Analysis

- Analysis is the first systems development life cycle (SDLC) phase where you begin to understand, in depth, the need for system changes.
- Systems analysis involves a substantial amount of effort and cost, and is therefore undertaken only after management has decided that the systems development project under consideration has merit and should be pursued through this phase.
- Most observers would agree that many of the errors in developed systems are directly traceable to inadequate efforts in the analysis and design phases of the life cycle.
- The purpose of analysis is to determine what information and information processing services are needed to support selected objectives and functions of the organization. Gathering this information is called requirements determination.

- The results of the requirements determination can be structured according to three essential views of the current and replacement information systems:
- **Process:** The sequence of data movement and handling operations within the system.
- **Logic and timing:** The rules by which data are transformed and manipulated and an indication of what triggers data transformation.
- **Data:** The inherent structure of data independent of how or when they are processed.

3.1 System requirements

3.1.1 Introduction,

3.1.2 Performing Requirements Determination,

3.1.3 Traditional Methods For Determining Requirements,

3.1.4 Contemporary Methods For Determining System Requirements,

3.1.5 Radical Methods For Determining System Requirements,

3.1.6 Requirements Management Tools,

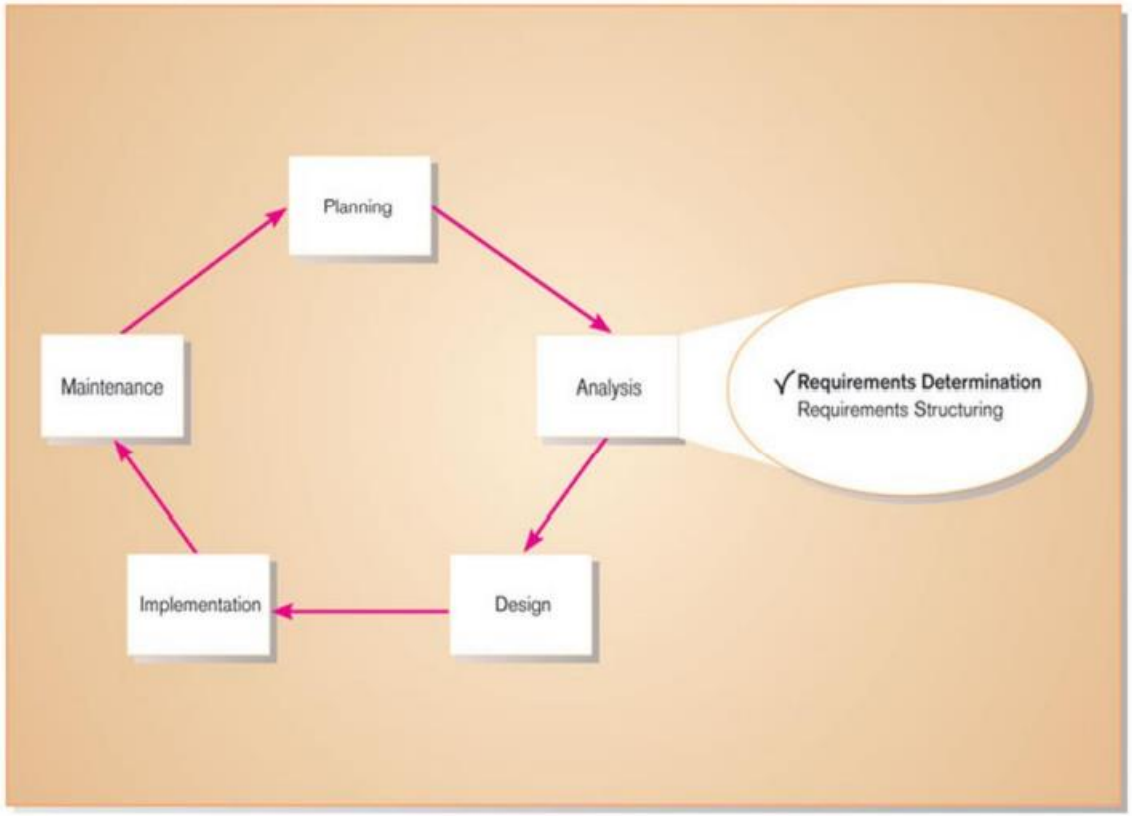
3.1.7 Requirement Determination Using Agile Methodologies

3.1SystemRequirements

3.1.1 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.**
- **Requirements Determination:** This is primarily a fact finding activity.
- **Requirements Structuring:** This activity creates a thorough and clear description of current business operations and new information processing services.

