III. Class and Object Diagrams

Classes, Attributes and Operations
Objects and Multi-objects
Generalization and Inheritance
Associations and Multiplicity
Aggregation and Composition
Business Objects and Rules

Classes
- A class describes a group of objects with
  - similar properties (attributes),
  - common behaviour (operations),
  - common relationships to other objects,
  - and common meaning (“semantics”).
- Finding classes: Listen to the domain experts (…the people who know the domain you are modeling!)

Diagrammatic Notation for Classes

System Classes
- This is a Java class to be included in the design of the new system

Attributes
- Each class can have attributes which represent useful information about instances of a class.
- Each attribute has a type.
- For example, Campaign has attributes title and datePaid.

Objects are Class Instances
- SaveTheKids: Campaign
  - title: "Save the kids"
  - datePaid: 28/01/02
### Object Diagrams

*Bill Clinton:* Student

*Monica:* Student

*Jaelson:* Instructor

*Someone:* Multiobjects

```
Course

courseNo: csc340
description: "OOAD"
```

### Multiobjects

A **multiobject** is a set of objects, with an undefined number of elements

```
I: Instructor

C1: Course

C2: Course

C3: Course

S: Student
```

### Operations

- Often derived from action verbs in the description of the application.
- Operations describe what can be done with the instances of a class.

```
Campaign

| Title: String |
| CampaignStartDate: Date |
| CampaignFinishDate: Date |
| EstimatedCost: Money |
| ActualCost: Money |
| CompletionDate: Date |
| DatePaid: Date |
| Completed(CompletionDate: Date, ActualCost: Money) |
| SetFinishDate(FinishDate: Date) |
| RecordPayment(DatePaid: Date) |
| CostDifference(): Money |
```

### Visibility

- As with Java, attributes and operations can be declared with different visibility modes:
  - **public:** any class can use the feature (attribute or operation);
  - **protected:** any descendant of the class can use the feature;
  - **private:** only the class itself can use the feature.

```
Staff

name : String
passwd : String
dateofB : Date

ChangePasswd()
Include()
```

### Relationships

- Classes and objects do not exist in isolation from one another.
- A relationship represents a connection among things.
- In UML, there are different types of relationships:
  - Generalization
  - Association
  - Aggregation
  - Composition
  - …more…
### Generalization

- **Superclass or parent**: StaffMember
- **Subclasses or children**: AdminStaff, CreativeStaff

- **Attributes**:
  - staff#: Integer
  - name: String
  - startDate: Date
  - grade

- **Operations**:
  - Hire()
  - ChangeGrade()
  - CalculateBonus()

### Inheritance

- **Inheritance of attributes**
- **Inheritance of operations**
- **Overriding inherited attributes or operations.**

### Finding Inheritance, ... Bottom Up

- **Advert**
  - Hoarding Advert
  - Press Advert
  - Video Advert
  - Billboard
  - Newspaper Advert
  - Magazine Advert

### Finding Inheritance, ... Bottom Up

- **RecordCD**
  - title
  - catalogue#
  - publisher
  - artist
  - acquisition#
  - Loan()
  - Return()

### Generalization Notation

- **Possibly overlapping**: Maria is both Lecturer and Student
- **Mutually exclusive**: A lecturer can't be a student and vice versa

- **All classes in this diagram model real world entities**

### ...Better!

- **LoanItem**
  - title
  - acquisition#
  - Loan()
  - Return()

- **Book**
  - title
  - author
  - publisher
  - ISBN

- **Record**
  - artist
  - catalogue#
  - recordCo

- **Person**
  - Student
  - Lecturer
  - Student
  - Lecturer
Classification

- This is the relationship between an object and the classes of which it is an instance.
- Traditional object models assume that classification is single and static.
- Multiple classification allows an object to be an instance of several classes that are not is-a related to each other; for example, Maria may be an instance of GradStudent and Employee.
- Dynamic classification allows an object to change its type during its lifetime.

Multiple Classification

```
Person
  +-- TA
  +-- Professor
  |    +-- TA
  |    +-- Professor
  +-- Staff
  |    +-- TA
  |    +-- Professor
  +-- Student
```

Association Relationships

```
StaffMember
  +-- name
  +-- staff#
  +-- startDate
  +-- qualification

Campaign
  +-- title
  +-- startDate
  +-- estimatedCost
```

Association Multiplicity

- “A staff member can manage zero or more campaigns”
- “Each campaign is managed by exactly one staff member”

Association Navigation: Uni-Directional Associations

```
PersonName

Telephone#
```

Associations and Roles

```
Company
  +-- * hires 1..* Person
```

```
Person
  +-- * employer 1..* Company
  +-- * supervisee 1..* Person
  +-- * employee 1..* Company
  +-- * supervisor 1..* Person
```
Association Classes

Company

Person

* employer

hires

1..*

Can't have the same person work for the same company more than once!

Job

description

salary

Aggregation Relationship

Campaign

Advert

contains

Composition Relationship

A composition relationship implies strong ownership of the part by the whole.

For example, the relationship between a person and her head is a composition relationship, and so is the relationship between a car and its engine.

In a composition relationship, the whole is responsible for the disposition of its parts, i.e., the composite must manage the creation and destruction of its parts.

Another Example

Engine

Person

Train

Car

Composition

Aggregation

Object Diagrams, Again
Business Objects and Rules

- Business objects and rules document -- in a structured way -- a class diagram.
- Such a documentation is also called data dictionary.
- Business objects are represented as classes.
- A business rule can be:
  - an integrity constraint on the data of the application,
  - a derivation rule, whereby information can be derived from other information within a class diagram.

Examples of Business Objects

<table>
<thead>
<tr>
<th>Classes</th>
<th>Description</th>
<th>Attributes</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>Employee working in the company</td>
<td>Code, Surname, Salary, Age</td>
<td>Code</td>
</tr>
<tr>
<td>PROJECT</td>
<td>Company project on which employees are working</td>
<td>Name, Budget, ReleaseDate</td>
<td>Name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Associations</th>
<th>Description</th>
<th>Classes involved</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANAGEMENT</td>
<td>Associate a manager with a department</td>
<td>Employee (0,1), Department (1,1)</td>
<td></td>
</tr>
<tr>
<td>MEMBERSHIP</td>
<td>Associate an employee with a department</td>
<td>Employee (0,1), Department (1,N)</td>
<td>StartDate</td>
</tr>
</tbody>
</table>

Examples of Business Rules

<table>
<thead>
<tr>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>(BR1) The manager of a department must belong to that department.</td>
</tr>
<tr>
<td>(BR2) An employee must not have a salary greater than that of the manager of the department to which he or she belongs.</td>
</tr>
<tr>
<td>(BR3) A department of the Rome branch must be managed by an employee with more than 10 years' employment with the company.</td>
</tr>
<tr>
<td>(BR4) An employee who does not belong to a particular department must not participate in any project.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Derivations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(BR5) The budget for a project is obtained by multiplying the sum of the salaries of the employees who are working on it by 3.</td>
</tr>
</tbody>
</table>

Additional Readings