Class Diagrams

- Basic Concepts
- Type/Class
- Attribute
- Association
- Association Role
- Multiplicity
- Navigability
- Aggregation and Composition
- Association Class
- n-Ary Associations
- Generalization
Basic Concepts

• This is the main “internal” view of the system
  - The focus is on static structure, things about the
    business policy/business process that should
    (effectively) always be true
    * A formal methods person would call these
      “invariants”

• The main concepts in class models are
  - Type/Class
  - Attribute
  - Association
  - Generalization
Type/Class

- A Class is the descriptor for a set of objects with similar structure, behavior, and relationships
  - “Class” tends to be interpreted as a design and implementation concept while “Type” tends to be interpreted as an analysis concept

- Notation(s)

<table>
<thead>
<tr>
<th>Purchase Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

  Default attribute compartment

  Default operation compartment

<table>
<thead>
<tr>
<th>Purchase Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>operations</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>responsibilities</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>exceptions</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

- Think of a type or class as representing a stencil from which the actual objects are created
Type/Class (cont)

• The following kinds of real-world things tend to end up being modeled as classes
  - Physical things
    * e.g., Airplane, Aardvark, Cyclotron
  - Important concepts
    * e.g., Account, Purchase Order
  - Roles (uses) of physical things or concepts
    * e.g., Customer, Branch Manager, Active Runway
  - Incidents (significant happenings)
    * e.g., Trouble Report, Milestone
  - Characteristics
    * e.g., Airplane-type, Computer-model, Recipe
  - Associations
    * e.g., Enrollment, License
  - ...

• Include each of the actors from the use cases on the class diagram
  - Suggestion: stereotype them as <<actor>>
Attribute

• An Attribute is a single, named fragment of the persistent state

• Notation

<table>
<thead>
<tr>
<th>Purchase Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date Ordered</td>
</tr>
<tr>
<td>Ship Via</td>
</tr>
<tr>
<td>Ship To</td>
</tr>
<tr>
<td>Status</td>
</tr>
</tbody>
</table>
Association

• An Association exists when some number of objects are connected (linked) in some significant manner and it is necessary to remember that connection
  - In set theory terms, each class corresponds to a set of things, and an association corresponds to a mapping between members of these sets

  Aardvark Lives In Burrow

• Associations permit dynamics of the form
  Given some member of set A, which member or members of set B is it linked with?

  Given an aardvark, which burrow does it live in?
  Given a burrow, which aardvark lives in it?

• Basic notation
<table>
<thead>
<tr>
<th>Aardvark</th>
<th>Burrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Location</td>
</tr>
<tr>
<td>Weight</td>
<td>Depth</td>
</tr>
<tr>
<td>Age</td>
<td>Number of Chambers</td>
</tr>
<tr>
<td>Sex</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Weight</th>
<th>Age</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aardvark</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Association Role

- An Association Role indicates the role (purpose) that the class plays in the association

- Notation

<table>
<thead>
<tr>
<th>Aardvark</th>
<th>Burrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Location</td>
</tr>
<tr>
<td>Weight</td>
<td>Depth</td>
</tr>
<tr>
<td>Age</td>
<td>Number of Chambers</td>
</tr>
<tr>
<td>Sex</td>
<td></td>
</tr>
</tbody>
</table>

- Roles can be explicitly named, but they don’t have to be
  - If not explicitly named, the role “defaults” to the name of the target class
Multiplicity

• Describes the rules about the number of objects in each class that may or may not be linked

• “Anchor point analysis” [Flavin81] is a useful tactic to help you determine multiplicity. It is based on the question “Assuming that I am an object of this class, what is the minimum and maximum number of objects in that class can I be linked with?”

The question is repeated from the point of view (“anchored at”) each of the participating classes

• Example
  
  If I were an Aardvark, what is the minimum and maximum number of Burrows I could live in?
  If I were a Burrow, what is the minimum and maximum number of Aardvarks that could live in me?

