

# Chapter 3: Analysis

## A. System Requirements

### 1. Introduction:

Systems analysis is the part of the systems development life cycle in which you determine how the current information system functions and assess what users would like to see in a new system.

**Analysis has two sub phases: requirements determination and requirements structuring.**

Techniques used in requirements determination have evolved over time to become more structured and increasingly rely on computer support.

We will first study the more traditional requirements determination methods, including interviewing, observing users in their work environment, and collecting procedures and other written documents.

We will then discuss more current methods for collecting system requirements.

The first of these methods is Joint Application Design (JAD).

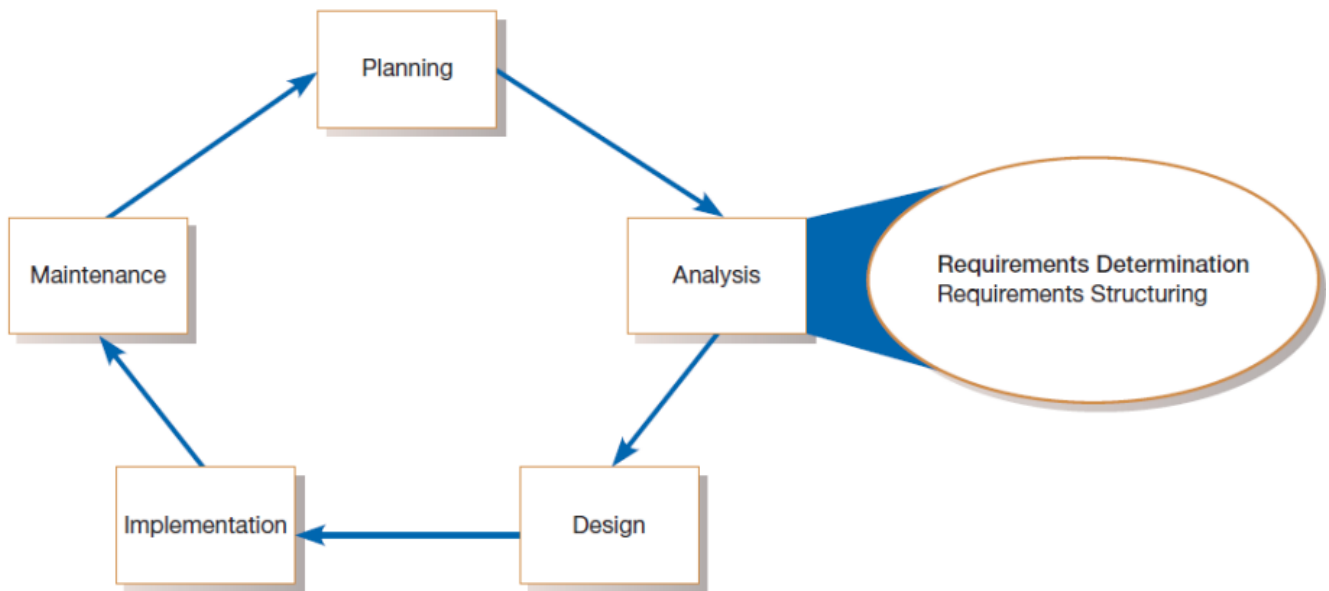
Next, we will read about how analysts rely more and more on information systems to help them perform analysis.

As you will see, CASE tools, discussed in Chapter 1, are useful in requirements determination, and prototyping has become a key tool for some requirements determination efforts.

Finally, you will learn how requirements analysis continues to be an important part of systems analysis and design, whether the approach involves business process redesign, new Agile techniques (such as constant user involvement or usage-centered design), or focuses on developing Internet applications.

### 2. Performing Requirements Determination

- There are two sub phases to systems analysis: requirements determination and requirements structuring.
- We will address these as separate steps, but you should consider the steps as parallel and iterative.
- **For example**, as we determine some aspects of the current and desired system(s), we begin to structure these requirements or build prototypes to show users how a system might behave. Inconsistencies and deficiencies discovered through structuring and prototyping lead us to explore further the operation of current system(s) and the future needs of the organization. Eventually, our ideas and discoveries converge in a thorough and accurate depiction (the way that something is represented or shown) of current operations and requirements for the new system.



- Once management has granted permission to pursue (follow or start) development of a new system (this was done at the end of the project identification and selection phase of the SDLC) and a project is initiated and planned, you begin determining what the new system should do.
- During requirements determination, you and other analysts gather information on what the system should do from as many sources as possible: from users of the current system; from observing users; and from reports, forms, and procedures.
- All of the system requirements are carefully documented and prepared for structuring.
- In many ways, gathering system requirements is like conducting any investigation.

**These characteristics include the following:**

**a. Impertinence:**

You should question everything. You need to ask questions such as: Are all transactions processed the same way? Could anyone be charged something other than the standard price? Might we someday want to allow and encourage employees to work for more than one department?

**b. Impartiality:**

Your role is to find the best solution to a business problem or opportunity. It is not, for example, to find a way to justify the purchase of new hardware or to insist on incorporating what users think they want into the new system requirements. You must consider issues raised by all parties and try to find the best organizational solution.

**c. Relax constraints:**

Assume that anything is possible and eliminate the infeasible. For example, do not accept this statement: “We’ve always done it that way, so we have to continue the practice.” Traditions are different from rules and policies. Traditions probably started for a good reason but, as the organization and its environment change, they may turn into habits rather than sensible procedures.

**d. Attention to details:**

Every fact must fit with every other fact. One element out of place means that even the best system will fail at some time. For example, an imprecise (not accurate or exact ) definition of who a customer is may mean that you purge (remove) customer data when a customer has no active orders, yet these past customers may be vital contacts for future sales.

**e. Reframing:**

Analysis is, in part, a creative process. You must challenge yourself to look at the organization in new ways. You must consider how each user views his or her requirements. You must be careful not to jump to the following conclusion: “I worked on a system like that once—this new system must work the same way as the one I built before.”

**f. Deliverables and outcomes:**

The primary deliverables from requirements determination are the various forms of information gathered during the determination process: transcripts of interviews; notes from observation and analysis of documents; sets of forms, reports, job descriptions, and other documents; and computer-generated output such as system prototypes. In short, anything that the analysis team collects as part of determining system requirements is included in the deliverables resulting from this sub phase of the systems development life cycle. Examples of some specific information that might be gathered during requirements determination.

We could group all this information in three groups:

- Information collected from conversations with or observations of users (interview transcripts, notes from observations etc.)
- Existing written information (business mission or strategy, forms, reports, training manuals, flow charts and documentation of existing system etc.)
- Computer-based information (results from JRP sessions, CASE repository contents)

**g. Analysis Paralysis:**

Too much analysis is not productive, and the term analysis paralysis has been coined (invented) to describe a systems development project that has become bogged down (to be/become so involved in something difficult or complicated that you cannot do anything else ) in an abundance of analysis work. Because of the dangers of excessive analysis, today’s systems analysts focus more on the system to be developed than on the current system. The techniques you will learn about later in this chapter, JAD and prototyping, were developed to keep the analysis effort at a minimum yet still keep it effective. Newer techniques have also been developed to keep requirements determination fast and flexible, including continual user involvement, usage centered design, and the Planning Game from eXtreme Programming.

### 3. Traditional methods for determining requirements

**TABLE 6-2** Traditional Methods of Collecting System Requirements

- Individually interview people informed about the operation and issues of the current system and future systems needs
- Interview groups of people with diverse needs to find synergies and contrasts among system requirements
- Observe workers at selected times to see how data are handled and what information people need to do their jobs
- Study business documents to discover reported issues, policies, rules, and directions as well as concrete examples of the use of data and information in the organization

- One of the best ways to get this information is to talk to the people who are directly or indirectly involved in the different parts of the organizations affected by the possible system changes: users, managers, funders, and so on.
- Another way to find out about the current system is to gather copies of documentation relevant to current systems and business processes.

#### **a. Interviewing and listening:**

Interviewing is one of the primary ways analysts gather information about an information systems project. Early in a project, an analyst may spend a large amount of time interviewing people about their work, the information they use to do it, and the types of information processing that might supplement their work. Other stakeholders are interviewed to understand organizational direction, policies, expectations managers have on the units they supervise, and other non-routine aspects of organizational operations.

During interviewing you will gather facts, opinions, and speculation (Guess, when you guess possible answers to a question without having enough information to be certain ) and observe body language, emotions, and other signs of what people want and how they assess current systems.

There are many ways to effectively interview someone, and no one method is necessarily better than another.

**TABLE 6-3 Guidelines for Effective Interviewing**

Plan the Interview
• Prepare interviewee: appointment, priming questions
• Prepare checklist, agenda, and questions
Listen carefully and take notes (record if permitted)
Review notes within 48 hours of interview
Be neutral
Seek diverse views

#### **Open-ended (no pre-specified answer)**

- Open-ended questions are usually used to probe for information for which you cannot anticipate all possible responses or for which you do not know the precise question to ask.
- The person being interviewed is encouraged to talk about whatever interests him or her within the general bounds of the question.
- An example is, “What would you say is the best thing about the information system you currently use to do your job?” or “List the three most frequently used menu options.” You must react quickly to answers and determine whether or not any follow-up questions are needed for clarification or elaboration.
- A major disadvantage of openended questions is the length of time it can take for the questions to be answered. In addition, open-ended questions can be difficult to summarize.

#### **Closed-ended**

- Closed-ended questions provide a range of answers from which the interviewee may choose.
- Closed-ended questions work well when the major answers to questions are well known.
- Another plus is that interviews based on closed-ended questions do not necessarily require a large time commitment—more topics can be covered.
- You can see body language and hear voice tone, which can aid in interpreting the interviewee’s responses. Closed-ended questions can also be an easy way to begin an interview and to determine which line of open-ended questions to pursue.
- Closed-ended questions, like objective questions on an examination, can follow several forms, including the following choices: True or false or MCQ.

#### **Interview Guidelines:**

- **First**, with either open- or closed-ended questions, do not phrase a question in a way that implies a right or wrong answer. The respondent must feel that he or she can put his or her true opinion and perspective and that his or her idea will be considered equally with those of others.
- **The second** guideline to remember about interviews is to listen very carefully to what is being said. Take careful notes or, if possible, record the interview (be sure to ask permission first!).

- **Third**, once the interview is over, go back to your office and type up your notes within 48 hours. If you recorded the interview, use the recording to verify the material in your notes. After 48 hours, your memory of the interview will fade quickly.
- **Fourth**, be careful during the interview not to set expectations about the new or replacement system unless you are sure these features will be part of the delivered system.
- **Fifth**, seek a variety of perspectives from the interviews. Find out what potential users of the system, users of other systems that might be affected by changes, managers and superiors, information systems staff who have experience with the current system, and others think the current problems and opportunities are and what new information services might better serve the organization.

**b. Interviewing groups:**

- In a group interview, several key people are interviewed at once. To make sure all of the important information is collected, you may conduct the interview with one or more analysts. In the case of multiple interviewers, one analyst may ask questions while another takes notes, or different analysts might concentrate on different kinds of information.
- For example, one analyst may listen for data requirements while another notes the timing and triggering of key events. The number of interviewees involved in the process may range from two to however many you believe can be comfortably accommodated.
- A group interview has a few advantages. One, it is a much more effective use of your time than a series of interviews with individuals (although the time commitment of the interviewees may be more of a concern). Two, interviewing several people together allows them to hear the opinions of other key people and gives them the opportunity to agree or disagree with their peers.
- The primary disadvantage of a group interview is the difficulty in scheduling it.

