Information Domain

- Software is built to accept the input, manipulate it on some way, and produce output.
- Software also process the event.
- An event represents some aspect of system control and is really nothing more than Boolean data – either on or off
- The information domain consists of three different views
- **Information content or data model** ◦
- shows the relationships among the data and control objects that make up the system
- **Information flow** ◦
- represents manner in which data and control objects change as each moves through system
- **Information structure** ◦
- representations of the internal organizations of various data and control items

Analysis Principles

1.The information domain of a problem must be represented and understood.

2.The **function** that the software is to be perform must be defined.

3. The behavior of the software must be represented

4. The model that depict information, function, and behavior must be partitioned in hierarchical fashion

5. The analysis process should move from essential information toward implementation details

Analysis Principles

- By applying these principles, the analysis principles, the analyst approaches a problem systematically .
- Information domain is examined so that the function may be understood more completely
- Models are used so that the characteristics of function and behavior can be communicated in a compact fashion.
- **Partitioning** is applied to reduce complexity

Analysis Principles

- In addition to these operational analysis, Davis suggests a set of guiding principles for requirement engineering
  - ***Understand the problem before you begin to create the analysis model.***
  - ***Develop the prototypes that enable a user to understand how human/machine interaction will occur***

Analysis Principles

- **Record the origin of and the reason for every requirement**
- **Use multiple views of the requirement**
- **Rank the requirements**
- **Work to eliminate ambiguity**

Information Domain

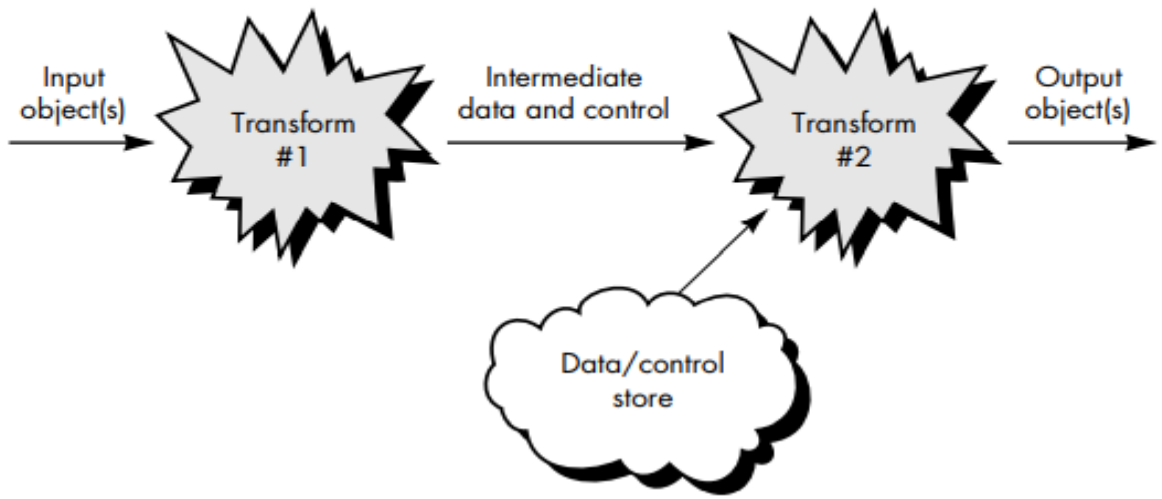

FIGURE 11.3 Information flow and transformation

Modeling

- **Data model** ◦
- shows relationships among system objects
- **Functional model** ◦
- software converts information and to accomplish this, it must perform at least three common tasks- input, processing and output.
- When functional models of an application are created, the software engineer emphasizes problem specific tasks.
- The functional model begins with a single reference level model (i.e., name of the software to be built). I
- n a series of iterations, more and more functional detail is given, until all system functionality is fully represented.

Modelling

Behavioral Modelling

- A computer program always exists in some state – an externally observable mode of behavior (e.g waiting, computing, printing) that is changed only when some events occurs.
- Describe manner in which software responds to events from the outside world

Partitioning

- Process that results in the elaboration of data, function, or behavior.
- **Horizontal partitioning** ◦
- breadth-first decomposition of the system function, behavior, or information, one level at a time.
- **Vertical partitioning** ◦
- depth-first elaboration of the system function, behavior, or information, one subsystem at a time.

Requirements Views

- **Essential view**
- presents the functions to be accomplished and the information to be processed while ignoring implementation
- **Implementation view**
- presents the real world realization of processing functions and information structures