MODELING A CLOUD SAAS
2. MODELING A CLOUD SAAS

2.1. Introduction

Model Driven Architecture (MDA) is a standardization effort by OMG which aims at reducing the gap between business modeling and software development significantly, by ensuring that business models drive the application development. While traditional software design and development processes create applications for deployment to a specific technology platform, MDA introduces higher levels of abstraction, enabling organizations to create models that are independent of any particular technology platform. The entire software development process in MDA is model-driven, with models as the prime artifacts which are created, transformed or generated during the process. Although the high level frameworks such as J2EE, .NET and other Web services also raise the level of abstraction but they still focus on computer-level concepts rather than business level concepts.

This chapter focuses on the incorporation of the model-driven software development methodology in the development of cloud SaaS (Cloud Software-as-a-Service). As mentioned earlier, the MDA approach defines the cloud software application at three different levels of abstraction. Accordingly, the CIM of the Cloud SaaS would specify the computation independent view of the system. It would capture the requirements of the system in a vocabulary familiar to the domain practitioners. The PIM of the Cloud SaaS would specify the system at the next lower level of abstraction. It would capture a platform independent view focusing on the operation of the system while hiding the platform specific details. The PSM of the Cloud SaaS is defined at the next lower level.
of abstraction, focusing on the details of use of a particular platform, thereby providing a platform specific view of the system. Based on MDA, a single PIM for a cloud software application can be mapped to several different PSMs targeted on different platforms by defining platform-specific transformation rules. An MDA based development of cloud SaaS (application) will enable defining the cloud services in a technology-independent manner and will play a significant role in improving the quality of cloud software services, making them more robust, flexible and agile [65][67].

In this chapter, we illustrate the Computation Independent Model (CIM) and Platform Independent Model (PIM) of a cloud software application – the Online Hotel Reservation System (OHRS), taken as example. The transformation of PIM into PSMs is discussed in the following chapter. Section 2.2 describes the OHRS cloud SaaS. Next in this chapter, section 2.3 and section 2.4 respectively elaborate the CIM and PIM of this OHRS Cloud SaaS. Section 2.5 draws the conclusion of the chapter.

2.2.OHRS Cloud SaaS Application

The model-driven approach to a cloud SaaS (application) development is illustrated with the help of an example – the Online Hotel Reservation System (OHRS) [65]. The OHRS is a software application that runs as a service in the cloud, performing a variety of tasks for its clients such as determining the availability of rooms in the hotel, online reservations, online cancellations, generating arrival charts, generating reports for decision-making, modifying room tariffs etc. Besides, as the application is web-based, it can be accessed by the prospective customers of the hotel enterprise to determine the availability of accommodation, or to book accommodation, or to cancel a previously booked accommodation. The OHRS cloud service may thus be utilized by any small or medium enterprise (SME), on payment basis. The proposal would also be beneficial for
budding entrepreneurs who wish to start up with a hotel business on a small scale but do not wish to initially invest huge capital in purchasing, installing or developing the hardware and software and other infrastructure for the purpose.

Although, the MDA approach does not restrict itself to the use of UML (Unified Modeling Language) for modeling the software applications, the authors are using UML to model different aspects of the cloud SaaS.

2.3. Computation Independent Model (CIM) of OHRS

As part of the CIM, the Use Case diagram and the Activity diagram available in UML have been used to model the functional requirements of the system under consideration. The loosely defined requirements of the business process are distilled into these requirements models without losing essential details.

2.3.1. Use Case Diagram of OHRS

A Use Case diagram for OHRS cloud application is depicted in Figure 2.1 [65][66]. A use case, represented by an oval in the diagram, captures a piece of functionality that the system provides. It specifies ‘what’ a system is supposed to do. The characteristics of the actors interacting with the OHRS are:

- A Customer is a person who accesses the OHRS to view the availability status or to book/cancel an accommodation in the hotel.

