# *Analysis Modeling*

The analysis model is the first technical representation of a system.

*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.*

A software engineer (sometimes called an analyst) builds the model using requirements provided by the customer.
Software engineering begins with a series of modeling task that lead to a complete requirement specification and comprehensive design representation for the software to be built.
Two types of analysis model are commonly used:

- Structured Analysis
- Object-Oriented Analysis

**What are the steps?**

- *Data, functional, and behavioral requirements* are modeled using a number of different diagrammatic formats.
- *Data modeling defines data objects, attributes, and relationships.*
- *Functional modeling indicates how data are transformed within a system.*
- *Behavioral modeling depicts the impact of events.*
- Once preliminary models are created, they are refined and analyzed to assess their clarity, completeness, and consistency. A specification incorporating the model is created and then validated by both software engineers and customers/users.

➢ *Data object descriptions, entity relationship diagrams, data flow diagrams, state transition diagrams, process specifications, and control specifications are created as part of the analysis modeling activity.*

Analysis work products must be reviewed for completeness, correctness and consistency.

*************************************************************

## 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, and
(3) To define a set of requirements that can be validated once the software is built.

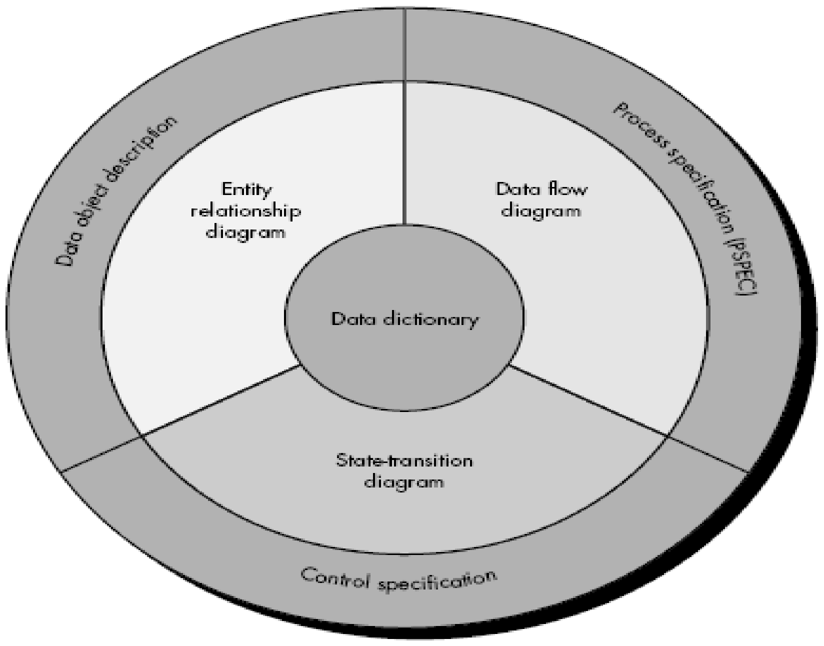


**FIGURE 12.1**
The structure of the analysis model

To accomplish these objectives, the analysis model derived during structured analysis takes the form illustrated in Figure 12.1.
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.
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 sub functions) 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).

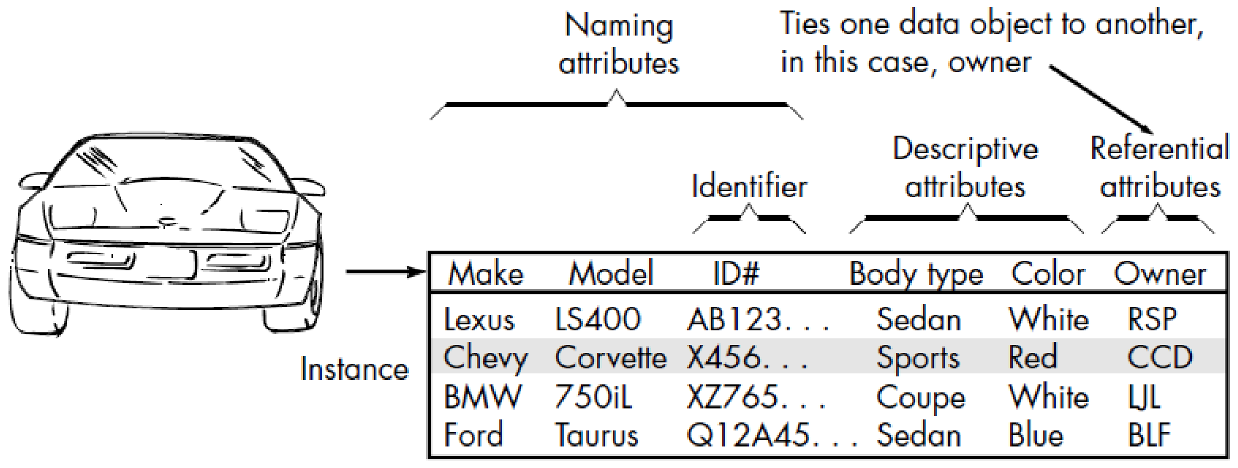
**The** ***State Transition Diagram*** **(STD)** indicates how the system behaves as a consequence of external events. The STD serves as the basis for behavioral modeling.
Additional information about the control aspects of the software is contained in the *control specification* (CSPEC).