• Notation

<table>
<thead>
<tr>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0..1</td>
</tr>
<tr>
<td>0</td>
<td>Many</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Many</td>
<td>1..*</td>
</tr>
</tbody>
</table>

[Flavin81]
Multiplicity (cont)

• The following are examples of the various forms of multiplicity

  On-Duty Crew IS ASSIGNED TO Ship at Sea  (1:1)
  Operating Power Supply POWERS Amplifier   (0..1:1)
  Batch of Chocolate IS CONTAINED IN Vat     (0..1:0..1)
    Slum Lord OWNS Apartment      (1:1..*)
  Ada Variable HAS DEFINED Data Type     (1:*)
  Secretary IS ASSIGNED TO Manager       (0..1:*)
  Student IS ENROLLED IN Course          (1..*:1..*)
  Clubmember IS MEMBER OF Committee      (1..*:*)
  Traveler HAS RESERVATION ON Scheduled Flight (*:**)

• Multiplicity may be even further constrained by the business policy/business process

  Commercial Pilot HOLDS RATING IN Airplane-Type  (*:0..2)

  - Document the policy-constrained limits in the model
Navigability

- In some cases, an association is constrained to be traversed in certain directions and it may be useful to include this information in the model
  - Uni-directional associations can be traversed in a single direction only
  - Bi-directional association can be traversed in either direction

- Notation

<table>
<thead>
<tr>
<th>Pascal Variable</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name Stack Offset</td>
<td>Type Name Size</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- If navigability is not specified, it is safest to assume that it is “undecided”
Aggregation and Composition

- Aggregation is a kind of association that represents a whole-part relationship
  - The implication is that if the whole is created or deleted, the parts must also be created or deleted at the same time

- Notation for Aggregation

![Diagram of Flight Plan and Waypoint](image1)

- Composition is like aggregation with the added constraint that the part may be the part of exactly one whole

![Diagram of Automobile, Engine, and Tire](image2)
• It might help to think of aggregation and composition as “patterns” for an association and its behavior
Association Class

• Associations may take on the characteristics of a class
  - Associations may have attributes
  - Associations may have important states and actions

• Notation

![Diagram showing the relationship between Flight, Traveller, and Reservation classes with attributes like Flight Number, From City Code, Departure Time, To City Code, Arrival Time, Name, Address, Smoker?, Date, Class of Service, Status, and ...]

• Even though in the notation it appears as an association and a class separately, semantically it is a single element in the model
n-Ary Associations

• So far, we have only seen associations between two classes
  - This is the usual case, but some associations involve more than two classes

• Notation

![Diagram of n-Ary Associations]
Generalization

• Generalization is a way of saying
  At one level of abstraction, the objects in this
class are all the same but at a finer level of
detail, they may be different
  - Represents an is-a or is-a-kind-of relationship

• Notation
Generalization (cont)

- The subclasses may represent a complete or incomplete breakdown of the superset (are all subsets represented?)
  - Use the constraint \{complete\} or \{incomplete\}

- The subsets may represent (non-)mutually exclusive groupings
  - Use the constraint \{disjoint\} or \{overlapping\}

- Multiple and Dynamic Classification
  - Multiple classification means that the same superset can be specialized in multiple dimensions
  - Dynamic classification means that an instance of a superset can change from one subset to another

![Diagram of Generalization in UML](image)
Key Points

• The class diagram is the main “internal” view
  - The focus is on static structure, things about the business policy/business process that should (effectively) always be true

• A Class is the descriptor for a set of objects with similar structure, behavior, and relationships

• An Attribute is a single, named fragment of the persistent state

• An Association exists when some number of objects are connected (linked) in some significant manner and it is necessary to remember that connection

• An Association Role indicates the role (purpose) that the class plays in the association

• Multiplicity describes the rules about the number of objects in each class that may or may not be linked

• Aggregation is a kind of association that represents a whole-part relationship
  - Composition is like aggregation with the added constraint that the part may be the part of exactly one whole

• Associations may take on the characteristics of a class
• Generalization is a way of saying, “At one level of abstraction, the objects in this class are all the same but at a finer level of detail, they may be different”