

## Software Engineering

### Third Class

#### Lecture 6

### Analysis Modeling

At a technical level, software engineering begins with a series of modeling tasks that lead to a complete specification of requirements and a comprehensive design representation for the software to be built. The *analysis model*, actually a set of models, is the first technical representation of a system. Over the years many methods have been proposed for analysis modeling.

However, two now dominate. The first, *structured analysis*, is a classical modeling method. The other approach, *object oriented analysis*.

Structured analysis is a model building activity. Applying the operational analysis principles we create and partition data, functional, and behavioral models that depict the essence of what must built.

#### **Note:**

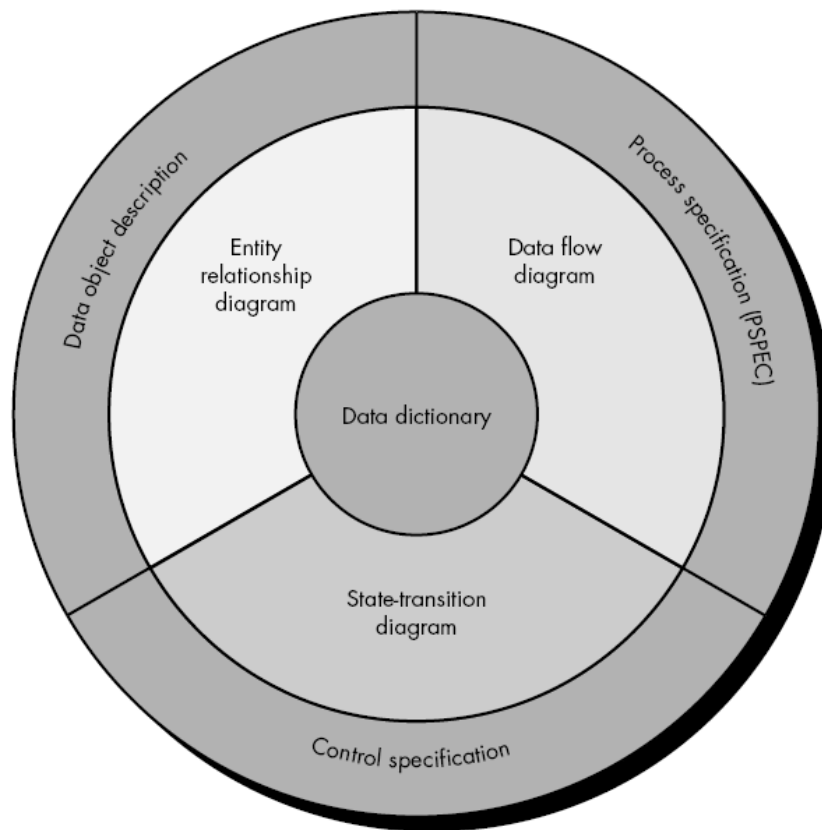
The written word is a wonderful vehicle for communication, but it is not necessarily the best way to represent the requirements for computer software. **Analysis modeling** uses a combination of text and diagrammatic forms to depict requirements for data, function, and behavior in a way that is relatively easy to understand, and more important, straightforward to review for correctness, completeness, and consistency.

#### **1 The Elements of the Analysis Model**

The analysis model must achieve three primary objectives:

1. To describe what the customer requires.
2. To establish a basis for the creation of a software design.
3. To define a set of requirements that can be validated once the software is built.

To accomplish these objectives, the analysis model derived during structured analysis takes the form illustrated in Figure 1.



**Figure 1** The structure of the analysis model

- At the core of the model lies the *data dictionary*—a repository that contains descriptions of all data objects consumed or produced by the software.
- Three different diagrams surround the core.
- The *entity relation diagram* (ERD) depicts relationships between data objects. The ERD is the notation that is used to conduct the data modeling activity. The attributes of each data object noted in the ERD can be described using a data object description.
- The *data flow diagram* (DFD) serves two purposes:
  1. To provide an indication of how data are transformed as they move through the system.
  2. To depict the functions (and subfunctions) that transform the data flow.

