CONCEPTUAL DESIGN: UML CLASS DIAGRAM
RELATIONSHIPS
A Simplified Object-Oriented Systems Analysis & Conceptual Design Methodology

Activities

1. Identify the information system’s purpose
2. Identify the information system’s actors and features
3. Identify Use Cases and create a Use Case Diagram
4. Identify Objects and their Classes and create a Class Diagram
5. Create Interaction/Scenario Diagrams
6. Create Detail Logic for Operations
7. Repeat activities 1-6 as required to refine the “blueprints”
Objects

• Objects have three responsibilities:
  - What they know about themselves – (e.g., Attributes)
  - What they do – (e.g., Operations)
  - What they know about other objects – (e.g., Relationships)
Defining Class

A CLASS is a template (specification, blueprint) for a collection of objects that share a common set of attributes and operations.

<table>
<thead>
<tr>
<th>HealthClubMember</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
</tr>
<tr>
<td>operations</td>
</tr>
</tbody>
</table>

Class → Objects
• Relationships

A RELATIONSHIP is what a class or an object knows about another class or object.

Generalization (Class-to-Class) (Superclass/Subclass)
- Inheritance
  - Ex: Person - FacultyPerson, StudentPerson, Staff...
  - Ex: ModesOfTravel - Airplane, Train, Auto, Cycle, Boat...

[Object] Associations
- FacultyInformation - CourseInformation
- StudentInformation - CourseInformation

[Object] Aggregations & Composition (Whole-Part)
- Assembly - Parts
- Group - Members
- Container - Contents
- **Relationships**
  - Exist to:
    1) show relationships
    2) enforce integrity
    3) help produce results

In this example:
- Removal of a University Course should also remove Students that are in the Course but not Student Information.
- Removal of a Student should also remove the Courses that the Student is in but not the University Course.
- Removal of a Student in a Course should not affect either University Course or Student Information.
UML Class Diagram Notation

Expanded view of a Class into its three sections:
- Top: Class Name
- Middle: attributes
- Bottom: operations

Class

attributes

operations

Member

- memberNumber
- firstName
- lastName
- telephone
- address
- city
- etc...

- checkOutVideo
- checkInVideo
- buyItem
- etc...
UML Class Diagram Notation

Class Generalization Relationship

Object Aggregation Association
1..* 0..*

Object Composition Association
1 0..*

Will always be “1”
Class Diagram Relationships

- **Class**
  - Generalization

- **Object**
  - Association
  - Aggregation
  - Composition
Generalization (Class-to-Class) (superclass – subclass; supertype – subtype)

- A Generalization follows a “is a” or “is a kind of” heuristic from a specialization class to the generalization class. (e.g., student “is a” person, video “is a kind of” inventory).
- Common attributes, operations and relationships are located in the generalization class and are inherited by the specialization classes.
- Unique attributes, operations and relationships are located in the specialization classes.
- Inherited attributes and operations may be overridden or enhanced in the specialization class depending on programming language support.
- Inherited operations in the specialization classes may be polymorphic.
- Only use when objects do NOT “transmute” (add, copy, delete).
- Multiple inheritance is allowed in the UML but can complicate the class model’s understanding and implementation (e.g., C++ supports but Java and Smalltalk do not).
Generalization Example

<<abstract>>
Role
attributes
operations

Faculty
attributes
operations

Student
attributes
operations

Staff
attributes
operations

Visitor
attributes
operations

Others:
• Transactions
• Things
• Places
• Etc...

Note: <<abstract>> = no objects
Poor Generalization Example
(violates the “is a” or “is a kind of” heuristic)
Generalization Inheritance

Generalization

<table>
<thead>
<tr>
<th>a1</th>
<th>a2</th>
<th>a3</th>
</tr>
</thead>
<tbody>
<tr>
<td>o1</td>
<td>o2</td>
<td>o3</td>
</tr>
</tbody>
</table>

Specialization

<table>
<thead>
<tr>
<th>a4</th>
<th>a5</th>
<th>a6</th>
</tr>
</thead>
<tbody>
<tr>
<td>o4</td>
<td>o5</td>
<td>o6</td>
</tr>
</tbody>
</table>

