

## “E-Film Hiring” Project

### HW2. Software Requirements Analysis

#### Deliveries: Domain Model, System Operation Model and Contracts

### SECOND MACRO-ACTIVITY: SOFTWARE REQUIREMENTS ANALYSIS (CONT.)

#### Second Step: Building the Domain Model

A domain model illustrates meaningful (to the modelers) conceptual classes in a problem domain and it is the most important artifact to create during object-oriented analysis. A domain model is a representation of real-world concepts, not of software components. It is *not* a set of diagrams describing software classes, or software objects with responsibilities. We will use it as a source of inspiration for designing software objects.

The domain model is the main candidate for reuse in software development. It derives from the business model by filtering the application meaningful business objects. So, the domain model step usually precedes or is carried out concurrently with the use case model step. During Software Requirements Analysis the reused domain model will be only updated from the requirements analysis.

In our case we derive the domain model directly from both the business model and use cases. Here is the procedure of the domain model construction:

- a. Identify the domain concepts from the problem statement and use case descriptions.
- b. Promote the domain concepts as classes in the domain model.
- c. For the objects of each class identified at the previous step, identify and analyze the application- meaningful properties.
- d. Identify conceptual connections between concepts from the problem statement and use case descriptions.
- e. Promote conceptual connections as associations between corresponding classes in the domain model. Refine them identifying the compositions (eventually aggregations)

#### Third Step: Design System Sequence Diagrams

Use cases describe how external actors interact with the software system we are interested in creating. During this interaction an actor generates events to a system, usually requesting some operation in response. To identify system events, it is necessary to have a clear choice of system boundary (see the previous HW).

It is desirable to isolate and illustrate the operations that an external actor requests of a system, because they are an important part of understanding system behavior. The

UML includes **sequence diagrams** as a notation that can illustrate actor interactions and the operations initiated by them.

**A system sequence diagram** is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and inter-system events. All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.

1. Design system sequence diagrams for the main success scenario of the use case, and frequent or complex alternative scenarios.
2. Identify **system operations** as the first messages in the system sequence diagrams.

#### **Fourth Step: Detail operations with contracts**

Use cases are our primary mechanism to describe system behavior, and are usually sufficient. However, sometimes a more detailed description of system behavior is needed. Contracts describe detailed system behavior in terms of state changes to objects in the Domain Model, after a system operation has executed (instance creation and deletion, attribute modification, link created and broken).

1. Write the contracts for at least two system operations identified at previous step.
2. Put emphasis on the postconditions. To describe the postconditions, use the following categories of changes in the system state:
  - instance creation and deletion
  - attribute modification
  - links formed and broken