The DFD provides additional information that is used during the analysis of the information domain and serves as a basis for the modeling of function. A description of each function presented in the DFD is contained in a *process specification* (PSPEC).

- The *state transition diagram* (STD) indicates how the system behaves as a consequence of external events. To accomplish this, the STD represents the various modes of behavior (called *states*) of the system and the manner in which transitions are made from state to state. The STD serves as the basis for behavioral modeling.

Additional information about the control aspects of the software is contained in the *control specification* (CSPEC).

The analysis model encompasses each of the diagrams, specifications, descriptions, and the dictionary noted in Figure 1.

## **2 Data Modeling**

Data modeling methods make use of the entity relationship diagram. The ERD enables a software engineer to identify data objects and their relationships using a graphical notation.

In the context of structured analysis, the ERD defines all data that are entered, stored, transformed, and produced within an application.

The entity relationship diagram focuses solely on data (and therefore satisfies the first operational analysis principles), representing a "data network" that exists for a given system. The ERD is especially useful for applications in which data and the relationships that govern data are complex. Unlike the data flow diagram, data modeling considers data independent of the processing that transforms the data.

### **2.1 Data Objects, Attributes, and Relationships**

The data model consists of three interrelated pieces of information: the data object, the attributes that describe the data object, and the relationships that connect data objects to one another.

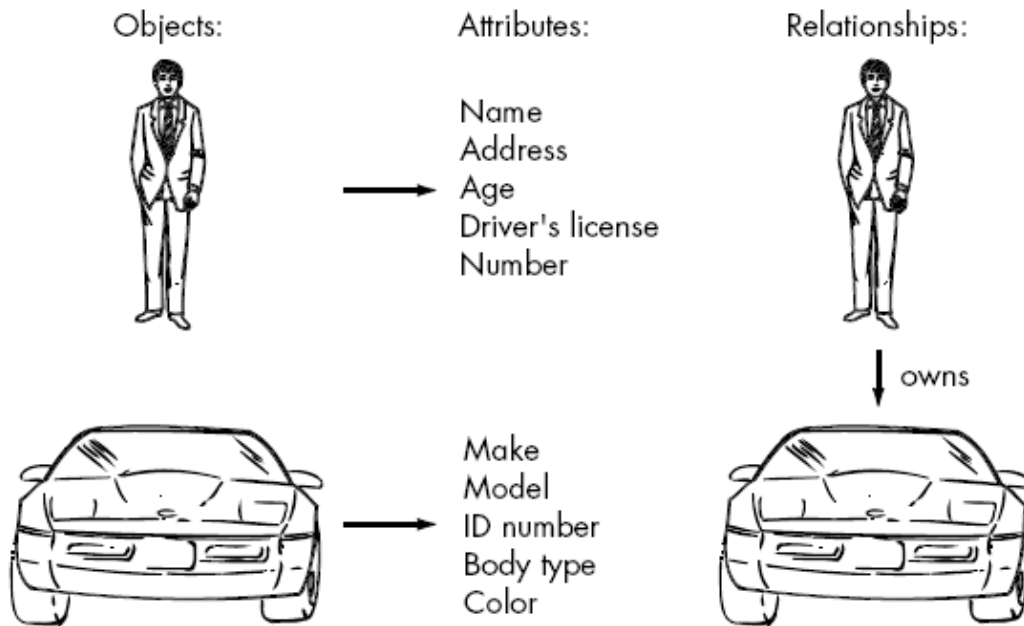
**Data objects.** A *data object* is a representation of almost any composite information that must be understood by software. By *composite information*, we mean something that has a number of different properties or attributes. Therefore, width (a single value) would not be a valid data object, but dimensions (incorporating height, width, and depth) could be defined as an object.

A data object can be an external entity (e.g., anything that produces or consumes information), a thing (e.g., a report or a display), an occurrence (e.g., a telephone call)

or event (e.g., an alarm), a role (e.g., salesperson), an organizational unit (e.g., accounting department), a place (e.g., a warehouse), or a structure (e.g., a file).

**EX:**

A person or a car (Figure 2) can be viewed as a data object in the sense that either can be defined in terms of a set of attributes. The data object description incorporates the data object and all of its attributes.

