A Meeting Scheduling Problem

Design a system to schedule meetings and meeting rooms. A user can use this system simply to request a room of a given size for a given period of time. For example, a user can request a room that will hold 30 people from 1 p.m. until 3 p.m. this Friday. In addition, a user can request that an existing meeting (already defined in the system with a set of attendees) be scheduled at with a particular starting time and ending time. For example, a user can ask to have the Browser Project staff meeting scheduled this Thursday from 2 p.m. until 3 p.m. (That meeting has already been defined in the system and currently includes 11 attendees.)

A user can cancel any scheduled meeting or any room assignment up until the point at which the meeting or assignment begins (i.e., up until 1 p.m. on Friday and 2 p.m. on Thursday in the above two examples, respectively).

When a meeting is scheduled, an electronic message about that meeting must be sent to each attendee. Likewise, when a meeting is canceled, each attendee must be informed by electronic mail about the cancellation.

A user must also be able to define or alter a meeting. When defining the meeting, the user provides a list of attendees. The user may alter a meeting definition by adding attendees to or removing attendees from the meeting. A user may also remove an entire meeting definition. Note that adding or removing attendees has no effect on scheduled instances of that meeting (unless the last attendee is removed from a meeting, in which case future scheduled occurrences of that meeting should be canceled). A result of removing a meeting, on the other hand, is that all scheduled instances of that meeting must be canceled.

Assume the existence of a Post Office package that contains Post Office and Address classes. The Post Office class defines one method:

> `deliverMessage(recipient : Address, message : String)`. It delivers the specified message to the specified recipient. (Assume that all messages are delivered.)

Assume the existence of an Employee Management package that defines an employee management component. That package exports a facade class, Employee Management, and an Employee interface class. The facade class defines the following methods:

> `employee(employeeNumber : integer) : Employee`. Given an employee number, this method returns as type Employee a reference to an object defining the employee with that employee number.

The Employee interface defines the (abstract) methods:

`address( ) : Address`. This method returns the electronic mail address of the employee.

`name( ) : String`. This method returns the name of the employee.

