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:

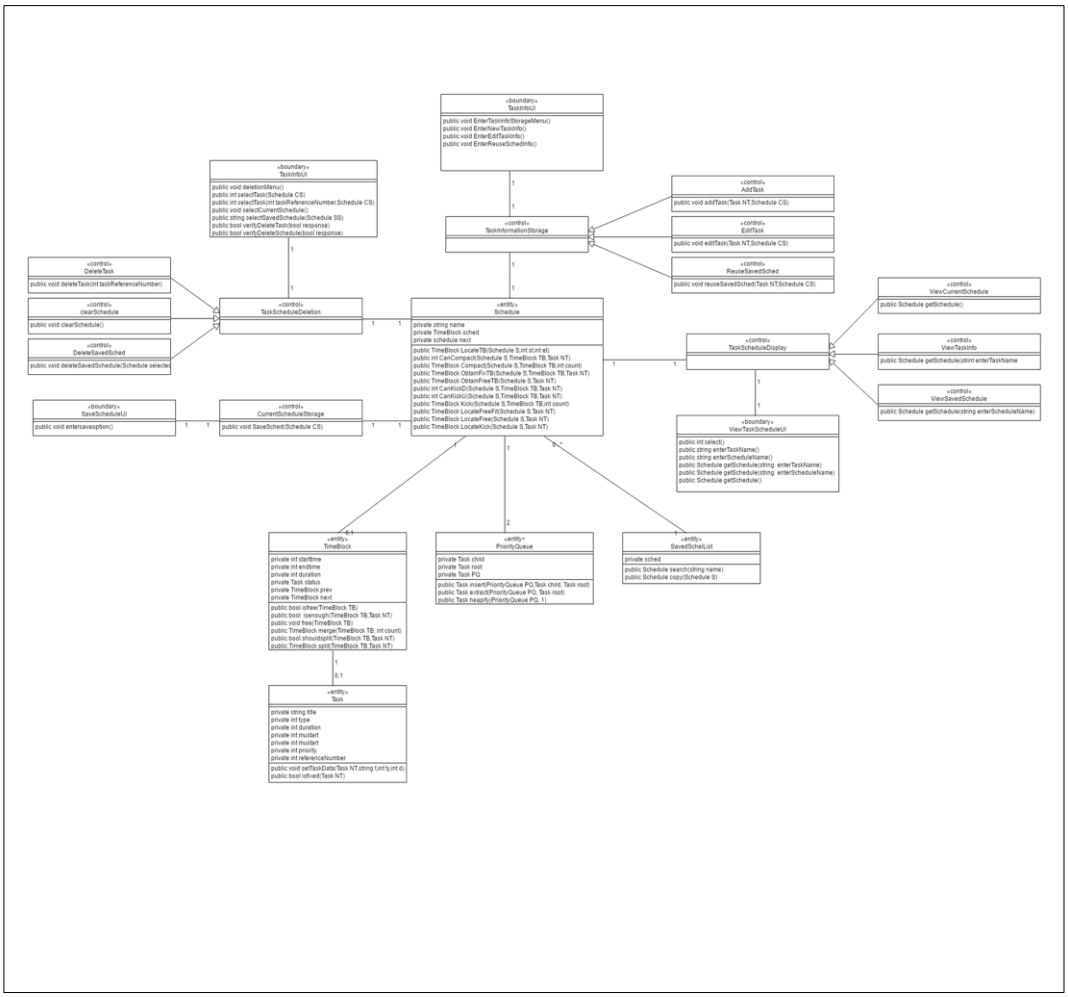
Revision Date	***Person Responsible***	***Version Number***	***Modification***
10/14/16	Sharleen Joy Y. Go	1.0	Initial Document. Filled in the documents general information. Added the entites: Task, TimeBlock, Schedule and SavedSchedList, boundary: TaskInfoUI and controls: AddTask, EditTask, and ReuseSavedSchedule. 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
10/20/2016	Gerry P. Agluba	4.1	Added methods in Deletion Classes, added attribute taskReferenceNumber to entity Task, redraw new class diagram