**c. Directly observing users:**

Employees who know they are being observed may be nervous and make more mistakes than normal, may be careful to follow exact procedures they do not typically follow, and may work faster or slower than normal. Moreover, because observation typically cannot be continuous, you receive only a snapshot image of the person or task you observe, which may not include important events or activities.

Because observation is very time consuming, you will not only observe for a limited time, but also a limited number of people and a limited number of sites. Again, observation yields only a small segment of data from a possibly vast variety of data sources. Exactly which people or sites to observe is a difficult selection problem. You want to pick both typical and atypical people and sites, and observe during normal and abnormal conditions and times to receive the richest possible data from observation.

**d. Analyzing Procedures and Other Documents**

By examining existing system and organizational documentation, system analyst can find out details about current system and the organization these systems supports.

In documents analyst can find information such as problem with existing systems, opportunities to meet new needs if only certain information or information processing were available, organizational direction that can influence information system requirements, and the reason why current systems are designed as they are, etc.

### What we get from documents:

- Problems with existing system
- Opportunity to meet new need
- Organize direction
- Reasons for current system design
- Rules for processing data

## 4. Contemporary Methods for determining system Requirements

Even though we called interviews, observation, and document analysis traditional methods for determining a system's requirements, all of these methods are still very much used by analysts to collect important information. Today, however, there are additional techniques to collect information about the current system, the organizational area requesting the new system, and what the new system should be like.

In this section, you will learn about several contemporary information-gathering techniques for analysis: **JAD, CASE tools to support JAD, and prototyping.**

As we said earlier, these techniques can support effective information collection and structuring while reducing the amount of time required for analysis.

**TABLE 6-5 Contemporary Methods for Collecting System Requirements**

- Bringing session users, sponsors, analysts, and others together in a JAD session to discuss and review system requirements
- Using *CASE tools* during a JAD to analyze current systems to discover requirements that will meet changing business conditions
- Iteratively developing system *prototypes* that refine the understanding of system requirements in concrete terms by showing working versions of system features

### a. Joint Application Design (JAD):

A structured process in which users, managers and analysts work together for several days in a series of intensive meetings to specify or review system requirements.

Joint Application Design (JAD) started in the late 1970s at IBM, and since then the practice of JAD has spread throughout many companies and industries.

The main idea behind JAD is to bring together the **key users, managers, and systems analysts** involved in the analysis of a current system.

The primary purpose of using JAD in the analysis phase is to collect systems requirements simultaneously from the key people involved with the system.

The result is an intense (extreme and forceful or (of a feeling) very strong) and structured, but highly effective, process. As with a group interview, having all the key people together in one place at one time allows analysts to see where there are areas of agreement and where there are conflicts.

Meeting with all of these important people for over a week of intense sessions allows you the opportunity to resolve conflicts, or at least to understand why a conflict may not be simple to resolve.

The idea behind such a practice is to keep participants away from as many distractions as possible so that they can concentrate on systems analysis.

A JAD may last anywhere from four hours to an entire week and may consist of several sessions.

A JAD employs thousands of dollars of corporate resources, the most expensive of which is the time of the people involved.

The **typical participants in a JAD** are listed below:

**JAD session leader:**

The JAD session leader organizes and runs the JAD. This person has been trained in group management and facilitation as well as in systems analysis. The JAD leader sets the agenda and sees that it is met; he or she remains neutral on issues and does not contribute ideas or opinions, but rather concentrates on keeping the group on the agenda, resolving conflicts and disagreements, and soliciting (manage) all ideas.

**Users:**

The key users of the system under consideration are vital participants in a JAD.

They are the only ones who have a clear understanding of what it means to use the system on a daily basis.

**Managers:**

Managers of the work groups who use the system in question provide insight into new organizational directions, motivations for and organizational impacts of systems, and support for requirements determined during the JAD.

**Sponsor:**

As a major undertaking due to its expense, a JAD must be sponsored by someone at a relatively high level in the company. If the sponsor attends any sessions, it is usually only at the very beginning or the end.

**Systems analysts:**

Members of the systems analysis team attend the JAD, although their actual participation may be limited. Analysts are there to learn from users and managers, not to run or dominate the process.

**Scribe:**

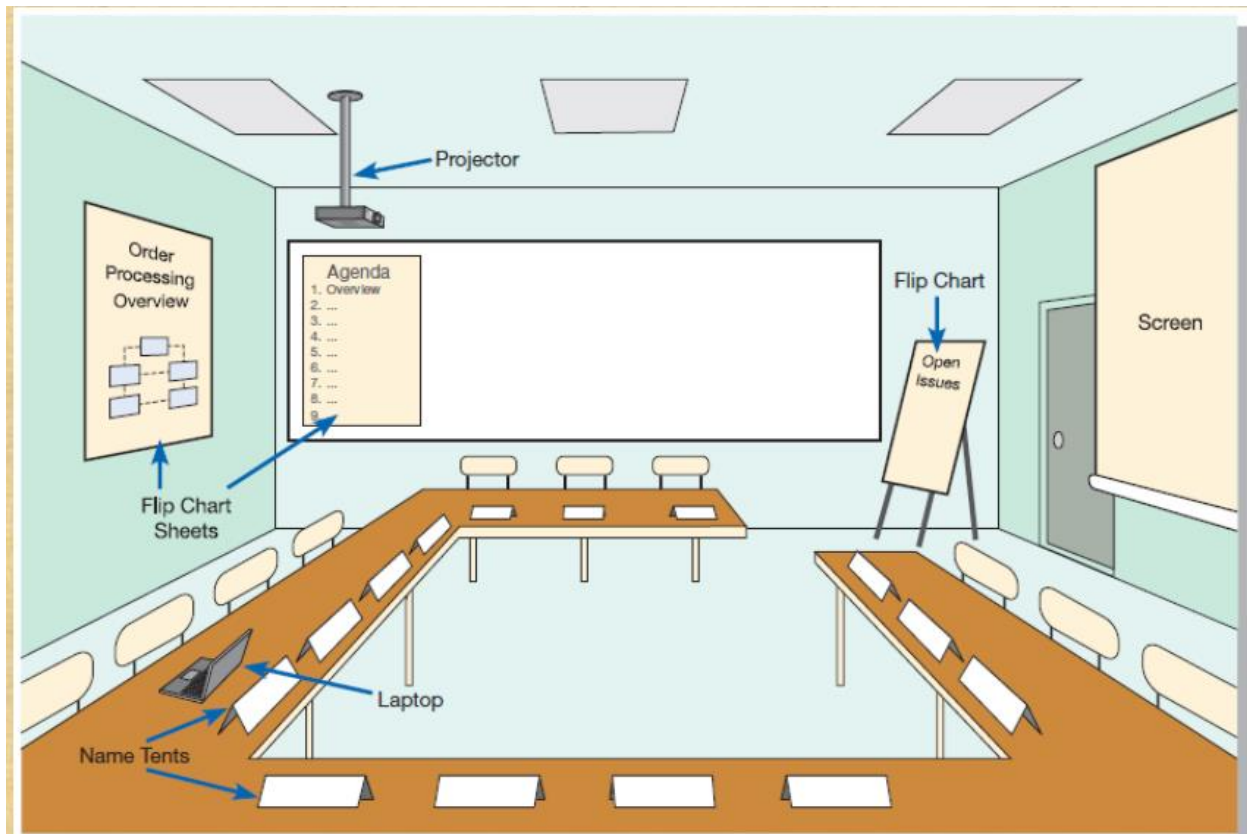
The scribe takes notes during the JAD sessions. This is usually done on a laptop. Notes may be taken using a word processor, or notes and diagrams may be entered directly into a CASE tool.

**IS staff:**

Besides systems analysts, other information systems (IS) staff, such as programmers, database analysts, IS planners, and data center personnel, may attend to learn from the discussion and possibly contribute their ideas on the technical feasibility of proposed ideas or the technical limitations of current systems.



## JAD Meeting Room design:



When a JAD is completed, the end result is a set of documents that detail the workings of the current system related to the study of a replacement system. Depending on the exact purpose of the JAD, analysts may also walk away from the JAD with some detailed information on what is desired of the replacement system.

**CASE Tools During JAD:** For requirements determination and structuring, the most useful CASE tools are used for diagramming and form and report generation.

Some observers advocate using CASE tools during JADs (Lucas, 1993). Running a CASE tool during a JAD enables analysts to enter system models directly into a CASE tool, providing consistency and reliability in the joint model-building process.

The CASE tool captures system requirements in a more flexible and useful way than can a scribe or an analysis team making notes. Further, the CASE tool can be used to project menu, display, and report designs, so users can directly observe old and new designs and evaluate their usefulness for the analysis team.

### **Advantage:**

- JAD allows you to resolve difficulties more simply and produce better, error-free software.
- The joint collaboration between the company and the clients lowers all risks.
- JAD reduces costs and time needed for project development.
- Well-defined requirements improve system quality.
- Due to the close communication, progress is faster.
- JAD encourages the team to push each other to work faster and deliver on time.

### Disadvantage:

- Different opinions within the team make it difficult to align goals and maintain focus
- Depending on the size of the project, JAD may require a significant time commitment.

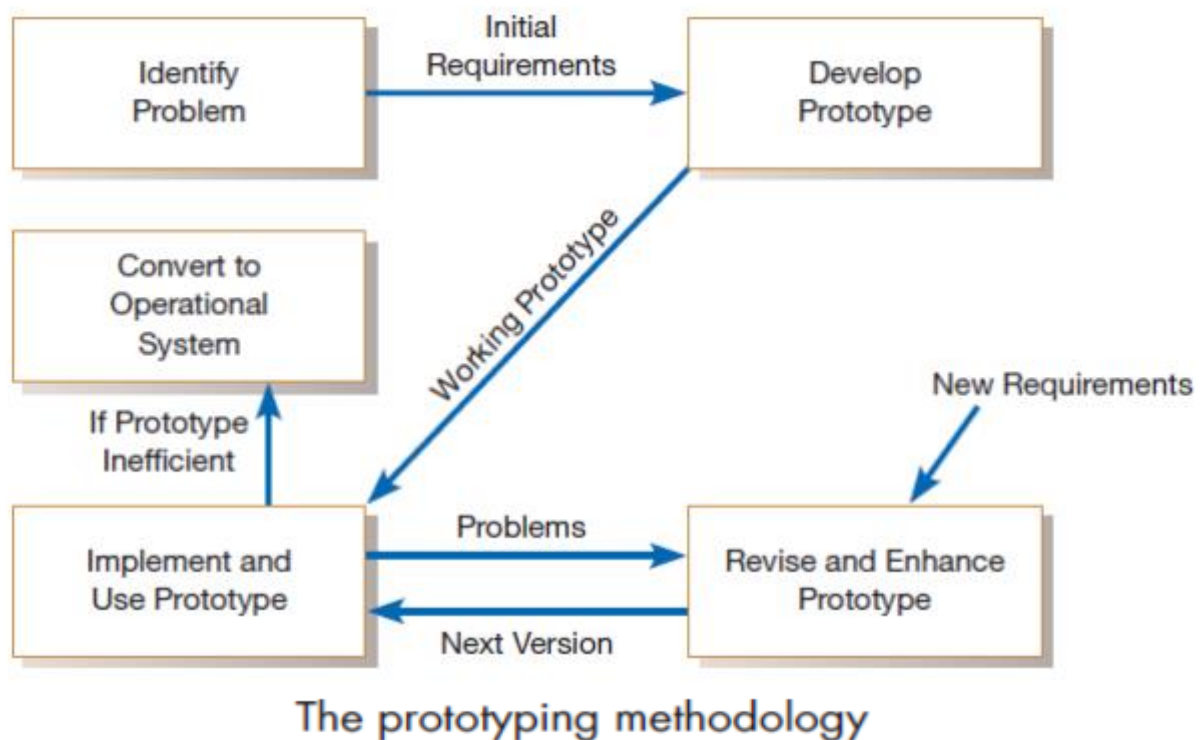
### b. Prototyping:

An iterative process of systems development in which requirements are converted to a working system that is continually revised through close collaboration between an analyst and users.