## DATA MODELING

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 modeling methods make the use of *Entity Relationship Diagram (ERD).*
*ERD enables software engineers to identify data objects and their relationship* using a graphical notation.
ERD defines all the data that are entered, stored, transformed and produced within an application.
The ERD focuses solely in data, representing a "data network" that exists for a given system.

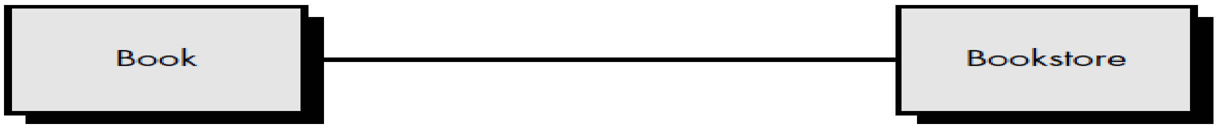
**Data objects:** A *data object* is a representation of almost any composite information that must be understood by software. By *composite information,* we mean something that has a number of different properties or attributes.
**Attributes:** Attributes define the properties of a data object and take 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.



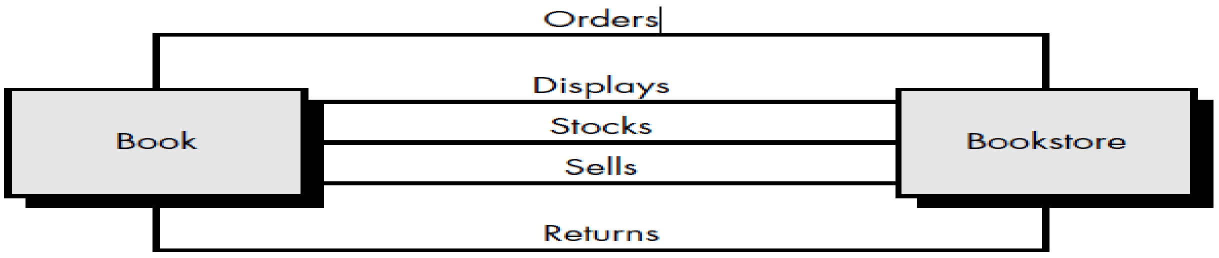
| Make | Model | ID# | Body type | Color | Owner |
|---|---|---|---|---|---|
| Lexus | LS400 | AB123. . . | Sedan | White | RSP |
| Chevy | Corvette | X456. . . | Sports | Red | CCD |
| BMW | 750iL | XZ765. . . | Coupe | White | LJL |
| Ford | Taurus | Q12A45. . . | Sedan | Blue | BLF |

**Relationships:** *indicate the manner in which data objects are one another.*
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.
A connection is established between **book** and **bookstore** because the two objects are related.



(a) A basic connection between objects

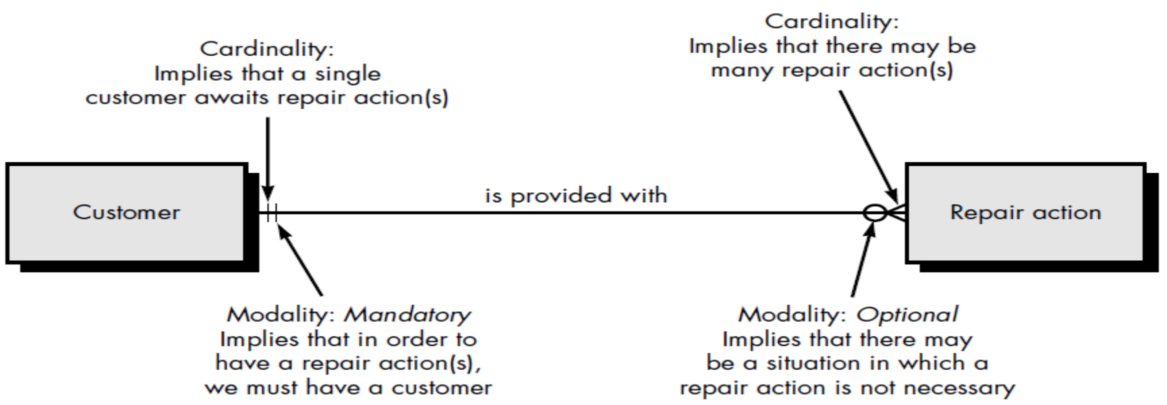


(b) Relationships between objects

## Cardinality and Modality

**Cardinality:** The data model must be capable of representing the number of occurrences objects in a given relationship.
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.' For example, a husband can have only one wife (in most cultures), while a parent can have many children.
Two [objects] can be related as
**One-to-one** (l:l)—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** (l: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.'

**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.


Cardinality:
Implies that a single
customer awaits repair action(s)
Cardinality:
Implies that there may be
many repair action(s)
Customer
is provided with
Repair action
Modality: *Mandatory*
Implies that in order to
have a repair action(s),
we must have a customer
Modality: *Optional*
Implies that there may
be a situation in which a
repair action is not necessary