MODEL TRANSFORMATION – PIM TO PSM
3. MODEL TRANSFORMATION –
PIM TO PSM

3.1. Introduction

In MDA, a system is described by a single PIM in a platform agnostic manner which is then used to derive several PSMs targeted on different platforms by employing different implementation technologies. A key characteristic of MDA, which places it above the other model-driven software development approaches, is the use of automated or semi-automated tools for M2M and M2C transformations. These tools execute transformation definitions that are specified for the purpose. This chapter describes the transformation of the PIM of Online Hotel Reservation System (OHRS) specified in the previous chapter into two PSMs. Section 3.2 describes the transformation of the PIM into an EJB PSM. It begins with a brief discussion of the EJB Component Model and is followed by the specification of the transformation definition for transforming the PIM into an EJB PSM. Section 3.3 briefly discusses the Relational Model, and also specifies the transformation definition for transforming the PIM into a Relational PSM. Section 3.4 concludes the chapter.

3.2. Transforming OHRS PIM into an EJB PSM

3.2.1. EJB Component Model

A component model for the software system may be built by simply generating an EJB component for each class. However, to optimize the performance of the EJB components in a distributed environment with remote access, it becomes crucial that the communication between the components remains minimal. Therefore, instead of
exchanging the attributes of an object individually by invoking the get- or set- operation on each attribute value and burden the communication network with large scale movement of data, a better option is to exchange all the attributes of an object in a single remote call based on value-object pattern. This will result in messages that are more complex with relatively high amount of data, and in component interfaces that have a relatively low number of operations, thus relieving the burden on communication network [23].

Furthermore, the coarse-grained EJB component model may be used wherein the components are large and have infrequent interactions with relatively high amount of data in each interaction. This can be achieved by clustering closely related classes into one single component, the EJB data schema. With fewer components in the system, the inter-component communication will not load the network.

![Diagram](image)

**Figure 3.1 Top-Level EJB Component Model of OHRS**

The granularity of the EJB components is based on the composition of classes in the domain model. A class which is a part of the whole in a composition relationship is clustered into the data schema generated from the whole. Besides, the association class is clustered into the data schema generated from the associated class that is able to navigate to the other associated class. Accordingly, with reference to the OHRS PIM depicted in Figure 2.3 in the previous chapter, the class UNITTYPE is clustered with class HOTEL,
and the classes PAYMENT and BOOKING are clustered with the class CUSTOMER. Figure 3.1 depicts the resultant four EJB components – HOTEL, CUSTOMER, ADMINISTRATOR and USER.

### 3.2.1.1. Transformation Definition for PIM to EJB PSM Transformation

The transformation definition for transforming a PIM into an EJB PSM comprises of a set of transformation rules which are listed as under [23]:

1. For each PIM class, an EJB key is generated.

2. Each PIM class that is not a composite part of another PIM class is transformed into an EJB component and an EJB data schema.

3. Each PIM class is transformed into an EJB data class residing in an EJB data schema generated from the PIM class that is the outermost composition of the transformed PIM class.

4. Each PIM association is transformed into an EJB association within an EJB data schema generated from the PIM class that is the outermost composition of the transformed PIM association.

5. Each PIM association class is transformed into two EJB associations and an EJB data class. The EJB associations and the EJB data class are generated within the EJB data schema generated from the PIM class that is the outermost composition of the PIM class that can navigate across the transformed PIM association class.

6. Each PIM attribute of a class is transformed into an EJB attribute of the mapped EJB data class.
Figure 3.2 EJB Component Model of OHR System Including the Data Schema (PSM)
Each PIM operation is transformed into an EJB operation of the generated EJB component generated from the PIM class that is the outermost composition of the PIM class that owns the transformed PIM operation.

The term 'outermost composition of x' in the transformation rules refers to the class that is not a part (via a composite association) of another class and is equal to or the (direct or indirect) container of the class x.

Figure 3.2 depicts an EJB PSM for the OHRS cloud software system. The USER and ADMINISTRATOR entity components have not been depicted in this figure due to space constraints.

3.3. Transforming OHRS PIM to Relational PSM

3.3.1. Relational Data Model

In a relational data model, the relation is the only construct required to represent the entities, attributes and relationships among different entities. A relation may be visualized as a named table. Each column of the table corresponds to an attribute of the relation and is named correspondingly. The rows of a relation are referred to as tuples of the relation and the columns are its attributes. Each attribute of a relation has a distinct name. In a relational model, no two rows are identical and the ordering of the rows is not significant. A relationship is represented by combining the primary keys of the entities involved in a relation, besides its own attributes, if any [68].