Prototyping will enable you to quickly convert basic requirements into a working, though limited, version of the desired information system.

The prototype will then be viewed and tested by the user. Typically, seeing verbal descriptions of requirements converted into a physical system will prompt the user to modify existing requirements and generate new ones.

For example, in the initial interviews, a user might have said that he wanted all relevant utility billing information (e.g., the client's name and address, the service record, and payment history) on a single computer display form. Once the same user sees how crowded and confusing such a design would be in the prototype, he might change his mind and instead ask to have the information organized on several screens, but with easy transitions from one screen to another. He might also be reminded of some important requirements (data, calculations, etc.) that had not surfaced during the initial interviews.



### Evolutionary prototyping:

- As the prototype changes through each iteration more and more of the design specifications for the system are captured in the prototype. The prototype can then serve as the basis for the production system in a process called **evolutionary prototyping**.

- In evolutionary prototyping, you begin by modeling parts of the target system and, if the prototyping process is successful, you evolve the rest of the system from those parts.
- A life-cycle model of evolutionary prototyping illustrates the iterative nature of the process and the tendency to refine the prototype until it is ready to release.
- Systems must be designed to support scalability, multiuser support, and multiplatform support. Few of these design specifications will be coded into a prototype. Further, as much as 90 percent of a system's functioning is devoted to handling exceptional cases.
- Prototypes are designed to handle only the typical cases, so exception handling must be added to the prototype as it is converted to the production system. Clearly, the prototype captures only part of the system requirements.

#### **Throwaway prototyping:**

- Alternatively, the prototype can serve only as a model, which is then used as a reference for the construction of the actual system. In this process, called **throwaway prototyping**, the prototype is discarded after it has been used.
- Unlike evolutionary prototyping, throwaway prototyping does not preserve the prototype that has been developed. With throwaway prototyping, there is never any intention to convert the prototype into a working system.
- Instead, the prototype is developed quickly to demonstrate some aspect of a system design that is unclear or to help users decide among different features or interface characteristics.
- Once the uncertainty the prototype was created to address has been reduced, the prototype can be discarded, and the principles learned from its creation and testing can then become part of the requirements determination.

## **5. Radical Methods for Determining System Requirements**

The overall process by which current methods are replaced with radically new methods is generally referred to as **business process reengineering (BPR)**.

To better understand **BPR**, consider the following analogy. Suppose you are a successful European golfer who has tuned your game to fit the style of golf courses and weather in Europe. You have learned how to control the flight of the ball in heavy winds, roll the ball on wide open greens, putt on large and undulating greens, and aim at a target without the aid of the landscaping common on North American courses. When you come to the United States to make your fortune on the US tour, you discover that incrementally improving you're putting, driving accuracy, and sand shots will help, but the new competitive environment is simply not suited to your style of the game. You need to reengineer your whole approach, learning how to aim at targets, spin and stop a ball on the green, and manage the distractions of crowds and the press. If you are good enough, you may survive, but without reengineering, you will never be a winner. Organizations realize that creatively using information technologies can yield significant improvements in most business processes.

### **Identifying Processes To Reengineer**

A first step in any BPR effort relates to understanding what processes to change. To do this, you must first understand which processes represent the key business processes for the organization. Key business processes are the structured set of measurable activities designed to produce a specific output for a

particular customer or market. The important aspect of this definition is that key processes are focused on some type of organizational outcome, such as the creation of a product or the delivery of a service. Key business processes are also customer focused. In other words, key business processes would include all activities used to design, build, deliver, support, and service a particular product for a particular customer.

### **Disruptive technologies:**

Once key business processes and activities have been identified, information technologies must be applied to radically improve business processes. To do this, Hammer and Champy (1993) suggest that organizations think “inductively” about information technology. Induction is the process of reasoning from the specific to the general, which means that managers must learn the power of new technologies and think of innovative (using new methods or ideas) ways to alter the way work is done. This is contrary to deductive thinking, where problems are first identified and solutions are then formulated.

## **6. Requirements Management Tools**

Organizations and developers have always been looking for more effective and creative ways to create and maintain requirements documents. One method that has been developed is computer-based requirements management tools. These tools make it easier for analyst to keep requirements, current documents and to define links between different parts of the overall specifications package. Requirements management tools are typically designed to work with many of the standards now available for requirements specification such as the Unified Modeling Language 2.0 (Structural thing, behavioral thing, grouping thing and notational thing) (help to specify, visualize and document models), System modeling language, Business process modeling notation etc.

Requirements management tools is best to traditional, planning based systems development approaches. They do not work well with **agile methodologies** which tend to employ different approaches to requirements gathering.

### **Requirements Determination Using Agile Methodologies:**

Three techniques are presented in this section.

- The first is **continual user involvement** in the development process, a technique that works especially well with small and dedicated development teams.
- The second approach is a **JAD-like process called Agile Usage-Centered Design.**
- The third approach is the **Planning Game, which was developed as part of eXtreme Programming.**

### **Continual User Involvement**

In Chapter 1, we read about the criticisms of the traditional waterfall SDLC. One of those criticisms was that the waterfall SDLC allowed users to be involved in the development process only in the early stages of analysis. Once requirements had been gathered from them, the users were not involved again in the process until the system was being installed and they were asked to sign off on it. Typically, by the time the users saw the system again, it was nothing like what they had imagined. The system most likely did not adequately address user needs. One approach to the problem of limited user involvement is to involve the users continually, throughout the entire analysis and design process. Such an approach works best when

development can follow the analysis–design–code–test cycle favored by the Agile Methodologies (Figure 6-9), because the user can provide information on requirements and then watch and evaluate as those requirements are designed, coded, and tested. This iterative process can continue through several cycles, until most of the major functionality of the system has been developed.

Extensive involvement users in the analysis and design process are a key part of many Agile approaches, but it was also a key part of Rapid Application Development (see Chapter 1).

### **Agile Usage-Centered Design**

Continual user involvement in systems development is an excellent way to ensure that requirements are captured accurately and immediately implemented in system design. However, such constant interaction works best when the development team is small, as was the case in the Boeing example. Also, it is not always possible to have continual access to users for the duration of a development project. Thus, agile developers have come up with other means for effectively involving users in the requirements determination process. One such method is called Agile Usage-Centered Design, originally developed by Larry Constantine (2002) and adapted for Agile Methodologies by Jeff Patton (2002).

### **The Planning Game From eXtreme Programming**

You read about eXtreme Programming in Chapter 1, and you know that it is an approach to software development put together by Kent Beck (Beck and Andres, 2004). You also know that it is distinguished by its short cycles, its incremental planning approach, its focus on automated tests written by programmers and customers to monitor the process of development, and its reliance on an evolutionary approach to development that lasts throughout the lifetime of the system. One of the key emphases of eXtreme Programming is its use of two-person programming teams and having a customer on-site during the development process.

The relevant parts of eXtreme Programming that relate to requirements determination are:

- (1) How planning, analysis, design, and construction are all fused together into a single phase of activity
- (2) Its unique way of capturing and presenting system requirements and design specifications.

All phases of the life cycle converge into a series of activities based on the basic processes of coding, testing, listening, and designing.

## **B. System Process Requirements:**

### **1. Introduction:**

- In this topic, our focus will be on one tool that is used to coherently represent the information gathered as part of requirements determination—data flow diagrams.
- Data flow diagrams enable you to model how data flow through an information system, the relationships among the data flows, and how data come to be stored at specific locations.
- Data flow diagrams also show the processes that change or transform data. Because data flow diagrams concentrate on the movement of data between processes, these diagrams are called process models.

- Decision tables allow you to represent the conditional logic that is part of some data flow diagram processes.

## 2. Process Modeling:

Process modeling involves graphically representing the functions, or processes, that capture, manipulate, store, and distribute data between a system and its environment and between components within a system. A common form of a process model is a data flow diagram (DFD).

DFDs, the traditional process modeling technique of structured analysis and design and one of the techniques most frequently used today for process modeling.

### Modeling a system's Process for structured Analysis:

Analysis phase of the systems development life cycle has two sub phases: **requirements determination and requirements structuring.**

The analysis team enters the requirements structuring phase with an abundance of information gathered during the requirements determination phase.

During requirements structuring, you and the other team members must organize the information into a meaningful representation of the information system that currently exists and of the requirements desired in a replacement system.

### Deliverables and outcomes:

**TABLE 7-1 Deliverables for Process Modeling**

- |   |
|---|
| <ol style="list-style-type: none"><li>1. Context DFD</li><li>2. DFDs of the system (adequately decomposed)</li><li>3. Thorough descriptions of each DFD component</li></ol> |
|---|

In structured analysis, the primary deliverables from process modeling are a set of coherent, interrelated DFDs.

Table 7-1 provides a more detailed list of the deliverables that result when DFDs are used to study and document a system's processes.

First, a context diagram shows the scope of the system, indicating which elements are inside and which are outside the system.

Second, DFDs of the system specify which processes move and transform data, accepting inputs and producing outputs. These diagrams are developed with sufficient detail to understand the current system and to eventually determine how to convert the current system into its replacement.

Finally, entries for all of the objects included in all of the diagrams are included in the project dictionary or CASE repository.

### 3. Data Flow Diagramming Mechanics:

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself.

DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools.

Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

DFD consists four symbols:


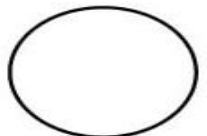
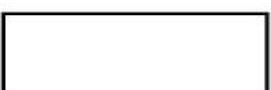
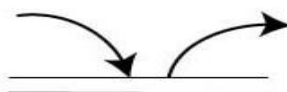
- data flows,
- data stores,
- processes, and
- Sources/sinks (or external entities).

**Data flow** is the path for data to move from one part of the system to another. It may be a single data element or set of data element. The symbol of data flow is the arrow which shows the flow direction.

A **data store** is data at rest. A data store may represent one of many different physical locations for data; for example, a file folder, one or more computer-based file(s), or a notebook. A data store might contain data about customers, students, customer orders, or supplier invoices.

A **process** is the work or actions performed on data so that they are transformed, stored, or distributed. When modeling the data processing of a system, it does not matter whether a process is performed manually or by a computer.