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 Classes:

Class Name	**Description**
TaskInfoUI	This user interface has the responsibility of allowing the user to enter any new information that he wishes to add to the current schedule. This interface gives the user three options: add a new task, edit the information of any task existing in the current schedule or reuse a schedule that was previously saved. (1) EnterTaskInfoStorageMenu is a method which displays the three options that this UI offers. (2) EnterNewTaskInfo receives the user's inputted information regarding a new task that he want to add to the current schedule.(3) EnterEditTaskInfo will prompt the user to enter the start time and end time of an existing task in the current schedule to identify the specific task that the user wants to edit. It also receives the new information regarding the said task. (4) EnterReuseSchedInfo will prompt the user for the name of the saved schedule which he wants to reuse.
DeleteTaskScheduleUI	This user interface has the responsibility of allowing the user to delete a task or a schedule as a whole. Specifically, this interface gives the user 3 options all mainly for the purpose of deletion. These are deleting a task from the current schedule, clearing the current schedule(equivalent of deleting every task from the current schedule), and also the capability to delete selected schedule from the list of saved schedules. (1) deletionMenu is a method which displays the three options that this UI offers. Task Selection can be done in two ways. These is handled by (2) selectTask. This method can have an integer or string as an argument. The integer argument refers to the task's reference number from the Schedule, this variant of the selectTask method is useful when user just want to pick the task without fully entering the title of the task to be deleted. The string argument refers to entering fully the title of the task to be deleted. Either way, this similar methods returns an integer(use as a reference in searching for the task in the current schedule) that will be passed to the control class TaskScheduleDeletion after verification is done. (3) selectCurrentSchedule allows the user to select the current schedule subject to deletion. (4)selectSavedSchedule is a method that allows user to select schedule from the list of saved schedules. After selection, verification from the user is necessary. Verification methods are as follow: (5) verifyDeleteTask() method prompts the user if he wants to really delete selected task. (6) verifyDeleteSchedule prompts the user if he/she wants to surely delete selected schedule. Also these verification methods, will signal the methods in the control class to proceed to deletion or not.

SaveScheduleUI	This user interface has the responsibility of allowing the user to save the current schedule.
ViewTaskScheduleUI	This user interface has the responsibility of allowing user to choose from the three different view options such as view current schedule, view saved schedule, and view task's information. When user has chosen a particular viewing option, the UI will ask the user of the needed information according to that selected viewing option. (1) Select receives user's integer input such that 1 means to view current schedule, 2 means to view saved schedule, and 3 means to view task's info. (2) enterTaskName asks user to input the task name he/she wished to display its information. (3) enterScheduleName prompts the user to enter the schedule's name he/she wants to be displayed. (4) getschedule is responsible of displaying what the user wanted to see with respect to the data entered by the user beforehand

Control Classes:

Class Name	**Description**
TaskInformationStorage	This control class has the responsibility of adjusting the current schedule in order to agree with the new information that the user had entered into the system through the TaskInfoUI. Under this control class is the AddTask, EditTask and ReuseSavedSchedule control classes.
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.
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.
ReuseSavedSchedule	This control class will make the current schedule exactly like the chosen schedule to be reused.
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.
ClearSchedule	This control class will iteratively delete all the task in the current schedule.
DeleteSavedSched	This control class will delete the selected schedule the user has specified. Deletion only happens if it is verified by the user and selected task is present in the list of saved schedules.
CurrentScheduleStorage	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.
Task Schedule Display	This control class will choose one viewing option.
View Current Schedule	This control class will retrieve the list of task for the day.
View Task Information	This control class will check if the task name entered is existing in the list of task. If it is existing, information of that task will be retrieved, otherwise a null information will be returned.
View Saved Schedule	This control class will check if the schedule name entered is existing in the list of saved schedule. If it is existing, that schedule will be retrieved otherwise no schedule will be returned.

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) and computed end time(for fixed task).It also includes task reference number reference purposes. This number is also necessary to maintain the uniqeness of the task in the a particular schedule. (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 startime=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. (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