Figure 6-1 Systems development life cycle with analysis phase highlighted



3.1.2 Performing Requirements Determination

- 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.
- We can detect some similar characteristics for a good systems analysts during the requirements determination subphase.

- These characteristics include:
- **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?
- **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.
- **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

- **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.
- **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."

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 subphase of the systems development life cycle. Table 6-1 lists examples of some specific information that might be gathered during requirements determination.

TABLE 6-1 Deliverables for Requirements Determination

1. Information collected from conversations with or observations of users: interview transcripts, notes from observation, meeting minutes
2. Existing written information: business mission and strategy statements, sample business forms and reports and computer displays, procedure manuals, job descriptions, training manuals, flowcharts and documentation of existing systems, consultant reports
3. Computer-based information: results from JAD sessions, CASE repository contents and reports of existing systems, and displays and reports from system prototypes

3.1.3 Traditional Methods For Determining Requirements

- At the core of systems analysis is the collection of information. At the outset, you must collect information about the information systems that are currently being used and how users would like to improve the current systems and organizational operations with new or replacement information systems.
- 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.
- We will learn about various ways to get information directly from stakeholders: interviews, group interviews, the Nominal Group Technique, and direct observation. You will learn about collecting documentation on the current system and organizational operation in the form of written procedures, forms, reports, and other hard copy.

- These traditional methods of collecting system requirements are listed in Table 6-2.

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

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. Some guidelines you should keep in mind when you interview, summarized below:

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

- First, you should prepare thoroughly before the interview.
- Set up an appointment at a time and for a duration convenient for the interviewee. The general nature of the interview should be explained to the interviewee in advance. You may ask the interviewee to think about specific questions or issues or to review certain documentation to prepare for the interview.
- You should spend some time thinking about what you need to find out and write down your questions.
- Do not assume that you can anticipate all possible questions.
- You want the interview to be natural, and, to some degree, you want to spontaneously direct the interview as you discover what expertise the interviewee brings to the session.

Choosing Interview Questions

- You need to decide what mix and sequence of open-ended and closed-ended questions you will use.
- **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. Sometimes body language will suggest that a user has given an incomplete answer or is reluctant to divulge (to make something secret known) some information; a follow-up question might yield additional insight.

- One advantage of open-ended questions is that previously unknown information can surface. You can then continue exploring along unexpected lines of inquiry to reveal even more new information.
- Open-ended questions also often put the interviewees at ease (move) because they are able to respond in their own words using their own structure; open-ended questions give interviewees more of a sense of involvement and control in the interview. A major disadvantage of open-ended 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 questions** provide a range of answers from which the interviewee may choose. Here is an example:
- Which of the following would you say is the one best thing about the information system you currently use to do your job (pick only one)?

a. Having easy access to all of the data you need

b. The system's response time

c. The ability to access the system from remote locations

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. Questions such as “Should the system continue to provide the ability to override the default value, even though most users now do not like the feature?”.
- 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!). The answers may contain extremely important information for the project.
- **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. As you type and organize your notes, write down any additional questions that might arise from lapses (a temporary failure) in your notes or from ambiguous information.

- Make a list of unclear points that need clarification. Call the person you interviewed and get answers to these new questions. Use the phone call as an opportunity to verify the accuracy of your notes. You may also want to send a written copy of your notes to the person you interviewed so the person can check your notes for accuracy. Finally, make sure you thank the person for his or her time.
- **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. Let the interviewee know that there are many steps to the project and the perspectives of many people need to be considered, along with what is technically possible. Let respondents know that their ideas will be carefully considered, but that due to the iterative nature of the systems development process, it is premature to say now exactly what the ultimate system will or will not do.