Finally, a **source/sink** is the origin and/or destination of the data. Sources/sinks are sometimes referred to as external entities because they are outside the system. Once processed, data or information leave the system and go to some other place.

Symbol	Name	Function
	Data flow	Used to Connect Processes to each other, to sources or Sinks; the arrow head indicates direction of data flow.
	Process	Performs Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.



## Advantages of DFD

- It helps us to understand the functioning and the limits of a system.
- It is a graphical representation which is very easy to understand as it helps visualize contents.
- Data Flow Diagram represent detailed and well explained diagram of system components.
- It is used as the part of system documentation file.
- Data Flow Diagrams can be understood by both technical and nontechnical person because they are very easy to understand.

## Disadvantages of DFD

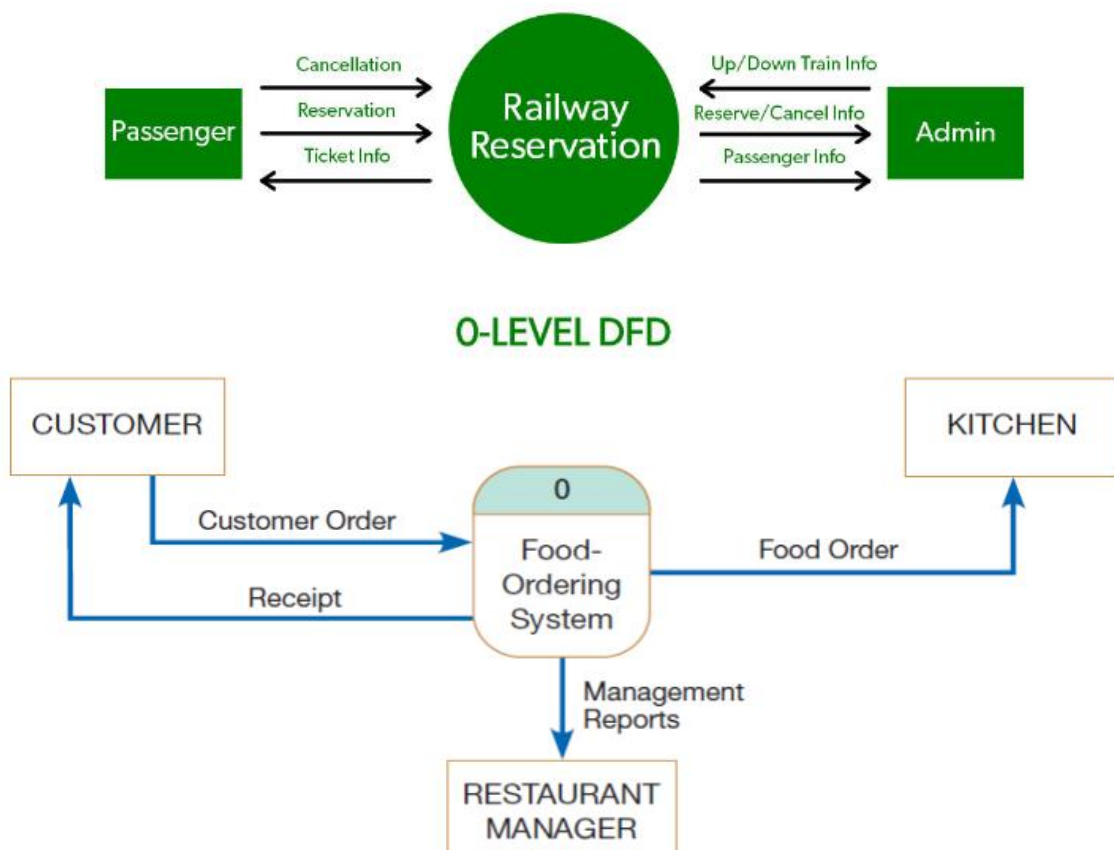
- At times DFD can confuse the programmers regarding the system.
- Data Flow Diagram takes long time to be generated, and many times due to this reasons analysts are denied permission to work on it.

## Levels in Data Flow Diagrams (DFD):

Three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

### a. 0-level DFD:

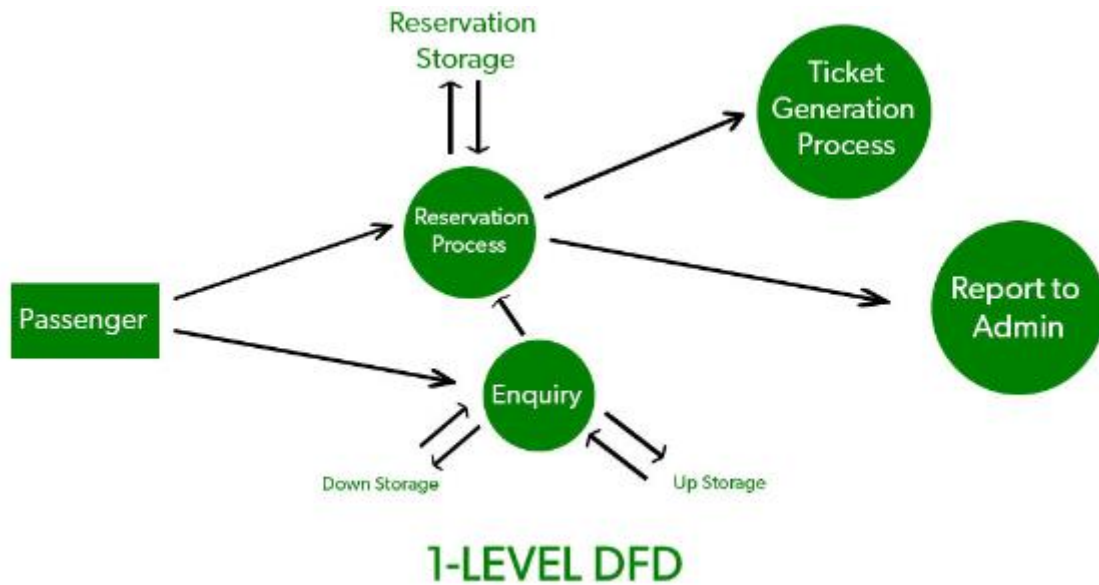
DFD Level 0 is also called a **Context Diagram**. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.





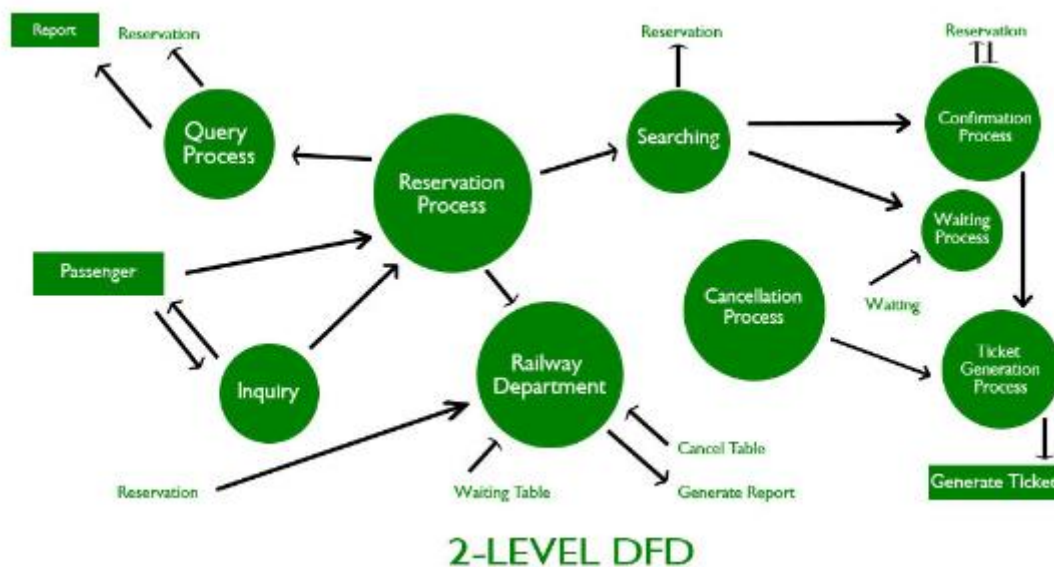
**b. 1-level DFD:**

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.



**c. 2-level DFD:**

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.



**DFD rules and tips:**

- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

**Process:**

- No process can have only outputs (a miracle)
- No process can have only inputs (black hole)
- No process can have partial inputs but complete output (grey hole)
- A process has a verb phrase label

**Data Store:**

- Data cannot be moved directly from one store to another
- Data cannot move directly from an outside source to a data store
- Data cannot move directly from a data store to a data sink
- Data store has a noun phrase label

**Source/Sink:**

- Data cannot move directly from a source to a sink
- A source/sink has a noun phrase label

**Data Flow:**

- A data flow has only one direction of flow between symbols
- A fork means that exactly the same data goes from a common location to two or more processes, data stores or sources/sinks

**Decomposition of DFDs**

The act of going from a single system to four component processes is called (functional) decomposition. Functional decomposition is an iterative process of breaking the description or perspective of a system down into finer and finer detail.

This process creates a set of hierarchically related charts in which one process on a given chart is explained in greater detail on another chart.

- Creates a set of charts in which one process on a given chart is explained in greater detail on another chart.
- Continues until no sub-process can logically be broken down any further.
- Lowest level is called a primitive DFD

**Level-N Diagrams**

A DFD that is the result of n nested decompositions of a series of sub processes from a process on a level-0 diagram.

**Balancing DFDs:****Conservation Principle:**

- Conserve inputs and outputs to a process at the next level of decomposition.

**Balancing:**

- Conservation of inputs and outputs to a data flow diagram process when that process is decomposed to a lower level.

**Balanced means:**

- Number of inputs to lower level DFD equals number of inputs to associated process of higher-level DFD.
- Number of outputs to lower level DFD equals number of outputs to associated process of higher-level DFD.

When you decompose a DFD from one level to the next, there is a conservation principle at work. You must conserve inputs and outputs to a process at the next level of decomposition. In other words, Process 1.0, which appears in a level-0 diagram, must have the same inputs and outputs when decomposed into a level-1 diagram. This conservation of inputs and outputs is called balancing.

## **4. Using Data flow Diagramming in the Analysis Process:**

Learning the mechanics of drawing DFDs is important because DFDs have proven to be essential tools for the structured analysis process. Beyond the issue of drawing mechanically correct DFDs, there are other issues related to process modeling with which an analyst must be concerned. Such issues, including whether the DFDs are complete and consistent across all levels, which covers guidelines for drawing DFDs. Another issue to consider is how you can use DFDs as a useful tool for analysis.

### **Guidelines for drawing DFDs:**

#### **a. Completeness:**

The concept of DFD completeness refers to whether you have included in your DFDs all of the components necessary for the system you are modeling. If your DFD contains data flows that do not lead anywhere or data stores, processes, or external entities that are not connected to anything else, your DFD is not complete.

#### **b. Consistency:**

The concept of DFD consistency refers to whether or not the depiction of the system shown at one level of a nested set of DFDs is compatible with the depictions of the system shown at other levels. A gross violation of consistency would be a level-1 diagram with no level-0 diagram. Another example of inconsistency would be a data flow that appears on a higher-level DFD but not on lower levels (also a violation of balancing).