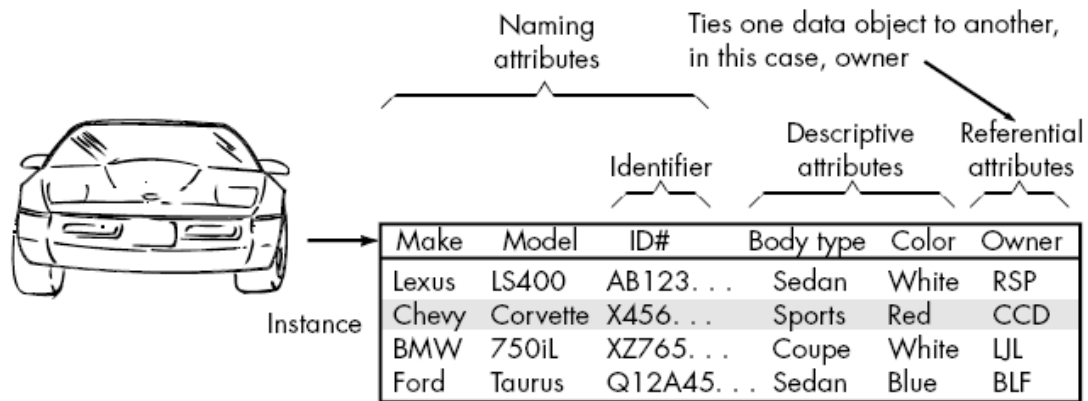


**Figure 2** Data objects, attributes and relationships

Data objects (represented in bold) are related to one another. For example, **person** can *own* **car**, where the relationship *own* connotes a specific "connection" between **person** and **car**. The relationships are always defined by the context of the problem that is being analyzed.

A data object encapsulates data only—there is no reference within a data object to operations that act on the data (This distinction separates the data object from the *class* or *object* defined as part of the object-oriented paradigm). Therefore, the data object can be represented as a table as shown in Figure 3. The headings in the table reflect attributes of the object. In this case, a car is defined in terms of make, model, ID number, body type color and owner. The body of the table represents specific instances of the data object.

For example, a Chevy Corvette is an instance of the data object **car**.



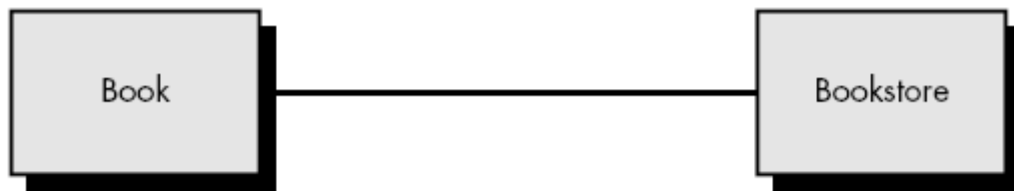
**Figure 3:** Tabular representation of data objects

**Attributes.** Attributes define the properties of a data object and take on one of three different characteristics. They can be used to (1) name an instance of the data object, (2) describe the instance, or (3) make reference to another instance in another table. In addition, one or more of the attributes must be defined as an *identifier*—that is, the identifier attribute becomes a "key" when we want to find an instance of the data object. In some cases, values for the identifier(s) are unique, although this is not a requirement. Referring to the data object car, a reasonable identifier might be the **ID** number. The set of attributes that is appropriate for a given data object is determined through an understanding of the problem context.

**Relationships.** Data objects are connected to one another in different ways. Consider two data objects, **book** and **bookstore**. These objects can be represented using the simple notation illustrated in Figure 4a. A connection is established between **book** and **bookstore** because the two objects are related. But what are the relationships? To determine the answer, we must understand the role of books and bookstores within the context of the software to be built. We can define a set of object/relationship pairs that define the relevant relationships. For example,

- A bookstore orders books.
- A bookstore displays books.
- A bookstore stocks books.
- A bookstore sells books.
- A bookstore returns books.

