Chapter VIII

Information Systems Analysis with FOOM

This chapter elaborates on the activities and products of the analysis stage with functional and object oriented methodology (FOOM). The products of this stage include a data/objects model in the form of an initial class diagram, and a functional model in the form of hierarchical object oriented Data flow diagram (OO-DFDs). The two diagram types are synchronized in order to guarantee the correctness and completeness of the two models. The chapter presents various examples of diagrams of both types.

Data Modeling: Creating an Initial Class Diagram

An initial class diagram includes, as we already know, data/entity classes. These classes are derived from the users’ requirement and are made up of objects which keep real-world data.1 For each class in the initial class diagram we only define its attributes and relationships (but not its functions). At this stage we already know how to create an initial class diagram. Recall that there are two alternative ways to create it: (1) “directly”—based on the users’ requirements; and (2) “indirectly”—first to create an entity relationship diagram (ERD) and then to map it to an initial class diagram using the mapping algorithm.2
In Chapter VII we dealt with the possible orders of activities in the analysis stage and came to the conclusion that it is preferred to start by creating a data model and continue by creating a functional model. Since we have already studied how to create an initial class diagram, here we will only show another example of such a diagram. The example is more detailed than previous examples and involves both data (structural) and functional requirements, because we will use it also to demonstrate functional modeling.

**Example: The Apartments Building Information System**

An information system (IS) needs to be developed for a company which manages an apartment building. The system will provide information on the tenants, their monthly payments to the company (for maintenance of the building), and maintenance works requested by the tenants and conducted by contractors. Here is a description of the requirements from the system:

**Tenant Fees**

The IS will be able to store, update, and retrieve information on the tenants of the building and their payments to the company. For each apartment it will store the apartment number, its size (in square meters), the family name living in the apartment, their phone numbers, and the number of persons. Based on these details (and other considerations), the managing company will determine, for each apartment/family the monthly fees for maintenance of the building. The fee may be updated from time to time by the company. The tenants may pay their dues in cash or by check. The system will store the payment details (including the amount, date, and form of payment), and produce a receipt for the tenant. The system will enable the company to retrieve and report on the payments and debts of certain apartments (families) in certain periods (months). At the end of each month the system will produce a report on all tenants detailing the amounts paid and debt (as based on the monthly fees on one hand and the actual payments on the other hand). The total debt of all tenants will be stored to enable easy retrieval. In addition, the system will issue a notification (reminder) on the debt to each tenant whose debt is bigger than a three months fee, or whose debt is bigger than a two months fee but who already received a notification in the previous month.

**Maintenance Works**

The maintenance works for the building will be dealt with as follows: A tenant or several tenants together may request a maintenance work to be commenced in any of the common (public) parts of the building. For every
Related Content

**Automatic Verification of OOD Pattern Applications**
[www.igi-global.com/chapter/automatic-verification-ood-pattern-applications/27361?camid=4v1a](www.igi-global.com/chapter/automatic-verification-ood-pattern-applications/27361?camid=4v1a)

**Heuristics and Metrics for OO Refactoring: A Consolidation and Appraisal of Current Issues**
[www.igi-global.com/chapter/heuristics-metrics-refactoring/27363?camid=4v1a](www.igi-global.com/chapter/heuristics-metrics-refactoring/27363?camid=4v1a)

**Design Patterns as Laws of Quality**
[www.igi-global.com/chapter/design-patterns-laws-quality/27360?camid=4v1a](www.igi-global.com/chapter/design-patterns-laws-quality/27360?camid=4v1a)

**Model-Based Development: Metamodeling, Transformation and Verification**
[www.igi-global.com/chapter/model-based-development/25641?camid=4v1a](www.igi-global.com/chapter/model-based-development/25641?camid=4v1a)