- An Administrator is the hotel personnel who accesses the OHRS for not only viewing the availability status or booking/cancelling accommodations for its customers over the counter, but also to generate various reports for decision making or for modifying the tariffs of unit types.
In OHRS, the process of booking an accommodation involves checking the availability and if available, verifying the payment details prior to booking. The Use Case diagram depicts this by means of include relationship which declares that the use case at the head of the dotted arrow completely reuses all the steps from the use case being included. The ‘Book accommodation’ use case includes the ‘Check Availability’ and ‘Verify Payment Details’ use cases. Similarly, the ‘Cancel booking’ use case includes

Figure 2.1 Use Case Diagram of OHRS
the ‘Verify Customer Information’ use case. The ‘View Customer Status’ use case is a
generalization of ‘View Reservation’ and ‘View Cancellation’ use cases. The

generalization relationship represents use case inheritance by applying a use case
with small changes for a collection of specific situations. Also, the ‘Generate Report’ use
case is a generalization of ‘Reservation Status’, ‘Cancellation Status’ and ‘Revenue
Earned’ use cases. The ‘Cancel booking’ use case extends ‘Generate Cancellation
Receipt’ use case. An extend relationship represents an optional reuse of a use case
depending on a runtime behavior or a system implementation decision [17].

2.3.2. Activity Diagram of OHRS

The Activity diagram specifies ‘how’ a system would accomplish its goals. It models a
business process which represents a set of coordinated tasks that achieve a business goal.

Figure 2.2 depicts the Activity diagram for the OHRS cloud SaaS under consideration
[65][66]. The high-level actions are chained together to represent the business process of
the OHRS.

The various steps in the process may be listed as:

1. A user (customer or administrator) logs into the hotel website with a valid user ID
   and password.

2. A customer may perform various tasks such as viewing the availability status, book
   accommodation or cancel a previously booked accommodation.

3. In order to view the availability status, the customer submits the date in response to
   which the system displays the availability status of the hotel for a predefined period.

4. The customer may then either quit the application or may proceed to book an
   accommodation based on its availability.
5. In response to the book accommodation action, the system generates a unique ID for the customer.

6. Next, the customer submits his details such as name, address, phone, email address, unit type, number of units, book-from date and book-to date and payment details.

![Activity Diagram of OHRS](image)

**Figure 2.2 Activity Diagram of OHRS**
7. A booking receipt bearing all the details is generated for the customer and the process terminate:

8. A customer may also cancel a previous booking, in which case he submits the customer ID that was generated at the time of booking.

9. In response to step 8, the system displays the details of the customer.

10. The customer submits the cancellation request.

11. A cancellation receipt is generated for the customer and the process terminates.

12. An administrator, after successful log-in, is able to perform all the tasks that a customer does and can also generate reports or modify tariffs by submitting the required information.

Ideally, in MDA-based software development, the requirements model should be simply submitted to the generators that would produce the required systems. But, in actual practice the requirements model need to be refined further into a computational model that a generator can process.

### 2.4. Platform Independent Model (PIM) of OHRS

A PIM specifies the system at a higher level of abstraction as compared to a PSM. There is only one single PIM for the software system and it can be mapped to several different PSMs targeted on different platforms by defining platform-specific transformation rules for the purpose.

The PIM of a system may be represented using a UML class diagram which exhibits classes and relationships among them. A class is a blueprint or a template for the objects that are instantiated from it. It describes two essential pieces of information – the state of the object represented by attributes and the behavior of the object represented by
operations. The classes do not exist in isolation. Instead, they work together using different types of relationships such as dependency, association, aggregation, composition and generalization (also known as inheritance). A dependency is a weak relationship between two classes, and requires a class to know about another class in order to use objects of that class. An association, as compared to dependency, is a stronger relationship between two classes, and implies that a class contains a reference to
one or more objects of the other class. Besides, an association may sometimes introduce
a new class, called the association class. *Aggregation* represents a stronger version of
association indicating that a class actually owns but may share objects of another class. A
*composition* is a relatively stronger relationship than aggregation and represents a whole-
part relationship between the classes. A *generalization* (or *inheritance*) is used to
describe a class that is a type of another class.

The PIM for the OHRS is depicted in Figure 2.3, by means of a class diagram [65][66].
The classes, their attributes and operations, and the relationships among various classes
are shown in the model. The accessor and mutator methods for the attributes are not
defined explicitly in the class, as the MDA mappings automatically expand the attributes
into corresponding accessor and mutator operations. A multiplicity adorns each
relationship. This PIM defines the static aspects of the OHRS application through a static
view. Though this model reflects the technicalities of the system, it is non-committal to
the platform that would implement and host the system.