The relationships *orders*, *displays*, *stocks*, *sells*, and *returns* define the relevant connections between **book** and **bookstore**. Figure 12.4b illustrates these object/relationship pairs graphically.



(a) A basic connection between objects



(b) Relationships between objects

**Figure 4** Relationships

**Note:**

It is important to note that object/relationship pairs are bidirectional. That is, they can be read in either direction. A bookstore orders books or books are ordered by a bookstore

**2.2 Cardinality and Modality**

The elements of data modeling—data objects, attributes, and relationships— provide the basis for understanding the information domain of a problem. However, additional information related to these basic elements must also be understood.

We have defined a set of objects and represented the object/relationship pairs that bind them. But a simple pair that states: **object X** *relates* to **object Y** does not provide enough information for software engineering purposes. We must understand how

many occurrences of **object X** are related to how many occurrences of **object Y**. This leads to a data modeling concept called *cardinality*.

**Cardinality.** The data model must be capable of representing the number of occurrences objects in a given relationship. Tillmann defines the *cardinality* of an object/relationship pair in the following manner:

Cardinality is the specification of the number of occurrences of one [object] that can be related to the number of occurrences of another [object].

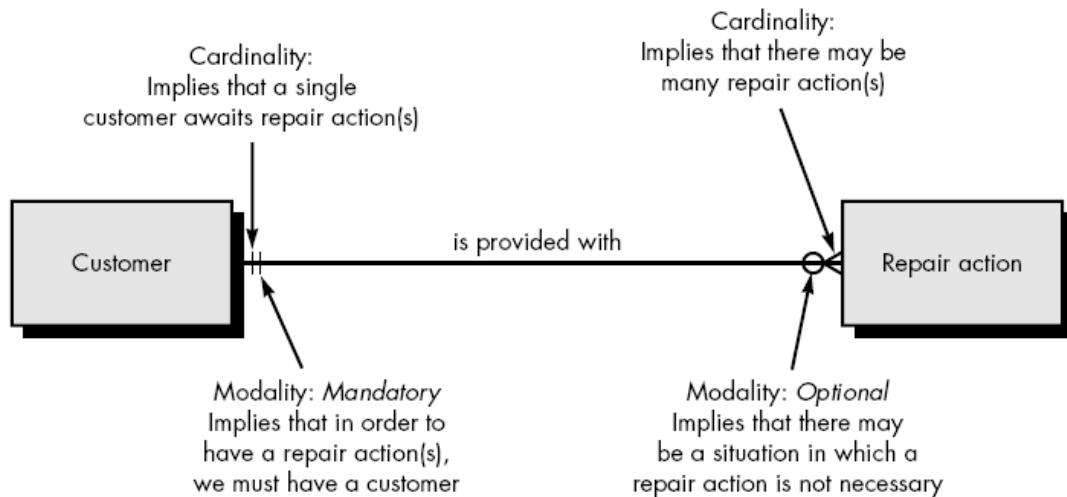
Cardinality is usually expressed as simply 'one' or 'many.'

Taking into consideration all combinations of 'one' and 'many,' two [objects] can be related as

- One-to-one (1:1)—An occurrence of [object] 'A' can relate to one and only one occurrence of [object] 'B,' and an occurrence of 'B' can relate to only one occurrence of 'A.'
- One-to-many (1:N)—One occurrence of [object] 'A' can relate to one or many occurrences of [object] 'B,' but an occurrence of 'B' can relate to only one occurrence of 'A.'
- Many-to-many (M:N)—An occurrence of [object] 'A' can relate to one or more occurrences of 'B,' while an occurrence of 'B' can relate to one or more occurrences of 'A.'

Cardinality defines “the maximum number of objects that can participate in a relationship”. It does not, however, provide an indication of whether or not a particular data object must participate in the relationship. To specify this information, the data model adds modality to the object/relationship pair.

**Modality.** The *modality* of a relationship is 0 if there is no explicit need for the relationship to occur or the relationship is optional. The modality is 1 if an occurrence of the relationship is mandatory. To illustrate, consider software that is used by a local telephone company to process requests for field service. A customer indicates that there is a problem. If the problem is diagnosed as relatively simple, a single repair action occurs. However, if the problem is complex, multiple repair actions may be required. Figure 5 illustrates the relationship, cardinality, and modality between the data objects **customer** and **repair action**.