3.3.1.1. Transformation Definition for PIM to Relational PSM Transformation

The transformation rules for generating a relational database model are consistent with object-relational mapping [23]. These rules describe how the elements in PIM can be mapped to elements in Relational PSM. The Relational PSM specifies the database and
is described by a relational model depicted in an Entity-Relationship diagram in Figure 3.3. Based on the various elements in the PIM, the transformation rules are categorized into [65][66]:

A. Transformation Rules for Data types: A basic data type is mapped to corresponding data type in the relational model according to the following rules:

i. A UML string is mapped onto a SQL VARCHAR(n).

ii. A UML integer is mapped onto a SQL INTEGER.

iii. A UML real is mapped onto a SQL REAL.

iv. A UML date is mapped onto a SQL DATE.

A data type may be a simple data type or a derived data type.

a. Transforming Simple Data types: A simple data type is mapped directly to a column in a table.

b. Transforming Derived Data types:

i. A UML data type, such as a struct or an array, that has attributes but no operations, is mapped onto a number of columns, each representing a field in the derived data type.

ii. A UML data type such as a class is mapped to a table itself. The column holds a reference (foreign key) to a key value in the other table.

B. Transformation Rules for Classes and Attributes:

a. Each UML class is transformed into a Relation/Table with the same name.

b. Each UML attribute is transformed into a field (column) in the table with the same name.

c. When the type of attribute is not a data type but a class, the field in the table holds a foreign key to the table representing that class.
C. Transformation Rules for Associations:

a. Associations in the UML model are transformed into a foreign key relation in a database model, possibly introducing a new class.

b. An association class represents a relationship between two classes. It is transformed into a table with foreign keys in this new table referring to the two related tables.

c. The multiplicities of an association from class A to class B may be:

- The multiplicity at A is zero-or-one,
- The multiplicity at A is one, or
- The multiplicity at A is more than one.

The same holds for the multiplicity at B resulting in nine different combinations of multiplicities at both the ends. A pseudocode for the rule may be expressed as:

```plaintext
if the association A to B is adorned by an association class or the multiplicity at both ends of the association is more-than-one
then create a table representing the association class or the association and create foreign keys in this new table referring to the related tables
else if the multiplicity at one end of the association is zero-or-one
then create a foreign key in the table representing the class at more-than-one end, referencing class at the zero-or-one end
```
Figure 3.3 Relational Model of OHRS (PSM)
else /* the multiplicity of the association is one-to-one */

create a foreign key in one of the tables, referencing the other end

dendif
dendif

The PIM of OHRS is transformed into a Relational PSM based on the transformation rules defined above. The Relational PSM specifies the database and is described by a relational model depicted in an Entity-Relationship diagram in Figure 3.3.

3.4. Formalizing PIM to Relational PSM Transformation

Definition

In this section the authors have formalized the transformation definition specified in section 3.3.1.1. A transformation tool has been developed by the authors for executing the transformation rules. The tool itself has been implemented on Java platform and has been developed using NetBeans IDE. Java Server Pages (JSP) has been used to implement the presentation logic, while Java HTTP servlets have been used to implement application logic.

Annexure A displays the snapshots of PIM to Relational PSM transformation process executed using the transformation tool developed by the authors.

The workflow diagram representing the PIM to Relational PSM transformation logic is depicted in Figure 3.4.
Figure 3.4 Work Flow Diagram of PIM to Relational PSM Transformation
3.5. Conclusion

As pointed out earlier, the MDA essentially pays off when (semi)automated transformation tools are used for transforming the source model into target model. The transformation definitions become more complex when the structures in the source and target models differ greatly. Small changes in the source model may have huge impact on the structure of target models and therefore on the implementation code.

In this chapter, two platform-specific models (PSM) – a Relational PSM and an EJB PSM for the OHRS cloud application have been presented. These PSMs mainly focus on the structural aspects of the system. The Relational model is based on the Entity-Relationship model with each PIM class being transformed into an entity type or a relation and the associations among the classes being transformed into relationships. The EJB model leverages UML Profiles. The EJB PSM is based on the coarse-grained EJB component model. In this model, the closely related classes are clustered into a single component, resulting in fewer components in the system. Consequently, the reduced number of components helps to lessen the traffic on the network. The transformation definitions for transforming the PIM into respective PSMs are also specified in this chapter.