#### **c. Timing:**

You may have noticed in some of the DFD examples we have presented that DFDs do not do a very good job of representing time. On a given DFD, there is no indication of whether a data flow occurs constantly in real time, once per week, or once per year. There is also no indication of when a system would run.

#### d. Iterative Development:

The first DFD you draw will rarely capture perfectly the system you are modeling. You should count on drawing the same diagram over and over again, in an iterative fashion. With each attempt, you will come closer to a good approximation of the system or aspect of the system you are modeling. One rule of thumb is that it should take you about three revisions for each DFD you draw

#### e. Primitive DFDs:

One of the more difficult decisions you need to make when drawing DFDs is when to stop decomposing processes. One rule is to stop drawing when you have reached the lowest logical level; however, it is not always easy to know what the lowest logical level is.

### Rules for stopping decomposition

- When each process has been reduced to a single decision, calculation or database operation
- When each data store represents data about a single entity
- When the system user does not care to see any more detail
- When every data flow does not need to be split further to show that data are handled in various ways
- When you believe that you have shown each business form or transaction, on-line display and report as a single data flow
- When you believe that there is a separate process for each choice on all lowest-level menu options

## 5. Modeling logic with decision Tables

- Data flow diagrams do not show the logic inside the processes
- Logic modeling involves representing internal structure and functionality of processes depicted on a DFD
- Logic modeling can also be used to show when processes on a DFD occur

#### The table has three parts:

The **condition stubs** contain the various conditions that apply to the situation the table is modeling.

- There are two condition stubs for employee type and hours worked. Employee type has two values: “S,” which stands for salaried, and “H,” which stands for hourly. Hours worked has three values: **less than 40, exactly 40, and more than 40.**

The **action stubs** contain all the possible courses of action that result from combining values of the condition stubs.

- There are four possible courses of action in this table: **Pay Base Salary, Calculate Hourly Wage, Calculate Overtime, and Produce Absence Report.**

Rules specify which actions are to be followed for a given set of conditions

### Complete decision table for payroll system

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce absence report		X				

#### Decision Table (Extended)

##### Steps:

- Identify
  - Conditions
 

Criteria: What and How many?
  - Actions
  - Rules
 

Figure out total number of rules
- Draw a 2D grid

#### Decision Table Compression

##### Steps:

- Start from extended table
- Identify indifferent rules
 

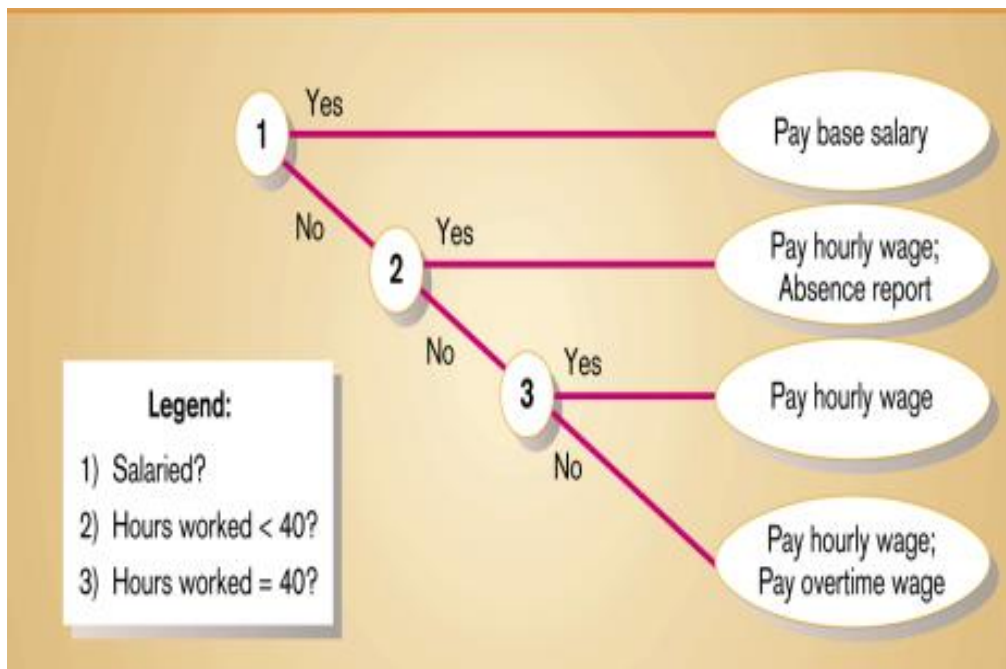
Does not change the action if one or more conditions are changed
- Replace that condition by a dash “-”
- Merge two identical rules
- It is better to do this one condition a time

**Reduced (Compression) decision table for payroll system**

Conditions/ Courses of Action	Rules			
	1	2	3	4
Employee type	S	H	H	H
Hours worked	–	< 40	40	>40
Pay base salary	X			
Calculate hourly wage		X	X	X
Calculate overtime				X
Produce absence report		X		

## 6. Modeling Logic with Decision Trees

- A graphical representation of a decision situation
- Decision situation points are connected together by arcs and terminate in ovals
- Two main **components**:
  - Decision points represented by nodes
  - Actions represented by ovals
- Read from **left to right**
- Each node corresponds to a numbered choice a legend
- All possible actions are listed on the **far right**



## C. System Data Requirements:

### 1. Introduction:

- In previous chapter, we learned how to model and analyze data. We learned how to show data stores, or data at rest, in a data flow diagram (DFD). DFDs, use cases, and various processing logic techniques show how, where, and when data are used or changed in an information system, but these techniques do not show the definition, structure, and relationships within the data.
- **Data modeling** develops these missing, and crucial, descriptive pieces of a system. The most common format used for data modeling is entity-relationship (E-R) diagramming.
- **Data models** that use E-R and class diagram notations explain the characteristics and structure of data independent of how the data may be stored in computer memory. A data model is usually developed iteratively, either from scratch or from a purchased data model for the industry or business area to be supported.

### 2. Conceptual Data Modeling

A conceptual data model is a representation of organizational data.

The purpose of a conceptual data model is to show as many rules about the meaning and interrelationships among data as are possible.

Conceptual data modeling is typically done in parallel with other requirements analysis and structuring steps during systems analysis.

On larger systems development teams, a subset of the project team concentrates on data modeling while other team members focus attention on process or logic modeling.

Analysts develop (or use from prior systems development) a conceptual data model for the current system and then build or refine a purchased conceptual data model that supports the scope and requirements for the proposed or enhanced system.

The work of all team members is coordinated and shared through the project dictionary or repository.

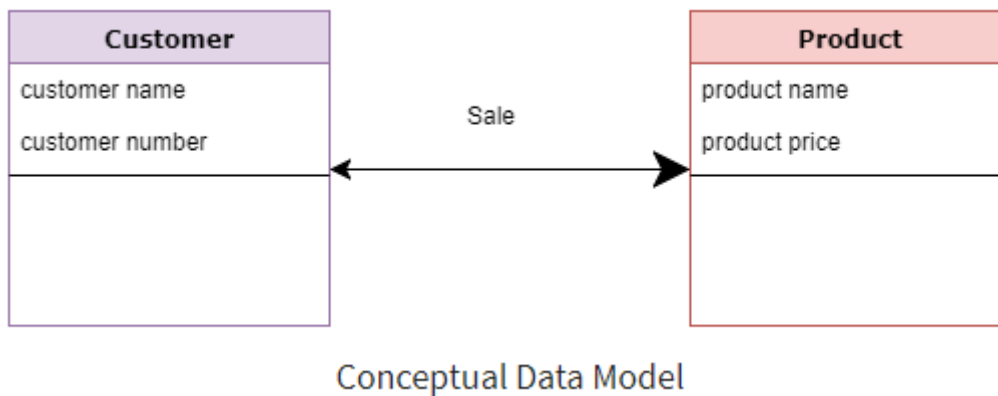
This repository is often maintained by a common Computer- Aided Software Engineering (CASE) or data modeling software tool.

**The 3 basic terms of Conceptual Data Model are:**

- **Entity:** A real-world thing
- **Attribute:** Characteristics or properties of an entity
- **Relationship:** Dependency or association between two entities

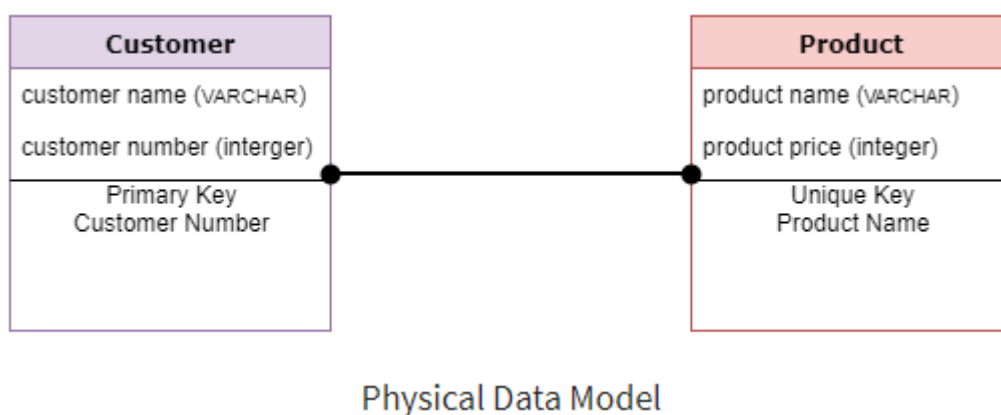
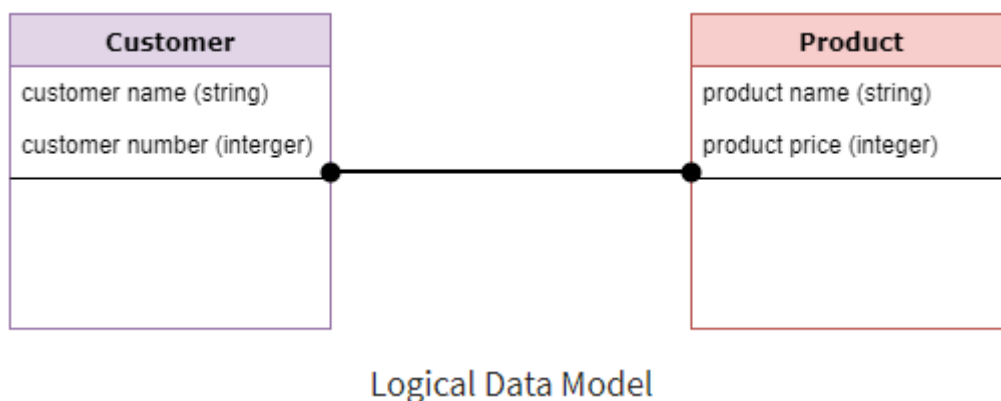
**Data model example:**

- Customer and Product are two entities. Customer number and name are attributes of the Customer entity
- Product name and price are attributes of product entity
- Sale is the relationship between the customer and product



**The Logical Data Model** is used to define the structure of data elements and to set relationships between them. The logical data model adds further information to the conceptual data model elements. The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic.

**A Physical Data Model** describes a database-specific implementation of the data model. It offers database abstraction and helps generate the schema. This is because of the richness of meta-data offered by a Physical Data Model. The physical data model also helps in visualizing database structure by replicating database column keys, constraints, indexes, triggers, and other RDBMS features.





