Conceptual Model

• A representation of concepts in a problem domain.

• Drawn with a set of static structure where
  – concepts are associated;
  – concepts have attributes;
  – concepts has no operations. [Operations (responsibilities) are added later when concepts are implemented as classes and collaboration among classes are explored.]
Class diagram perspectives

• Conceptual.
  – Representing concepts in the domain of under study
  – Drawn with little or no regard for implementation

• Specification.
  – Type or interface, rather than the class that implements it.

• Implementation.
  – Class.
Language support

• In C++, type, interface and class are represented by the same language construct: *class*.

• In Java, type and interface are represented by *interface*. Ideally, class should be reserved for implementation perspective.

• Regardless of language support, it is still important to distinguish among the three different perspectives. [Folwer, GoF]
Decomposition

• Structured analysis: decomposition by functions
• Object-oriented analysis: decomposition by concepts.
Steps To Build A Conceptual Model

- Identify concepts
  - obtain concepts from use case and requirement documents

- Identify and add relationships to concepts
  - How are concepts related?

- Add attributes
  - what are the attributes of the concepts?
Strategies for Identifying Concepts

• Scan the nouns and noun phrases from
  – the use case and
  – requirement documents.

• Use a domain list of concepts (concept categories list)
  – built upon the analyst’s experience.

• We want to extract as many concepts as possible
Expanded use case: Take a Test

**Name:** Take a Test

**Actors:** Student

**Overview:** Student logins to a test station to take a test. A number of problems are given; upon completion, test grade is shown to the student.

**Typical course of events:**

<table>
<thead>
<tr>
<th>Actor action</th>
<th>System response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a Student logs in to a test station.</td>
<td>2. Checks account and prints welcome message and presents test menu.</td>
</tr>
<tr>
<td>3. The student makes a test selection.</td>
<td>4. Generate test items according to a predetermined strategy.</td>
</tr>
<tr>
<td>5. The student composes answers to problems until all done or time’s up.</td>
<td>6. Grades the test and presents test grades to the student. Test grade is logged for the student.</td>
</tr>
<tr>
<td>7. The student logs out or take another test.</td>
<td></td>
</tr>
</tbody>
</table>
Conceptual Model Diagram

- Student
- Account
- Grade
- Test
- TestMenu
- TestTime
- TestStation
- TestItem
- Answer
- Strategy
Conceptual Model: Associations

• An association is a relationship between concepts that indicates some meaningful connection

• In UML, an association is defined as “structural relationships between objects of different types.”
Associations (I)

- Two concepts are associated (*permanently linked*) when one needs to know the other during its lifetime in a business process. Note that instance of the concept may change over time.
- Association is represented as a solid line between two concepts.
- Association is directional.
- Model transient knowledge (relationship that lasts for a relatively short duration of time, e.g., parameter reference) between two concepts by *dependencies.*
Associations (II)

- Associations can be named to describe nature of the association.
- Each of the two concepts plays a role in the association.
- Roles can have multiplicity.
Multiplicity

Figure 10.4 Multiplicity values.
Strategies for finding associations

• Common association list
  – Physical whole/part
  – Logical whole/part
  – Description, record.

• Need-to-know associations
  – Concept A needs concept B to support it
## Common association list (I)

<table>
<thead>
<tr>
<th>Category</th>
<th>POST System</th>
</tr>
</thead>
<tbody>
<tr>
<td>A is a physical part of B</td>
<td>not applicable</td>
</tr>
<tr>
<td>A is a logical part of B</td>
<td>SalesLineItem—Sale</td>
</tr>
<tr>
<td>A is physically contained in/on B</td>
<td>POST—Store</td>
</tr>
<tr>
<td></td>
<td>Item—Store</td>
</tr>
<tr>
<td>A is logically contained in B</td>
<td>ProductSpecification—Product-Catalog</td>
</tr>
<tr>
<td></td>
<td>ProductCatalog—Store</td>
</tr>
<tr>
<td>A is a description for B</td>
<td>ProductSpecification—Item</td>
</tr>
<tr>
<td>A is a line item of a transaction or report B</td>
<td>SalesLineItem—Sale</td>
</tr>
<tr>
<td>A is logged/recorded/reported/captured in B</td>
<td>(completed) Sales—Store</td>
</tr>
<tr>
<td></td>
<td>(current) Sale—POST</td>
</tr>
<tr>
<td>A is a member of B</td>
<td>Cashier—Store</td>
</tr>
<tr>
<td>A is an organizational subunit of B</td>
<td>not applicable</td>
</tr>
<tr>
<td>A uses or manages B</td>
<td>Cashier—POST</td>
</tr>
<tr>
<td></td>
<td>Manager—POST</td>
</tr>
<tr>
<td></td>
<td>Manager—Cashier, but probably not applicable.</td>
</tr>
<tr>
<td>A communicates with B</td>
<td>Customer—Cashier</td>
</tr>
</tbody>
</table>
Common association list (II)

<table>
<thead>
<tr>
<th>Category</th>
<th>POST System</th>
</tr>
</thead>
<tbody>
<tr>
<td>A is related to a transaction B</td>
<td>Customer—Payment</td>
</tr>
<tr>
<td></td>
<td>Cashier—Payment</td>
</tr>
<tr>
<td>A is a transaction related to another transaction B</td>
<td>Payment—Sale</td>
</tr>
<tr>
<td>A is next to B</td>
<td>POST—POST, but probably not applicable</td>
</tr>
<tr>
<td>A is owned by B</td>
<td>POST—Store</td>
</tr>
</tbody>
</table>
Automatic Test System
Three perspectives of association

- Conceptual: relationships between concepts.
- Specification: represents responsibilities.
- Implementation: there are pointers (references) in both direction between the related classes.
Specification perspective

• TestItem and Answer are associated implies:
  – A TestItem should have one or more methods which tell you what its Answer(s) is (are).
  – Similarly, an Answer might have methods to tell you to which TestItem it is applicable.
  – If the association is from TestItem to Answer, add navigability arrow to show so.
  – Lacking arrows: navigability undecided yet.
Attributes

• Things that need to be remembered for a concept in a conceptual model.
• Attributes should be simple
  – Boolean, Number, String, Time, ID, ZIP code
• If an attribute gets to complex, or if it acts as a foreign key, consider representing it as a concept that is associated with the present concept.
Figure 10.7 A point-of-sale conceptual model.
Further Readings

• Larman, Chapters 9-11
• Jacobson, Section 3.4
• Fowler/Distilled, Chapter 4.
• GoF, Chapter 1, esp. pp.13-18