**4.2.2 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. 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, while a parent can have many children. 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.' For example, a mother can have many children, but a child can have only one mother.
* 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.' For example, an uncle can have many nephews, while a nephew can have many uncles.

**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. Figure 4.4 illustrates the relationship, cardinality, and modality between the data objects customer and repair action.



Figure 4.4 Cardinality and modality

**4.2.3 Entity/Relationship Diagrams (ERD)**

The object/relationship pair is the cornerstone of the data model. These pairs can be represented graphically by using the entity/relationship diagram, figure 4.5 shows an expand ERD.



Figure 4.5 an expanded ERD

**4.3 Functional Modeling And Information Flow**

Information is transformed as it flows through a computer-based system. The system accepts input in a variety of forms; applies hardware, software, and human elements to transform it; and produces output in a variety of forms. Input may be a control signal transmitted by a transducer a series of numbers typed by a human operator, a packet of information transmitted on a network link, or a voluminous data-file retrieved from secondary storage. The transform(s) may comprise a signal logical comparison, a complex numerical algorithm, or a rule-inference approach of an expert system. Output may light a single LED or produce a 200 page report. In effect, we can create a flaw model for any computer based -system, regardless of size and complexity.

**4.3.1 Data Flow Diagrams**

As information moves through software, it is modified by a series of transformations; A data flow diagram is a graphical representation that depicts information flow and the transforms that are applied as data move from input to output. The basic form of a data flow diagram, also known as a data flow graph or a bubble chart, is illustrated in Figure 4.1.



Figure 4.6 Information Flow Model

A rectangle is used to represent an external entity, that is a system element (e.g., hardware, a person, another program) or another system that produces information for transformation by the software or receives information produced by the software. A circle (sometimes called a bubble) represents a process or transform that is applied to data (or control) and changes it in some way. An arrow represents one or more data items (data objects). All arrows on a data flow diagram should be labeled. The double line represents a data store—stored information that is used by the software. The data flow diagram may be used to represent a system or software at any level of abstraction. In fact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Therefore, the DFD provides a mechanism for functional modeling as well as information flow modeling.

A level 0 DFD also called a fundamental system model or a context model represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows, respectively additional processes (bubbles) and information flow paths are represented as the level 0 DFD is partitioned to reveal more detail. For example, a level 1 DFD might contain five or six bubbles with interconnecting arrows each of the processes represented at level 1 is a sub function of the overall system depicted in the context model. As we noted earlier, each of the bubbles may be refined or layered to depict more detail, Figure 4.7 illustrates this concept. A fundamental model for system F indicates the primary input is A and ultimate output is B. We refine the F model into transform f1 to f7. Note that information flow continuity must be maintained; that is, input and output to each refinement must remain the same. This concept, sometimes called balancing, is essential for the development of consistent models. Further refinement of f4 depicts detail in the form of transforms f41 to f45. Again, the input (X, y) and output (Z) remain unchanged.

DFD graphical notation must be augmented with descriptive text. A process specification (PSPEC) can be used to specify the processing details implied by a bubble within a DFD. The process specification describes the input to a function, the algorithm that is applied to transform the input, and the output that is produced in addition, the PSPEC indicates restrictions and limitations imposed on the process (function). Performance characteristics that are relevant to the process, and design constraints that may influence the way in which the Process will be implemented.



Figure 4.7 Information flow refinement