One-Way Inheritance from the Generalization to the Specialization

(a = attribute; o = operation)
Generalization - Multiple Inheritance

Generalization1
- a1
- a2
- a3
- o3
- o4
- o5

Specialization
- a6
- a7
- a8
inherited attributes
- o6
- o7
- o8
inherited operations

Generalization2
- a2
- a4
- a5
- o1
- o2
- o3

(o1) which one? o3 which one? o4 o5

a1 (which one?) a3
a4 a5

unified modeling language
UML Generalization Notation

Note: Supertype = Superclass; Subtype = Subclass
Generalization - Multiple Classification

Diagram:
- Female
- Male
- <abstract> Person
- Gender {complete}
- Patient
- Discriminator
- Doctor
- Nurse
- Physical-therapist
- role
- patient

Explanation:
This diagram illustrates the generalization of roles in a health care setting. The roles Doctor, Nurse, and Physical-therapist are specific instances of the abstract role Person. The gender discriminator is used to distinguish between male and female individuals, which is a complete attribute for the Person class.
Associations

- Relationships between instances (objects) of classes

- Conceptual:
  - associations can have two roles (bi-directional):
    - source --> target
    - target --> source
  - roles have multiplicity (e.g., cardinality, constraints)
  - To restrict navigation to one direction only, an arrowhead is used to indicate the navigation direction

- No inheritance as in generalizations
Object Association Relationship Patterns

a) Object Associations

b) Object Aggregation Associations

(\(y\) may not be “1”)

c) Object Composition Associations

(y may not be “1”)

Class A
attributes
operations

Class B
attributes
operations

Whole
attributes
operations

Part1
attributes
operations

Part2
attributes
operations

PartN
attributes
operations

1

1

y

y

1
Associations

Example:

Company \(\stackrel{\text{Employer}}{\longrightarrow}\) Employee \(\stackrel{\text{Person}}{\longrightarrow}\)
Multiplicities

1

Class

exactly one

0..*

Class

many (zero or more)

0..1

Class

optional (zero or one)

m..n

Class

numerically specified

Example:

Course 1 0..* CourseOffering
Aggregation & Composition

• Aggregation (shared aggregation):
  • is a specialized form of ASSOCIATION in which a whole is related to its part(s).
  • is known as a “part of” or containment relationship and follows the “has a” heuristic
  • three ways to think about aggregations:
    • whole-parts
    • container-contents
    • group-members

• Composition (composite aggregation):
  • is a stronger version of AGGREGATION
  • the “part(s)” may belong to only ONE whole
  • the part(s) are usually expected to “live” and “die” with the whole (“cascading delete”)

• Aggregation vs. Composition vs. Association???
**Aggregation**

- Faculty
- CourseTeaching: 1..* (team-teaching is possible)

(Another: assembly --> part)

**Composition**

- SalesOrder
- SalesOrderLineItem: 1..* (another: hand --> finger)
Composition

Composition is often used in place of Generalization (inheritance) to avoid “transmuting” (adding, copying, and deleting of objects)

Note: Attributes may need to be considered to more-fully understand
Association, Aggregation and Composition

Template/Pattern

- Whole
- 0..*
- Part
- 0..*

Example

(association, aggregation & composition look the same)
Multiplicity Example #1

- One Whole is associated with 5 Part1
- One Part1 is associated with 1 Whole

- One Whole is associated with 2 PartN
- One PartN is associated with 3 Whole
Multiplicity Example #2

Class 1

min. 1..n max.

Class 2

1

Class 3

2..5

0..*

etc...

min. 1..n

etc...

max. 2..5

etc...
Multiplicity Example #3

FacultyInformation

DegreeHeld

CommitteeAssign

CourseTeach

StudentInformation

CourseCompleted

ClubMember
“many-to-many” multiplicity

Attributes that represent the “union” of the two classes are located in this “association” class.
Reflexive Association Relationships

Objects within the same class have a relationship with each other.
Figure 3.10a Video Store UML Class Diagram with Attributes & Operations
Figure 3.10b Video Store UML Class Diagram with Attributes & Operations
Figure 3.10c Video Store UML Class Diagram with Attributes & Operations
Now, apply the concepts in Java