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

- StaffMember
  - staffName
  - CalculateBonus()
  - ChangeGrade()

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.

```
Campaign
  title: String
  datePaid: Date
```

Objects are Class Instances

```
SaveTheKids: Campaign
  title: "Save the kids"
  datePaid: 28/01/02
```
Object Diagrams

Multiobjects

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

<table>
<thead>
<tr>
<th>Title: String</th>
<th>CampaignStartDate: Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>CampaignFinishDate: Date</td>
<td>EstimatedCost: Money</td>
</tr>
<tr>
<td>ActualCost: Money</td>
<td>CompletionDate: Date</td>
</tr>
<tr>
<td>DatePaid: Date</td>
<td>Completed(CompletionDate: Date, ActualCost: Money)</td>
</tr>
<tr>
<td>SetFinishDate(FinishDate: Date)</td>
<td>RecordPayment(DatePaid: Date)</td>
</tr>
<tr>
<td>CostDifference(): Money</td>
<td></td>
</tr>
</tbody>
</table>
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**

- **StaffMember**
  - staff#: Integer
  - name: String
  - startDate: Date
  - grade
  - Hire()
  - ChangeGrade()
  - CalculateBonus()

- **AdminStaff**
  - Hire()
  - CalculateBonus()

- **CreativeStaff**
  - qualifications
  - Hire()
  - CalculateBonus()

**Superclass or parent**

**Subclasses or children**

---

**Inheritance**

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

Advert

Hoarding Advert

Press Advert

Video Advert

Newspaper Advert

Magazine Advert

Finding Inheritance, ... Bottom Up

Book

- title
- author
- publisher
- ISBN
- DeweyCode
- acquisition#
- Loan()
- Return()

RecordCD

- title
- catalogue#
- publisher
- artist
- acquisition#
- Loan()
- Return()
...Better!

All classes in this diagram model real world entities.

Generalization Notation

Possibly overlapping  
Maria is both Lecturer and Student

Mutually exclusive  
a lecturer can't be a student and vice versa
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
  \-- Male
  \-- Female
  \-- TA
  \-- Professor
  \-- Student
    \-- <<mandatory>> student
    \-- <<dynamic>> role
  \-- <<mandatory>> sex
  \-- <<dynamic>> role
```

```
Person
  \-- TA
  \-- Professor
  \-- Student
    \-- <<mandatory>> student
    \-- <<dynamic>> role
  \-- <<mandatory>> sex
  \-- <<dynamic>> role
```
**Association Relationships**

![Class Diagram](image)

**StaffMember**
- name
- staff#
- startDate
- qualification

**Campaign**
- title
- startDate
- estimatedCost

name \(\rightarrow\) manages \(\rightarrow\) role

**Association Multiplicity**

How many instances of a class can participate in an association of a particular type?

<table>
<thead>
<tr>
<th>StaffMember</th>
<th>Campaign</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>title</td>
</tr>
<tr>
<td>staff#</td>
<td>startDate</td>
</tr>
<tr>
<td>startDate</td>
<td>estimatedCost</td>
</tr>
</tbody>
</table>

1 \(\rightarrow\) 0..*  

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

- Some examples of specifying multiplicity:

  Optional (0 or 1) 0..1
  Exactly one 1 = 1..1
  Zero or more 0..* = *
  One or more 1..*
  A range of values 1..5
  A set of ranges 1..3,7..10,15,19..*

**Direction of an Association**

```
<table>
<thead>
<tr>
<th>StaffMember</th>
<th>Campaign</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>title</td>
</tr>
<tr>
<td>staff#</td>
<td>startDate</td>
</tr>
<tr>
<td>startDate</td>
<td>estimatedCost</td>
</tr>
<tr>
<td>qualification</td>
<td></td>
</tr>
</tbody>
</table>
```

manages →
Association Navigation:
Uni-Directional Associations

Associations and Roles
**Association Classes**

![Diagram showing association classes]

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

- **Person**
  - employee

- **Job**
  - description
  - salary

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

---

**Aggregation Relationship**

![Diagram showing aggregation relationship]

- **Campaign**
  - 1 contains Advert

- **Advert**
  - *
**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.

---

**Example**

```plaintext
Order
- date: Date
- code: Integer
- total: Currency
+ Confirm()
+ Cancel()
- Total(): Currency

OrderItem
- quantity: Integer
- price: Currency

Product
```

---

```
\* 1
```

```
\* 1
```

```
1
```
Another Example

Information Systems Analysis and Design

Class Diagrams -- 31

Composition

Aggregation

Object Diagrams, Again

Information Systems Analysis and Design

Class Diagrams -- 32

©2004 John Mylopoulos

©2004 John Mylopoulos
**Business Objects and Rules**

- Business objects and rules document -- in a structured way -- a class diagram.
- Such a documentation is also called *data dictionary*.
- 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>

©2004 John Mylopoulos
# Examples of Business Rules

## Constraints

- **(BR1)** The manager of a department must belong to that department.
- **(BR2)** An employee must not have a salary greater than that of the manager of the department to which he or she belongs.
- **(BR3)** A department of the Rome branch must be managed by an employee with more than 10 years’ employment with the company.
- **(BR4)** An employee who does not belong to a particular department must not participate in any project.

## Derivations

- **(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.

## Additional Readings