UML Specification by Example: Objects/Data

• With use cases defined, the next specification to look at is the meat and potatoes of the rest of this course — the data model

• In pre-object-oriented days, the conceptual schema was modeled using entity-relationship (E-R) notation

• Today’s tools and languages allow for a conceptual schema that maps pretty well to an object-oriented code base, which with just a little more work can translate into the relational data model in a straightforward manner

• Just so you don’t have to dig up the previous handout again, here is our natural-language write-up for our sample application:

We would like to create a student information and document management system, to augment the student records that are already maintained by the university. With this system, a user can maintain a list of student records. Linked to each student record would be a set of documents. Each document would have a timestamp, and can be bound to any number of keywords, which are also defined in a separate list. The actual document files (such as PDF) can be uploaded to the system and stored on a server; they can then be retrieved by (at least) student, date, and/or keyword. The keyword list can be added to as needed, and functions for adding new students and maintaining existing student data would also be required. The preferred end-user interface would be the Web, although the overall architecture may also accommodate other UIs such as Swing.
Structural Modeling Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Linguistic Role</th>
<th>E-R Term</th>
<th>UML Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>real-world items stored by the system</td>
<td>nouns</td>
<td>entity set</td>
<td>entity</td>
</tr>
<tr>
<td>connections between items</td>
<td>verbs</td>
<td>relationship set</td>
<td>relationship</td>
</tr>
<tr>
<td>properties that describe items</td>
<td>adjectives</td>
<td>attribute</td>
<td>attribute</td>
</tr>
<tr>
<td>properties that describe connections</td>
<td>adverbs</td>
<td>descriptive attribute</td>
<td>relationship</td>
</tr>
<tr>
<td>distinguishing/unique properties</td>
<td>proper nouns</td>
<td>key</td>
<td>object identifier</td>
</tr>
</tbody>
</table>

- Nouns and verbs have two “levels” of existence: as the category to which they belong (entity set; class) and as a specific occurrence of that category (entity; object)

- Generalization/specialization: entities may indicate that they are specializations or subclasses of other entities

- Containment: entities may “hold” other entities through aggregation or composition relationships

- Cardinality: relationships/associations typically indicate how many entities participate in them — 1-to-1, 1-to-many, many-to-1, many-to-many

- Composite/multivalued attributes: attributes may be broken down into sub-attributes or may hold multiple values in the same “slot”
E-R vs. UML

- E-R (1976) predates UML (version 1.1 in 1997)
- “Pure” E-R has been extended in different ways; collectively these variants are called “extended E-R”
- UML covers broader ground than E-R, covering structure (class, use case, component diagrams) and behavior (sequence/collaboration, state, activity diagrams)
- UML class notation is a direct descendant of E-R and extended E-R notations, integrating object-oriented constructs that weren’t around when E-R was born

Structural Diagram Notation

- Just as with use cases, your data model diagram is the means, not the end
- Unlike use cases, however, a data model is really primarily about structure, and so a diagram is definitely more compelling than a text write-up — but, it still helps in some situations
- These days, structural diagrams generally have two concrete destinations — an object-oriented environment at runtime (e.g. Java), and a relational database for persistence (e.g. PostgreSQL)
Depending on the size of the project and/or its development team, these can happen next:

- **Dive into further detail:**
  - Add more information to the data model, such as more specific attribute information (types, default values, constraints), defining methods, etc.
  - Create the other types of diagrams within UML's scope: component, behavioral diagrams

- **On the database front, the conceptual model would need to be implemented in terms of the target database's logical model**
  - When the conceptual model is implemented at runtime in an object-oriented environment (e.g., Java) and is persisted in a relational database, this process goes by the specific term “object-relational (OR) mapping”