Objects in Conceptual Model

Maintaining Calendars, Appointments, Contact Lists etc. in the Appointment Scheduling application are concepts. The Calendars, Appointments, Contact Lists etc. are the Objects.

Every Conceptual Model will have Objects for performing tasks.

The Conceptual Model should clearly identify those Objects. The users will eventually interact with these Objects while carrying out tasks within the web application.

Other than identifying the Objects, the Conceptual Model should also identify the relationships between these objects. This relationship between the objects can be hierarchical or non-hierarchical.

For example, Personal Calendars and Professional Calendars are types of Calendars or we can say that they are child of the Calendar object. Similarly, Calendars will contain Appointments. This forms the non-hierarchical relationship between the Calendar and the Appointment objects.

Defining such relationships between objects in the Conceptual Model will give you a good idea of how to present these objects to the user in the UI.

Actions and Attributes of the Objects

Every object will have a set of attributes and actions that can be performed on the object.
Start Date, Duration, Attendees etc. are all attributes of the Appointment object. While create, edit, delete etc. are actions that be be performed on the Appointment object.

The Conceptual Model should clearly identify these attributes and actions. Doing this will help you immensely when you sit down to design the actual UI of the web application.

Now, since you have already defined the relationships between objects, you can think of making the actions generic so that they can be used across the different objects.

For example, an action to inactivate a Calendar can be carried out on both Personal as well as Professional Calendars.

Defining the Terminology

Every Conceptual Model should clearly define the standard terminology that will be used within the application to represent the concepts, objects, actions etc.

For example, Appointments should not be called as Meetings. If you are standardizing on the term Appointment then make sure that this same term is used throughout the application. Do not call it as Meetings in one place and Appointments in another.

But if you want to use the term Meetings somewhere, then you need to justify why you want to change the term and how it is different from Appointments.

Defining the terms not only applies to the objects but also to the attributes, actions, help text, hints, messages etc.

Defining the terms early on in the Conceptual Model will help to keep your entire development team on the same page as well as give the users a better user experience.

Defining Use Cases

Once the Conceptual Model is designed, you can write the use cases detailing the scenarios of how the user will interact with the application while carrying out the various tasks within the application.

For the Appointment Scheduling web application example, you can write use cases of how the user will carry out the task of creating an Appointment, Calendar, adding a Contact to the list etc.

Note that the use cases are only detailing how the user will interact with the application and will not be mention the specifics of the UI.

For example, the use case will say that the user will open the Calendar and create an Appointment on the desired date. It will not say that the user will right click inside the Calendar on the desired date and invoke the create action to create an Appointment.

How the task will actually be carried out in the UI should be designed while designing the UI.

Designing the UI from the Conceptual Model

Now, that the Conceptual Model is in place and you have defined the use cases, you can start designing the UI based on the Conceptual Model and the use cases.

From the Conceptual Model you already know the following things:

1. The tasks the user can carry out,
2. The objects that the user will work with,
3. The relationship between the objects,
4. The attributes of the objects,
5. The actions the user can take on the objects,
6. The actions that can be made generic across the objects,
7. The terms to be used to refer the objects, attributes, actions etc.

From the use cases you know how the user will interact with the application.

Once you know these things you can now design the UI by turning the abstract concepts, tasks, actions etc. of the Conceptual Model into concrete presentations of UI fields, icons, menus, layouts etc.

You can now re-write the use cases to be more UI specific.