- **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.

3.1.4 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 (listed in Table 6-5): 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

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.

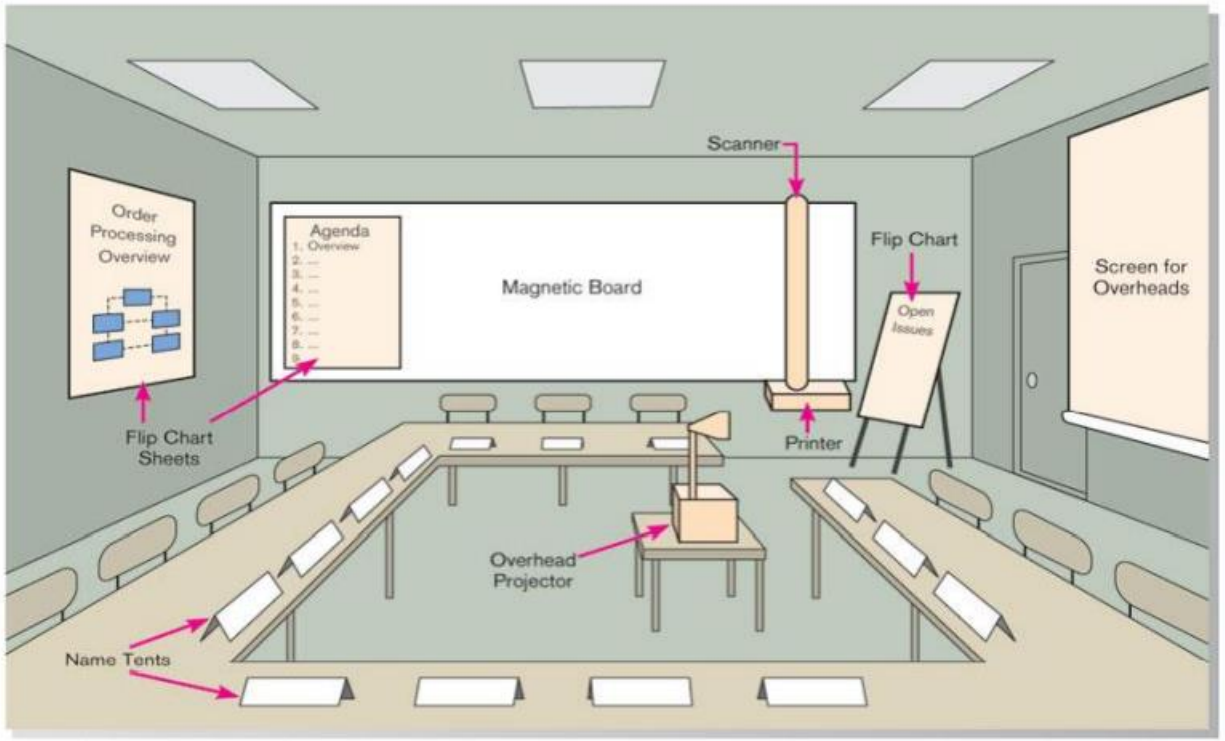
- JAD sessions are usually conducted at a location other than the place where the people involved normally work. The idea behind such a practice is to keep participants away from as many distractions as possible so that they can concentrate on systems analysis. A JAD may last anywhere from four hours 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. Other expenses include the costs associated with flying people to a remote site and putting them up in hotels and feeding them for several days.
- 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 sessions are usually held in special-purpose rooms where participants sit around horseshoe-shaped tables, as shown in Figure 6-6. These rooms are typically equipped with whiteboards. Other audiovisual tools may be used, such as magnetic symbols that can be easily rearranged on a whiteboard, flip charts, and computer-generated displays.

Figure 6-6 Illustration of the typical room layout for a JAD



Source: Adapted from Wood and Silver, 1995.

- **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.