2.4.1. Classes in the Class Diagram of OHRS PIM

The various classes specified in the PIM for OHRS are [66]:

1. *Hotel* – A hotel is composed of one or more unit types. The Hotel class is therefore
depicted at the *whole* end of the composition relationship between Hotel and
UnitType classes. The various attributes representing the state of the hotel object
include ID, name, address and phone. This class is also related to the Customer class
in a one-to-many association representing that a Hotel may have zero or more
customers associated with it. A one-to-many association also exists between Hotel
and Administrator classes representing that the Hotel has at least one Administrator
associated with it.
2. **UnitType** – The UnitType class is depicted at the part end of the composition relationship and includes the attributes such as ID, name, total_units and tariff to represent the state of its object. The class is related to Customer and Administrator classes in a many-to-many association displaying that a customer or an administrator may book one or more unit types in the hotel.

3. **Customer** – A Customer class is characterized by attributes such as ID, name, address, phone and email to represent the state of customer object. A unique ID is generated for each customer at the time of booking an accommodation. An ID relates the customer to a Hotel. The class also provides methods that enable an online customer to view the availability of rooms in the hotel, and also to book an accommodation or cancel a previously booked accommodation.

4. **Administrator** – The Administrator class has several attributes like ID, name, address, phone and email that characterize the state of administrator object. Besides, it includes the methods that facilitate viewing of availability/reservation/cancellation status of the various rooms in the hotel, book or cancel accommodation for the customer, update the hotel database and generate various reports for the management.

5. **User** – The User class with its attributes – username and password, is related to both Customer and Administrator classes in a one-to-one association. It also includes the methods to facilitate login and logout by the users.

Besides these classes there are two association classes – Payment and Booking – in the OHRS PIM. An association class is useful in complex cases. It is related to two classes which in turn have a relationship with each other.

6. **Payment** – This association class is related to Customer and Hotel classes, which themselves are related. The attributes of this class include payment ID, payment date,
payment mode and amount paid to represent the state of the class objects.

7. Booking – This class represents the association between Customer-UnitType and Administrator-UnitType classes. It includes various attributes such as date of booking, number of units booked, book from date and book to date for representing the state of Booking objects.

2.4.2. Relationships among the Classes in OHRS PIM

The relationships among the classes are defined as under [65]:

- A hotel is a composite aggregation of unit types.
- A hotel has one or more unit types.
- A hotel may have zero or more customers.
- A customer with a unique ID belongs to only one hotel.
- A hotel has one or more Administrators.
- An administrator belongs to only one hotel.
- A customer may book one or more unit types.
- A unit type may be booked by zero or more customers.
- An administrator may book zero or more unit types.
- A unit type may be booked by zero or more Administrators.
- A customer makes one or more payments to the hotel.
- A payment is related to only one customer.
- A unit type is related to zero or more bookings.
- A booking is related to only one unit type.
- A user may be a customer or an administrator.
- A customer is essentially a user of the system.
- An administrator is essentially a user of the system.
The PIM of the OHRS cloud SaaS describes the attributes and operations in a manner that is entirely independent of any programming language or operating system in which the system would finally be implemented.

2.5. Conclusion

The Model Driven Architecture is a software development framework defined by OMG. This framework is neutral with respect to the languages and methodologies that are used to model or code the system. Though the MDA development life cycle is not very different from the traditional software development life cycle, the software development process in MDA is driven by the modeling activity with well-defined, formal models used to describe the system. The models at the core of MDA are CIM, PIM and PSM. The MDA framework is complemented with a variety of other OMG standards such as UML, MOF, CWM etc enabling even the non-OMG organizations to develop their own standards that fit seamlessly in the MDA framework.

In this chapter, the authors have presented a description of a Cloud SaaS application – the Online Hotel Reservation System (OHRS), followed by the generation of its CIM and PIM. The CIM has been specified with the help of UML Use Case Diagram and Activity Diagram, whereas the PIM has been specified with the help of UML Class Diagram. The next chapter elaborates the transformation of the OHRS PIM into various specific PSMs.