Exercise: Borg Calendar as case study

The running software

...

Software requirements specification

Extract to hand round....

First task

Ignoring the implementation of the user interface as far as possible (but being guided by the fundamental capabilities the description of the UI requirements exposes), and ignoring all the things which say "explained later":

sketch a conceptual class model for this section, focusing on the relationships between projects, tasks, subtasks.

Second task

There is (in a class TaskModel) a saveProject operation, taking project p as argument. This has to validate the project as it saves it, which means it has to check that every task in the project has a due date which is no later than the due date of the whole project. If any task does have a due date which is too late, it will flag this on the task by sending it the setLate() message. It will return true if the validation succeeded, otherwise false.

Draw a sequence diagram to show this, using fragments as appropriate.

Third task

Tasks go through various states in their lifetime. In fact, the states and transitions are configurable by the user - you might want to look at the way this works. However, for now, suppose:

The user opens a new task and fills in initial details (details may be updated at any time). Normally, the user then marks the task as in progress until it's been done, then as closed. However, any time before the task is closed, it may be deferred. A deferred task can be reopened (so that it can be re-evaluated), or marked as being in progress. A closed task may be reopened (for evaluation) if more information comes to light. Eventually the closed task is archived, after which nothing more can happen to it.

Draw a protocol state machine diagram for Task.

Note: in the process, you have defined (part of) an API for Task.