches Schedule S for a TimeBlock that is free and enough to contain Task NT. (10) The LocateFree method traverses the list of TimeBlocks of Schedule S to check whether merging free TimeBlocks will form a large enough TimeBlock to contain NT. This method returns the merged TimeBlock. (11) The LocateKick method traverses the list of TimeBlocks starting from the TimeBlock with the lowest priority until the TimeBlock whose priority is only 1 less than that of NT to decide whether removing some task will give space for NT's addition.
SavedSchedList	This is simply a list of all the Schedules which the user chose to save.

	(1) The Search method traverses the list to find the Schedule whose name is the same as the inputted string. (2) The Copy method makes a copy of the chosen saved schedule and sets this as the current schedule.
PriorityQueue	This entity contains the list of task/s kicked by the scheduler. The task/s listed in this entity is/are either to be vanished totally from the current schedule or it could be rescheduled. (1)Insert method adds a task in the queue. (2) Extract returns the task that will be rescheduled. (3)Heapify maintains the characteristics of the heapsorted priority queue