**Figure 5** Cardinality and modality

Referring to the figure, a one to many cardinality relationship is established. That is, a single customer can be provided with zero or many repair actions. The symbols on the relationship connection closest to the data object rectangles indicate cardinality.

The vertical bar indicates one and the three-pronged fork indicates many.

Modality is indicated by the symbols that are further away from the data object rectangles. The second vertical bar on the left indicates that there must be a customer for a repair action to occur. The circle on the right indicates that there may be no repair action required for the type of problem reported by the customer.

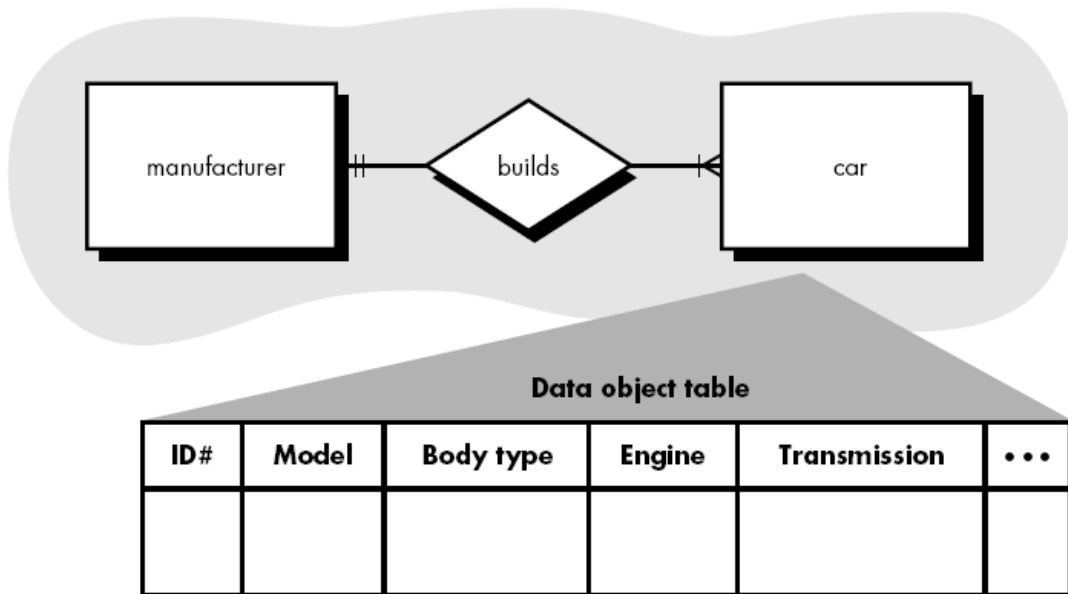
### 2.3 Entity/Relationship Diagrams

The object/relationship pair (discussed in Section 2.1) is the cornerstone of the data model. These pairs can be represented graphically using the *entity/relationship diagram*. The ERD was originally proposed by Peter Chen for the design of relational database systems and has been extended by others. A set of primary components are identified for the ERD: data objects, attributes, relationships, and various type indicators. The primary purpose of the ERD is to represent data objects and their relationships.

Rudimentary ERD notation has already been introduced in Section 2. Data objects are represented by a labeled rectangle. Relationships are indicated with a labeled line connecting objects. In some variations of the ERD, the connecting line contains a diamond that is labeled with the relationship. Connections between data objects and

relationships are established using a variety of special symbols that indicate cardinality and modality (Section 2.2).

The relationship between the data objects **car** and **manufacturer** would be represented as shown in Figure 6. One manufacturer builds one or many cars. Given the context implied by the ERD, the specification of the data object **car** (data object table in Figure 6) would be radically different from the earlier specification (Figure 3). By examining the symbols at the end of the connection line between objects, it can be seen that the modality of both occurrences is mandatory (the vertical lines).



**FIGURE 12.6** A simple ERD and data object table (Note: In this ERD the relationship *builds* is indicated by a diamond)