3.1.5 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).**
- **Business Process Reengineering (BPR):** search for, and implementation of radical change in business processes to achieve breakthrough improvements in products and services.
- 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 your putting, driving accuracy, and sand shots will help, but the new competitive environment is simply not suited to your style of the game.

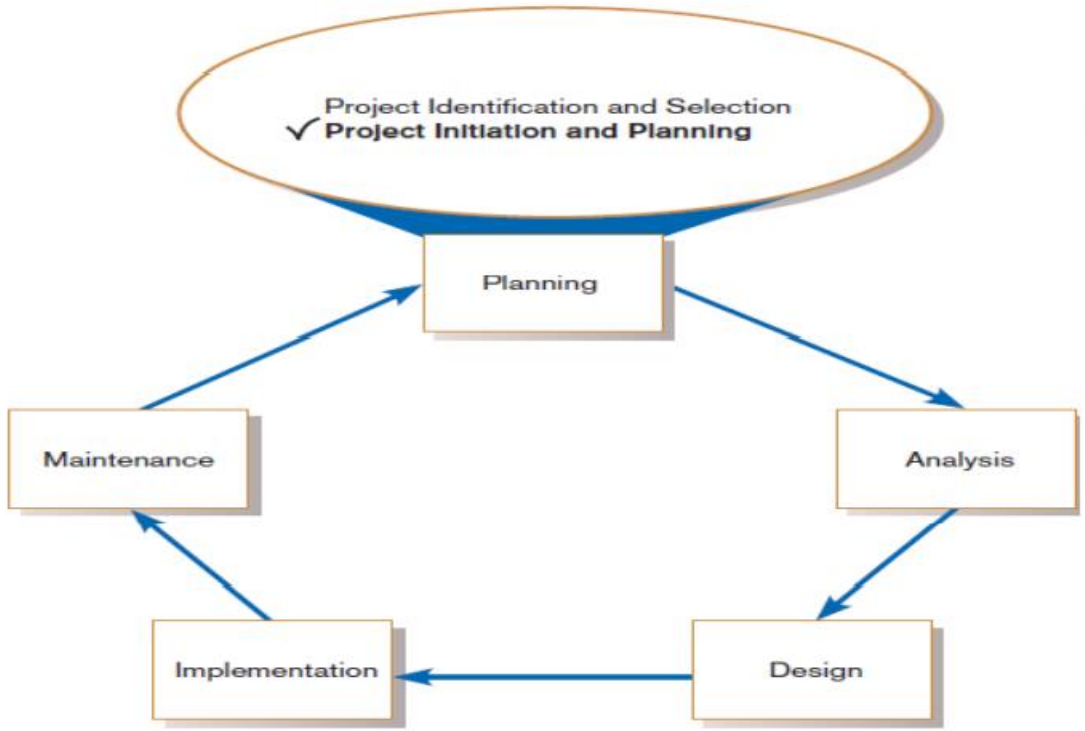


TABLE 5-1 Elements of Project Initiation

- Establishing the Project Initiation Team
- Establishing a Relationship with the Customer
- Establishing the Project Initiation Plan
- Establishing Management Procedures
- Establishing the Project Management Environment and Project Workbook
- Developing the Project Charter

FIGURE
Systems development life cycle with project initiation and planning highlighted

- Project planning, the second activity within PIP, is distinct from general information systems planning, which focuses on assessing the information systems needs of the entire organization.
- Project planning is the process of defining clear, discrete activities and the work needed to complete each activity within a single project.
- The objective of the project planning process is the development of a **Baseline Project Plan (BPP) and the Project Scope Statement (PSS).**
- The **BPP** becomes the foundation for the remainder of the development project.
- The **PSS** produced by the team clearly outlines the objectives and constraints of the project for the customer.