## The Conceptual Data Modeling process

- The process of conceptual data modeling begins with developing a conceptual data model for the system being replaced, if a system already exists.
- This is essential for planning the conversion of the current files or database into the database of the new system.
- Further, this is a good, but not a perfect, starting point for your understanding of the data requirements of the new system. Then, a new conceptual data model is built (or a standard one is purchased) that includes all of the data requirements for the new system.

### Deliverables and outcomes:

Most organizations today do conceptual data modeling using **E-R (entity-relationship) modeling**, which uses a special notation to represent as much meaning about data as possible. Because of the rapidly increasing interest in **object-oriented methods**, **class diagrams using unified modeling language (UML-- modeling language that is most often used for software engineering but has extended its use to business processes and other project workflows, e.g.: Lucid chart, draw.io) drawing tools** such as IBM's Rational products or Microsoft Visio are also popular.

- The primary deliverable from the conceptual data modeling step within the analysis phase is an E-R diagram.
- The other deliverable from conceptual data modeling is a full set of entries about data objects that will be stored in the project dictionary, repository, or data modeling software. The repository is the mechanism to link data, process and logic models of an information system.

## 3. Gathering Information for Conceptual Data Modeling:

Requirements determination methods must include questions and investigations that take a data, not only a process and logic, focus. For example, during interviews with potential system users—during Joint Application Design (JAD) sessions or through requirements interviews—you must ask specific questions in order to gain the perspective on data that you need to develop or tailor a purchased data model.

You typically do data modeling from a combination of perspectives.

- The first perspective is generally called the top-down approach. This perspective derives the business rules for a data model from an intimate understanding of the nature of the business, rather than from any specific information requirements in computer displays, reports, or business forms.
- You can also gather the information you need for data modeling by reviewing specific business documents—computer displays, reports, and business forms— handled within the system.
- This process of gaining an understanding of data is often called a bottom-up approach. These items will appear as data flows on DFDs and will show the data processed by the system and, hence, probably the data that must be maintained in the system's database. Consider, for example, Figure 8-4, which shows a customer order form used at Pine Valley Furniture (PVF).

ORDER NO  
ORDER DATE  
PROMISED DATE|  
PRODUCT NO  
DESCRIPTION  
QUANTITY ORDERED  
UNIT PRICE

CUSTOMER NO  
NAME  
ADDRESS  
CITY-STATE-ZIP

PVF CUSTOMER ORDER			
ORDER NO: 61384		CUSTOMER NO: 1273	
NAME:		Contemporary Designs	
ADDRESS:		123 Oak St.	
CITY-STATE-ZIP:		Austin, TX 28384	
ORDER DATE: 11/04/2014		PROMISED DATE: 11/21/2017	
PRODUCT NO	DESCRIPTION	QUANTITY ORDERED	UNIT PRICE
M128	Bookcase	4	200.00
B381	Cabinet	2	150.00
R210	Table	1	500.00

**FIGURE 8-4**

Sample customer form

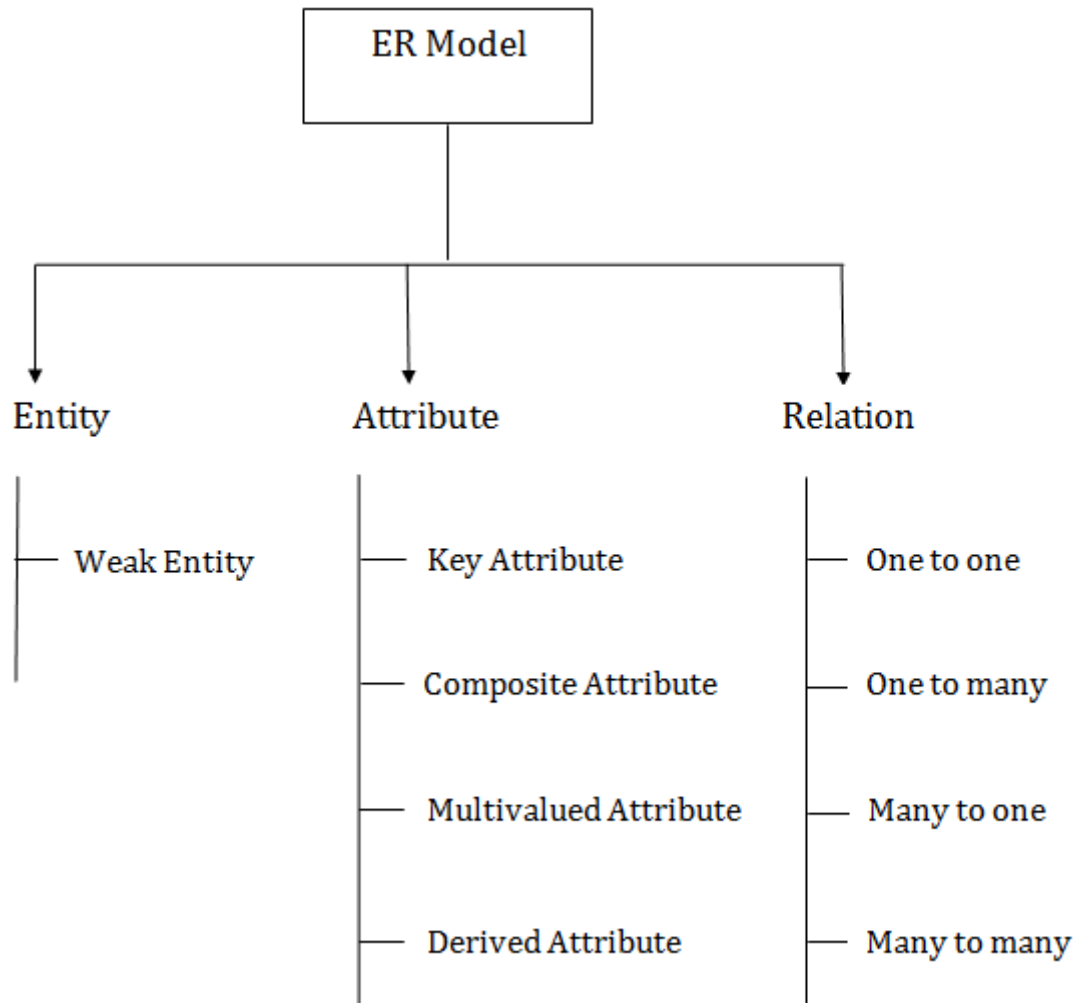
1. *What are the subjects/objects of the business?* What types of people, places, things, materials, events, etc. are used or interact in this business, about which data must be maintained? How many instances of each object might exist?—**data entities and their descriptions**
2. *What unique characteristic (or characteristics) distinguishes each object from other objects of the same type?* Might this distinguishing feature change over time or is it permanent? Might this characteristic of an object be missing even though we know the object exists?—**primary key**
3. *What characteristics describe each object?* On what basis are objects referenced, selected, qualified, sorted, and categorized? What must we know about each object in order to run the business?—**attributes and secondary keys**
4. *How do you use these data?* That is, are you the source of the data for the organization, do you refer to the data, do you modify it, and do you destroy it? Who is not permitted to use these data? Who is responsible for establishing legitimate values for these data?—**security controls and understanding who really knows the meaning of data**
5. *Over what period of time are you interested in these data?* Do you need historical trends, current “snapshot” values, and/or estimates or projections? If a characteristic of an object changes over time, must you know the obsolete values?—**cardinality and time dimensions of data**
6. *Are all instances of each object the same?* That is, are there special kinds of each object that are described or handled differently by the organization? Are some objects summaries or combinations of more detailed objects?—**supertypes, subtypes, and aggregations**
7. *What events occur that imply associations among various objects?* What natural activities or transactions of the business involve handling data about several objects of the same or a different type?—**relationships and their cardinality and degree**
8. *Is each activity or event always handled the same way or are there special circumstances?* Can an event occur with only some of the associated objects, or must all objects be involved? Can the associations between objects change over time (for example, employees change departments)? Are values for data characteristics limited in any way?—**integrity rules, minimum and maximum cardinality, time dimensions of data**

**TABLE 8-1** Requirements Determination Questions for Data Modeling

## 4. Introduction to E-R Modeling

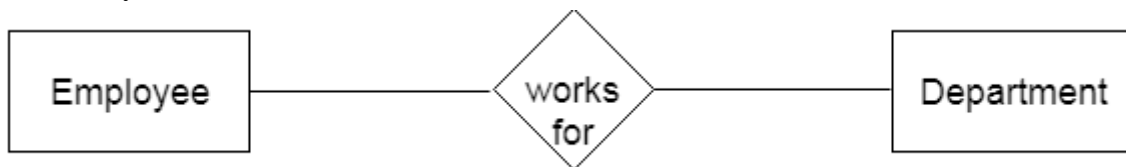
- An entity-relationship data model (E-R model) is a detailed, logical representation of the data for an organization or for a business area. The E-R model is expressed in terms of entities in the business environment, the relationships or associations among those entities, and the attributes or properties of both the entities and their relationships.
- An E-R model is normally expressed as an entity-relationship diagram (E-R diagram), which is a graphical representation of an E-R model.
- The basic E-R modeling notation uses three main constructs: data entities, relationships and their associated attributes.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

## Component/Structural/Constraints of ER Diagram



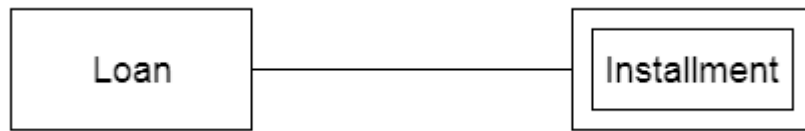
### a. Entity:

- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



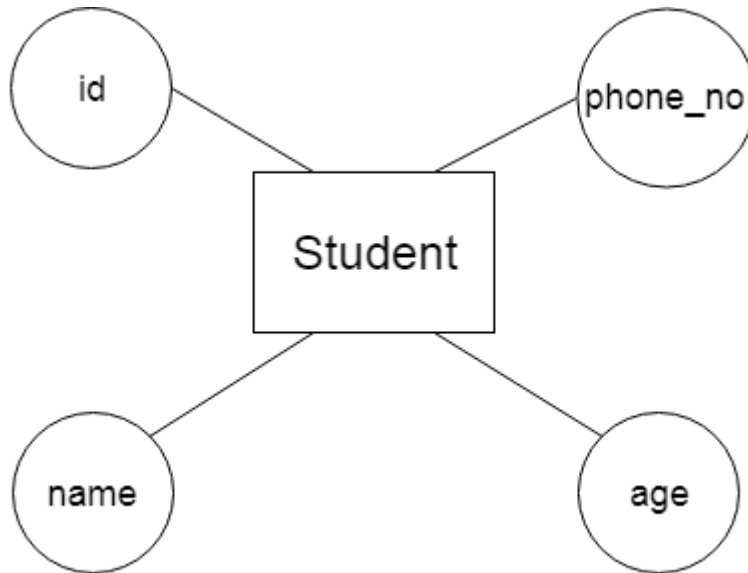
### i. Weak Entity

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



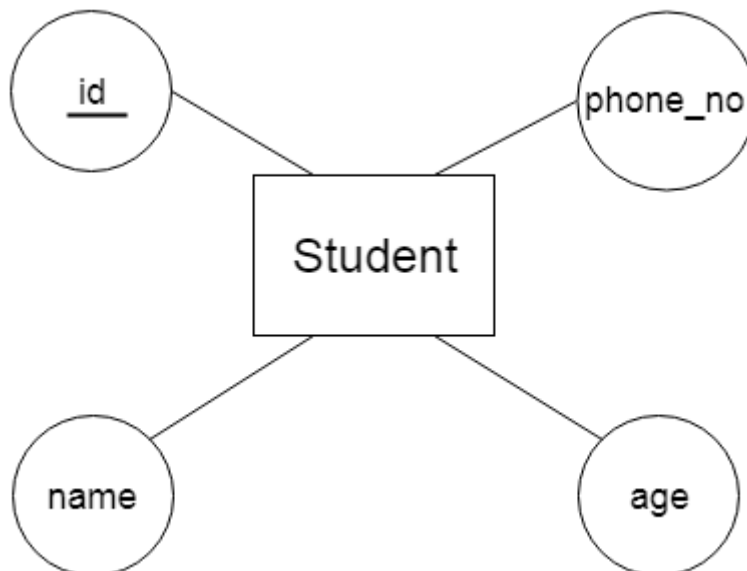
**b. Attribute**

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute. For example, id, age, contact number, name, etc. can be attributes of a student.



