Human-Centred Systems Design Group Project

3. UML Diagrams

You will develop a number of UML diagrams as part of your solution, capturing relevant information from the scenario. Marks are awarded for capturing information succinctly and accurately. Do not invent extra features; and try to create the simplest possible solution that captures the relevant information. *Some additional diagrams will be required for postgraduate teams*.

3.1 Use Case Diagram

[ALL STUDENTS]: Create a UML use case diagram to capture the information from the *business context* (see 2.1 above). The diagram should capture all the actors involved, all the tasks carried out, including those in the wider business context. Identify tasks which will be automated in the system by placing these inside the system boundary. Identify and label three subsystems dealing with registration, appointments and payments. You do not need to write full use case descriptions.

You may find it helpful to approach the problem in the following way:

- Identify the actors involved in the scenario and any commonality between actors. Hint: do some of them behave the same way, for some tasks?
- Identify the use cases in the scenario, choosing verb-style names or phrases if possible. Keep to the smallest set of high-level use cases. Hint: if the behaviour is identical for different actors, merge the use cases.
- Link the actors to the use cases in which they participate. In rough, this may look tangled, but you can rearrange the layout in your final version. Hint: can you simplify actor participations?
- Draw one or more boundaries to identify the system and different subsystems. In rough, the use cases may be badly placed for grouping inside a boundary, but you can rearrange them in your final version.
- Do any of the tasks have partial sub-tasks? Do any tasks have variant extensions? Not necessarily; but if so, use the appropriate dependency.

3.2 Activity Diagram

[POSTGRADUATES ONLY]: Create three activity diagrams to capture the information from the *business context* (see 2.1 above). Create one diagram to represent a regular check-up and hygienist visit, one diagram to represent a course of treatment visits, and one master diagram to represent registration, the visits, rebooking and payment. Ignore cancellation and re-booking of appointments.

You may find it helpful to approach the problem in the following way:

- Some activities are repeated in a cycle. Hint: separate out the cyclic activities and sketch the loop in rough, identifying choice and merge points for the loop.
- Some activities can happen in any order, or in either order. Hint: treat such activities as happening in parallel, identifying concurrent fork and join points.

- Some activities happen optionally. Hint: identify the choice point, and also any later merge point where the flows join again.
- Some activities happen in strict sequence, but others also require an elapsed time. Hint: use a timer trigger *in parallel with* the main control flow, meeting at a concurrent join.
- Make sure your diagrams have single entry points and, where appropriate, exit points. Hint: assume the master diagram continues forever.
- Using the UML syntax for guards, label each branching choice with a short phrase with a true/false interpretation. Hint: you may need state variables to regulate the flow.

3.3 Information Model (Class Diagram)

[ALL STUDENTS]: Create a UML class diagram to capture the information model from the *business information* (see 2.2 above), showing the classes, attributes and associations. Mark the ends of associations with suitable multiplicities and role-names. This stage requires an *Information Model*; not a normalised *Data Model*.

You may find it helpful to approach the problem in the following way:

- Identify and name the information entities in the description. Hint: entities are structured objects - do not create entities for unstructured concepts.
- Identify the attributes contained by each entity. Hint: attributes are atomic properties that are not deconstructed in the description; only include attributes mentioned explicitly in the information.
- Identify any associations between entities, marking off the multiplicities in each direction. Hint: for each source entity, how many destination entities?
- Sketch the diagram, using the information gathered above, making sure you distinguish entities and attributes. Hint: do not include, as an attribute, any entity that you have linked by an association.
- An information model should seek to capture exactly the information described - do not introduce extra key fields or extra linker tables that are part of a normalised data model (this comes later).
- Check whether there are any association classes required - are there any relationships for which we need to record attributes?
- [POSTGRADUATES ONLY]: check whether any classes should declare operation signatures, to support your OCL specifications (see below).

3.4 OCL Specifications

[POSTGRADUATES ONLY]: Using all the interview information given above, write suitable expressions in OCL to assert the following semantic properties of your design. Your answers may only refer to attributes, role-names and methods specified in your class diagram. Your answers must be in correctly-formed OCL syntax, specifying the relevant context:

- Write an OCL invariant stating that if the patient's age is under eighteen, then they have a NHS Free Plan, which has no monthly charge.
- Write an OCL invariant stating that the duration of an appointment is equivalent to its end time minus its start time. Hint: this is defining the result of a method.
- Write an OCL invariant stating that the total cost for an appointment is the sum of the costs of each treatment that the patient received.

- Write an OCL precondition stating that appointments can only be booked between 09:00 and 17:00, inclusively. Hint: constrain the construction arguments for an appointment.
- Write an OCL invariant stating that for all appointments, no appointment overlaps with any other appointment.
- Write an OCL postcondition for when an appointment is checked-out, stating how the remaining annual credit on the patient's healthcare plan is adjusted.

The last two are significantly harder to specify! See the OCL guides on Dr Simons' module page for further general OCL advice.

3.5 Data Model (Class Diagram)

[ALL STUDENTS]: Create a normalised *data model* in the UML class diagram database profile (not in other ERM tools please), by normalising your *information model* (see 3.3 above). Show any required linker tables, and all primary and foreign keys. Only add surrogate keys if the domain does not have a natural key or composite key, or if the composite key would need more than two attribute-parts. Add the direction of navigation to all associations.

You may find it helpful to approach the problem in the following way:

- Identify natural primary keys and composite keys in each class from your information model; if they have none, create a surrogate key for the corresponding table.
- Normalise any many-to-many associations between classes in your information model: these will need linker-tables in the normalised model;
- Identify association classes (if any) in your information model: these will be promoted to linker-tables in the normalised model;
- Write the revised, normalised multiplicities between the above linker-tables and the tables that they link;
- Identify the foreign keys used by linker-tables and ensure they correspond to the primary keys of the linked tables; these also become the composite key of the linker-table.
- Deal similarly with any one-to-many relationship, creating suitable foreign keys; and then add navigation directions to every association.

3.6 State Machine Diagram

[POSTGRADUATES ONLY]: Create a UML protocol state machine, describing the interaction modes of the touch-screen interface used by the partners. This machine should consist of a number of states in which certain actions are possible only in that state.

You may find it helpful to approach the problem in the following way:

- The default view will allow selection of either the dentist's or the hygienist's calendar for the day (different for each partner);
- The calendar will display a list of appointments for that partner; it is possible to exit to the default view, or select an appointment;
- When viewing an appointment, it is possible to select treatments, or exit to the calendar view (without commit), or commit the appointment (logging any treatments);

- When viewing treatments, it is possible to select a treatment to add it to the current appointment; selecting will also exit to the appointment view (displaying the added treatment); it is possible to exit (without selecting) to the appointment view.