A Hotel Management system  
COMP9116 Project  
Draft Requirements  

October 4, 2002  

Due date  Week 14  

Please note: this is a draft assignment specification.  
Please notify any problems in the specification.

1 The project

The project is to construct a specification, refinement and implementation of a hotel management system.

1.1 Brief Overview

The hotel management system maintains information on rooms, room bookings, services and services used by guests in a hotel. There will be a register of rooms available, their class and price. Guests will reserve certain classes of rooms for a particular period of time. At some point guests will arrive and begin using their rooms and the hotel’s services. Then at some time later guests will pay their account and checkout.

2 Requirements

2.1 Concepts to be Modelled

The system must model the following:

- **room classes**: There will be some set of room classes such as, single, double, twin, etc.
- **rooms**: rooms available in hotel.
- **room prices**: prices of rooms according to class.
- **services**: services available to guests.
- **service prices**: prices of services.
- **guests**: people who can use hotels rooms and services.
- **guest rating**: guest rating; used to determine discounts on rooms and services.
bookings:  room bookings made by guests. A guest may book more that one room.

waiting list:  if appropriate rooms are unavailable, guests may go onto a waiting queue.

accounts:  accounts per guest per stay.

dates:  used in bookings, waiting, account.

2.2 Operations

\[ \text{status} \leftarrow \text{AddRoom}(\text{room, class}) \]: add a room of a particular class.

\[ \text{status} \leftarrow \text{RemoveRoom}(\text{room}) \]: remove a room.

\[ \text{status} \leftarrow \text{ChangeRoom}(\text{room, class}) \]: change the class of a room

\[ \text{status} \leftarrow \text{AddService}(\text{service, price}) \]: add a service at price

\[ \text{status} \leftarrow \text{RemoveService}(\text{service}) \]: remove a service.

\[ \text{status} \leftarrow \text{UpdateService}(\text{service, price}) \]: update price of service

\[ \text{status} \leftarrow \text{UpdateClass}(\text{class, price}) \]: update price of room class

\[ \text{status, bid} \leftarrow \text{BookRoom}(\text{name, class, from, to}) \]: book a room of a particular class for a period from..to for name.

\[ \text{status} \leftarrow \text{CancelBooking}(\text{bid}) \]: cancel a previous booking.

\[ \text{status, wid} \leftarrow \text{AddWaiting}(\text{name, class, from, to}) \]: add booking details to waiting list.

\[ \text{status} \leftarrow \text{RemoveWaiting}(\text{wid}) \]: remove booking from waiting list.

\[ \text{status} \leftarrow \text{CheckWaiting} \]: check waiting list and promote requests to bookings if possible.

\[ \text{status, bid} \leftarrow \text{ConfirmBooking}(\text{wid}) \]: check whether booking on waiting list has become a firm booking.

\[ \text{status, rid} \leftarrow \text{CheckInBooked}(\text{name, bid}) \]: checkin for a room that was booked.

\[ \text{status, rid} \leftarrow \text{CheckInCasual}(\text{name, class, from, to}) \]: casual checkin (unbooked)

\[ \text{status, cost} \leftarrow \text{UseService}(\text{rid, service}) \]: use a service.

\[ \text{status} \leftarrow \text{CheckOut}(\text{rid}) \]: checkout.

\[ \text{status} \leftarrow \text{PayAccount}(\text{rid, payment}) \]: pay account.

2.3 Modelling Sets, Constants and Definitions

Context machines should be used for sets, constants and definitions.

You will need to model date.
3 What you should do

1. Develop “low-level” machines with preconditioned, fragile operations.

2. Animate the specifications and develop animation scripts that illustrate the checking of the various conditions in the requirements.

3. Discharge the proof obligations.

4. Develop robust specifications.

5. Discharge the proof obligations.

6. Develop refinements.

7. Discharge proof obligations.

8. Develop implementations.

9. Discharge as many proof obligations as possible.

4 The Challenge

The challenge of this assignment is to resist any temptation to rush to implementation, and to work your way through the refinements, implementations and associated proof obligations. This way you will experience the novelty of generating code that must be correct, modulo the correctness of the specification.

Good luck!