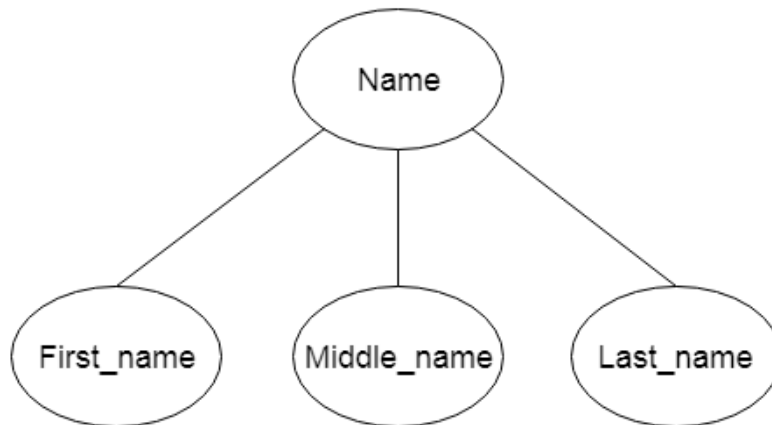
**i. Key Attribute:**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



## ii. Composite Attribute

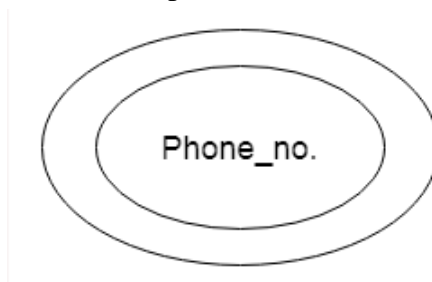
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



## iii. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

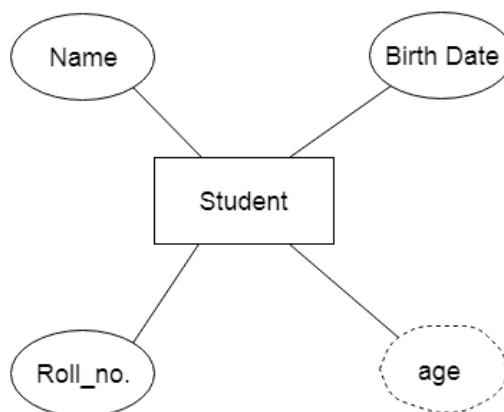
For example, a student can have more than one phone number.



## iv. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, a person's age changes over time and can be derived from another attribute like Date of birth.



### c. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



#### i. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

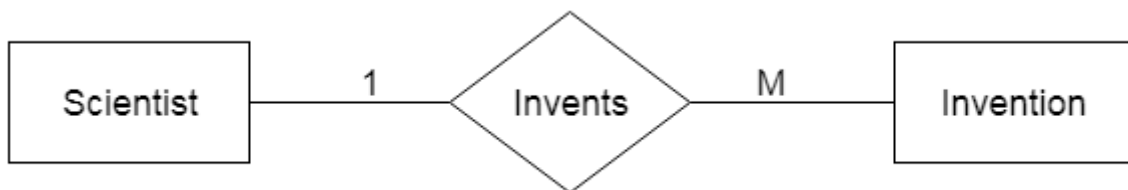
For example, a female can marry to one male, and a male can marry to one female.



#### ii. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

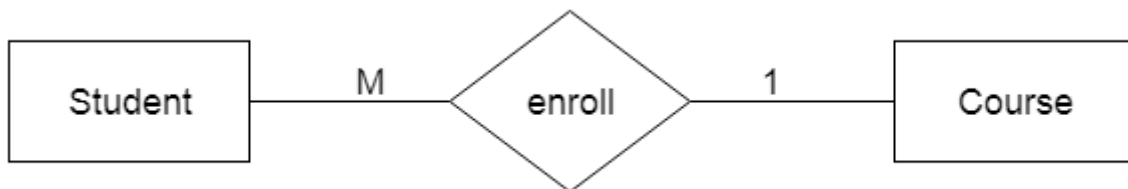
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



#### iii. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



#### iv. Many-to-many relationship

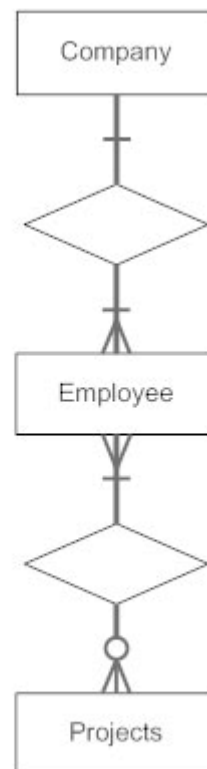
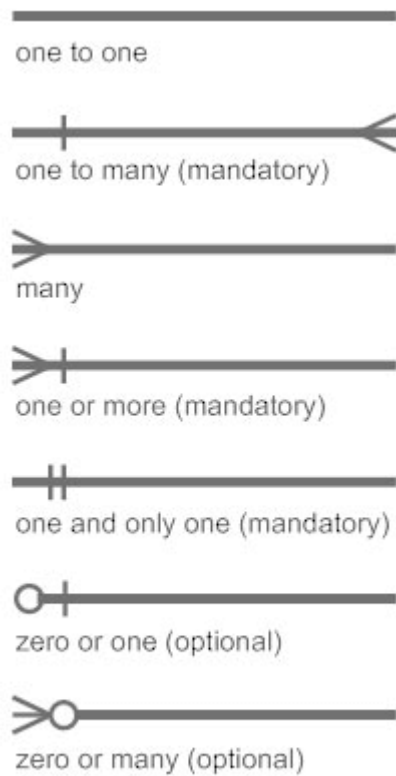
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.



#### Notation of ER diagram

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:





## 5. Conceptual Data Modeling and the E-R Model

### Degree of a Relationship

Number of entity sets that participate in a relationship set is called degree of the relationship set. On the basis of degree, relationships can be divided as below:

- **Unary Relationship**
- **Binary Relationship**
- **N-ary Relationship**

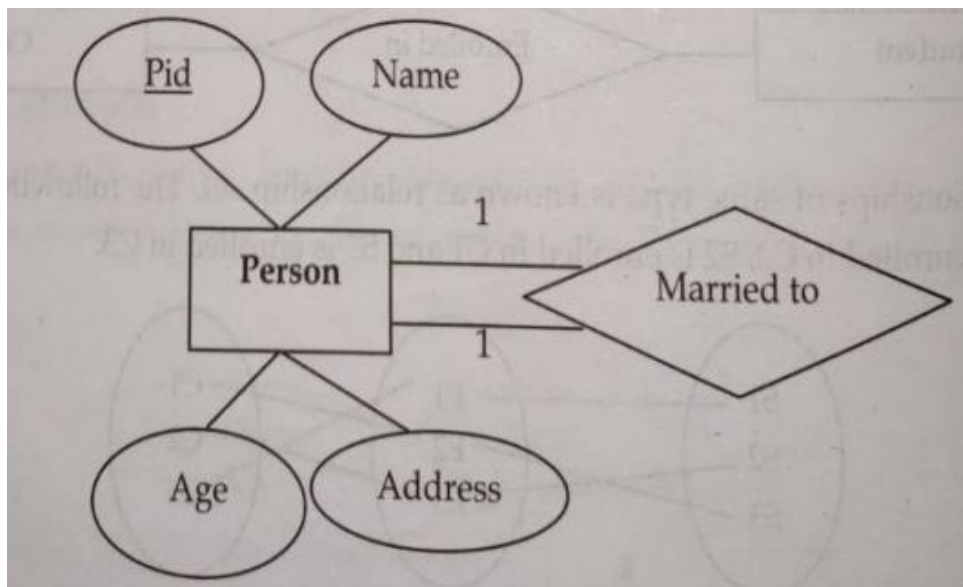
#### I. Unary Relationship

If only one entity set participates in a relation, the relationship is called as unary relationship. Here same entity set participates in relationship twice with different roles. Role names are specified above the link joining entity set and relationship set. This type of relationship set is sometimes called a recursive relationship set. There are three types of unary relationships:

- **1:1 unary relationship**
- **1: M unary relationship**
- **M: N unary relationship**

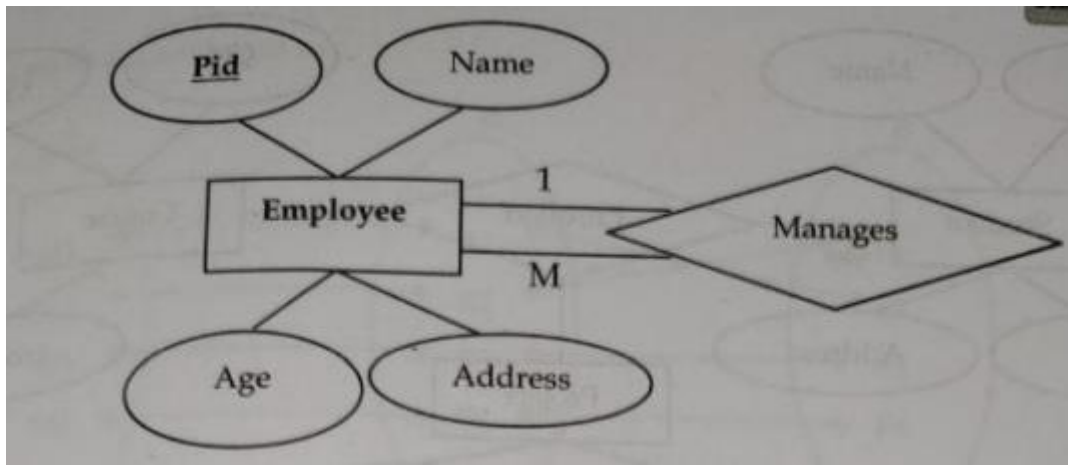
##### a. One to one (1:1) unary relationship

In the example below, one person is married to only one person.



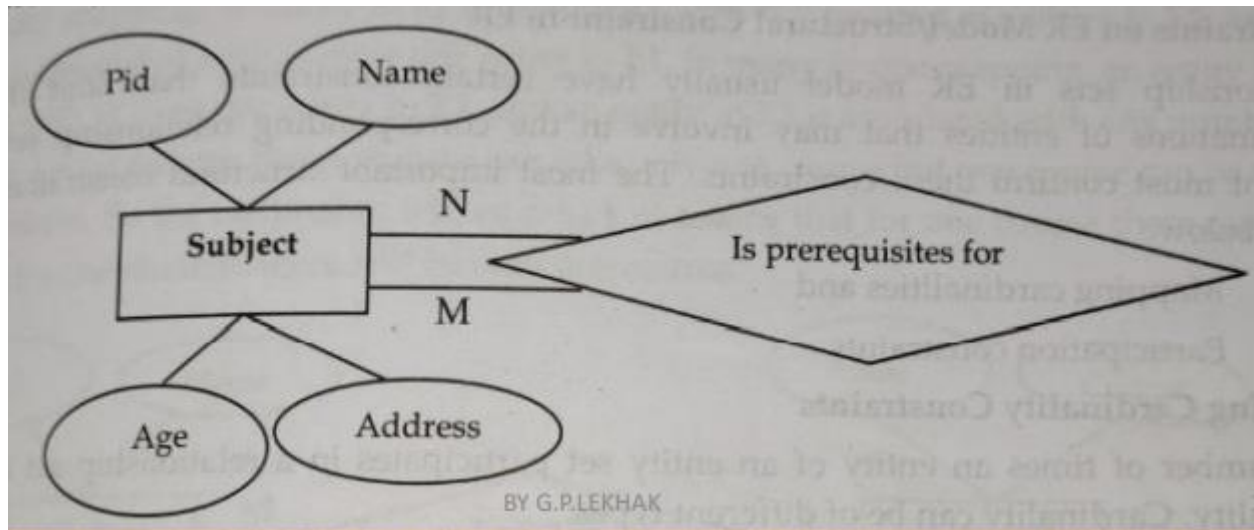
##### b. One to many (1: M) unary relationship

An employee may manage many employees but an employee is managed by only one employee. This types of relationship with employee relationship set itself is called 1: M unary relationship as shown in below;



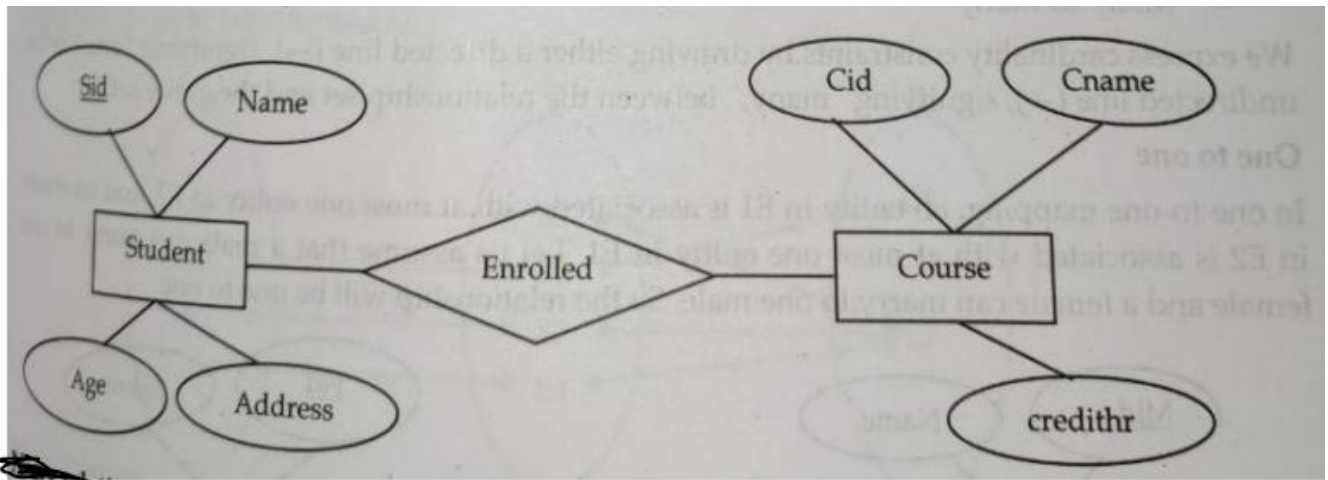
**c. Many to many (M: N) unary relationship**

A subject may have many other subjects as prerequisites and each subject may be a prerequisite to many other subjects.



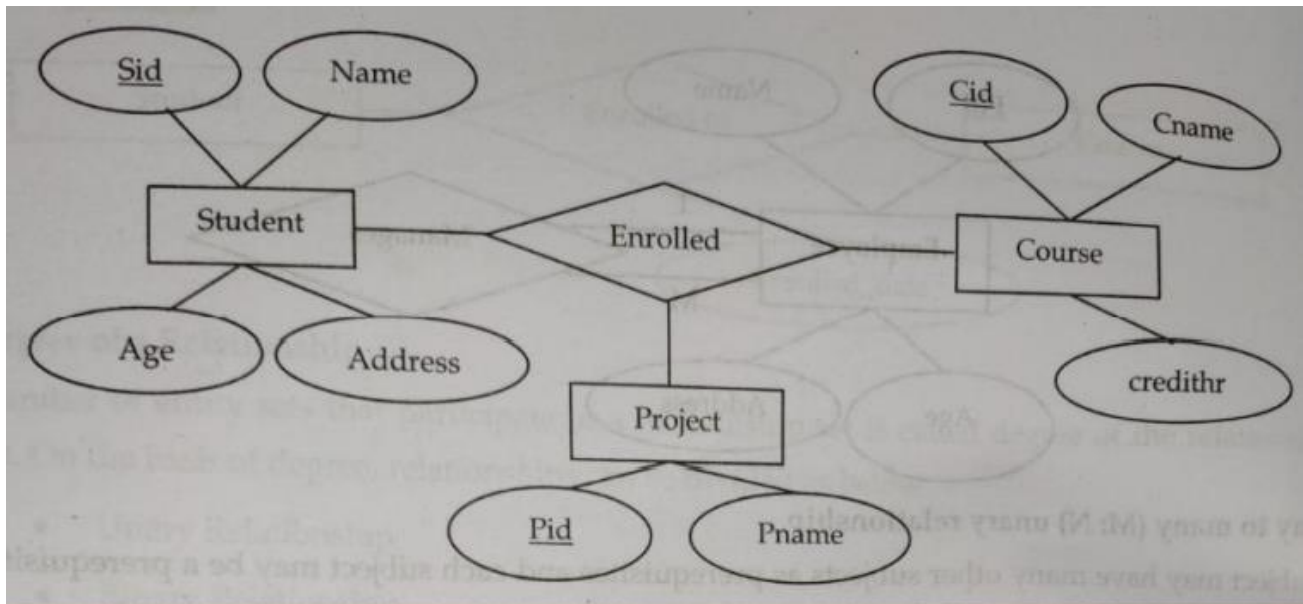
**II. Binary relationship**

When there are two entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course. This is the most common type of relationship in database systems.



### III. N-ary relationship

When there are  $n$  entities set participating in a relation, the relationship is called as  $n$ -ary relationship. If  $n=1$  then it is called unary relationship, if  $n=2$  then it is called binary relationship. Generally in  $N$ -ary relationship there are more than two entities participating with a single relationship i.e.  $n>2$ .



## 6. Representing Super types And Subtypes

- Often two or more entity types seem very similar (maybe they have almost the same name), but there are a few differences. That is, these entity types share common properties but also have one or more distinct attributes or relationships. To address this situation, the E-R model has been extended to include super type/subtype relationships.
- A **subtype** is a sub grouping of the entities in an entity type that is meaningful to the organization. **For example**, STUDENT is an entity type in a university. Two subtypes of STUDENT are GRADUATE STUDENT and UNDERGRADUATE STUDENT.

- A **super type** is a generic entity type that has a relationship with one or more subtypes.

### **Business Rules:**

Conceptual data modeling is a step-by-step process for documenting information requirements, and it is concerned with both the structure of data and with rules about the integrity of those data. Business rules are specifications that preserve the integrity of the logical data model. Four basic types of business rules are as follows:

1. Entity integrity: Each instance of an entity type must have a unique identifier that is not null.
2. Referential integrity constraints: Rules concerning the relationships between entity types.
3. Domains: Constraints on valid values for attributes.
4. Triggering operations: Other business rules that protect the validity of attribute values.

**Domains:** A domain is the set of all data types and ranges of values that attributes may assume.

**Triggering operations:** A triggering operation (also called a trigger) is an assertion or rule that governs the validity of data manipulation operations such as insert, update, and delete. The scope of triggering operations may be limited to attributes within one entity or it may extend to attributes in two or more entities.

## **7. Role of Packaged Conceptual Data Models in Database:**

There are two principal types of packaged data models: **universal data models** applicable to nearly any business or organization and **industry-specific data models**.

### **a. Universal Data Models:**

Numerous core subject areas are common to many (or even most) organizations, such as customers, products, accounts, documents, and projects. Although they differ in detail, the underlying data structures are often quite similar for these subjects. Further, there are core business functions such as purchasing, accounting, receiving, and project management that follow common patterns. Universal data models are templates for one or more of these subject areas and/or functions. All of the expected components of data models are generally included: **entities, relationships, attributes, primary and foreign keys, and even sample data.**

- **Identify the entities.** The process of data modeling begins with the identification of the things, events or concepts that are represented in the data set that is to be modeled. Each entity should be cohesive and logically discrete from all others.
- **Identify key properties of each entity.** Each entity type can be differentiated from all others because it has one or more unique properties, called attributes. For instance, an entity called “customer” might possess such attributes as a first name, last name, telephone number and salutation, while an entity called “address” might include a street name and number, a city, state, country and zip code.
- **Identify relationships among entities.** The earliest draft of a data model will specify the nature of the relationships each entity has with the others. In the above example, each customer “lives at” an address. If that model were expanded to include an entity called “orders,” each order would be

shipped to and billed to an address as well. These relationships are usually documented via unified modeling language (UML).

- **Map attributes to entities completely.** This will ensure the model reflects how the business will use the data. Several formal data modeling patterns are in widespread use. Object-oriented developers often apply analysis patterns or design patterns, while stakeholders from other business domains may turn to other patterns.
- **Assign keys as needed, and decide on a degree of normalization that balances the need to reduce redundancy with performance requirements.** Normalization is a technique for organizing data models (and the databases they represent) in which numerical identifiers, called keys, are assigned to groups of data to represent relationships between them without repeating the data. For instance, if customers are each assigned a key, that key can be linked to both their address and their order history without having to repeat this information in the table of customer names. Normalization tends to reduce the amount of storage space a database will require, but it can at cost to query performance.
- **Finalize and validate the data model.** Data modeling is an iterative process that should be repeated and refined as business needs change.

#### **b. Industry-Specific Data Models:**

Industry-specific data models are generic data models that are designed to be used by organizations within specific industries. Data models are available for nearly every major industry group, including health care, telecommunications, discrete manufacturing, process manufacturing, banking, insurance, and higher education. These models are based on the premise that data model patterns for organizations are very similar within a particular industry (“a bank is a bank”). However, the data models for one industry (such as banking) are quite different from those for another (such as hospitals).