- As with the project initiation process, the size, scope, and complexity of a project will dictate the comprehensiveness of the project planning process and resulting documents.
- Further, numerous assumptions about resource availability and potential problems will have to be made.
- Analysis of these assumptions and system costs and benefits forms a business case.
- *Business case is the justification for an information system, presented in terms of the tangible and intangible economic benefits and costs and the technical and organizational feasibility of the proposed system.*
- *Baseline Project Plan (BPP) is a major outcome and deliverable from the project initiation and planning phase that contains the best estimate of a project's scope, benefits, costs, risks, and resource requirements.*

Feasibility Analysis

- Feasibility: The measure of how beneficial or practical an information system will be to an organization.
- **Feasibility analysis:** A feasibility analysis is the process by which feasibility is measured
- The scope and complexity of an apparently feasible project can change after the initial problems and opportunities are fully analyzed or after the system has been designed.
- A project that is feasible at one point in time may become infeasible at a later point in time
- A feasibility study assesses the operational, technical, and economic merits of the proposed project

Types of Feasibility Analysis

- **Operational feasibility** is a measure of how well the solution of problems or a specific solution will work in the organization. It is also a measure of how people feel about the system/project.
- **Technical feasibility** is a measure of the practicality of a specific technical solution and the availability of technical resources and expertise.
- **Schedule feasibility** is a measure of how reasonable the project timetable is.
- **Economic feasibility** is a measure of the cost-effectiveness of a project or solution. This is often called a *cost-benefit analysis*.

Assessing Economic Feasibility

- The purpose of assessing economic feasibility is to identify the financial benefits and costs associated with the development project
- Economic feasibility is often referred to as **cost–benefit analysis.**
- During project initiation and planning, it will be impossible for you to precisely define all benefits and costs related to a particular project.
- Yet it is important that you spend adequate time identifying and quantifying these items or it will be impossible for you to conduct an adequate economic analysis and make meaningful comparisons between rival projects.
- Here we will describe typical benefits and costs resulting from the development of an information system and provide several useful worksheets for recording costs and benefits. Additionally, several common techniques for making cost–benefit calculations are presented.
- These worksheets and techniques are used after each SDLC phase as the project is reviewed in order to decide whether to continue, redirect, or kill a project.

Determining Project Benefits

- An information system can provide many benefits to an organization.
- For example, a new or renovated information system can automate monotonous jobs and reduce errors; provide innovative services to customers and suppliers; and improve organizational efficiency, speed, flexibility, and morale.
- In general, the benefits can be viewed as being both tangible and intangible.
- **Tangible** benefits refer to items that can be measured in dollars and with certainty. Examples of tangible benefits might include reduced personnel expenses, lower transaction costs, or higher profit margins.

- Most tangible benefits will fit within the following categories:
    - Cost reduction and avoidance
    - Error reduction
    - Increased flexibility
    - Increased speed of activity
    - Improvement of management planning and control
    - Opening new markets and increasing sales opportunities
- *Tangible benefit A benefit derived from the creation of an information system that can be measured in dollars and with certainty*

- **Intangible** benefits refer to items that cannot be easily measured in dollars or with certainty.
- Intangible benefits may have direct organizational benefits, such as the improvement of employee morale, or they may have broader societal implications, such as the reduction of waste creation or resource consumption.
- Potential tangible benefits may have to be considered intangible during project initiation and planning because you may not be able to quantify them in dollars or with certainty at this stage in the life cycle.
- During later stages, such intangibles can become tangible benefits as you better understand the ramifications of the system you are designing.
- In this case, the BPP is updated and the business case revised to justify continuation of the project to the next phase.
- *Intangible benefit A benefit derived from the creation of an information system that cannot be easily measured in dollars or with certainty*

Determining Project Costs

- Similar to benefits, an information system can have both tangible and intangible costs.
- **Tangible** costs refer to items that you can easily measure in dollars and with certainty.
- From an IS development perspective, tangible costs include items such as hardware costs, labor costs, and operational costs including employee training and building renovations.
- Alternatively, **intangible** costs are items that you cannot easily measure in terms of dollars or with certainty.
- Intangible costs can include loss of customer goodwill, employee morale, or operational inefficiency.
- One goal of a cost–benefit analysis is to accurately determine the total cost of ownership (TCO) for an investment.
- TCO is focused on understanding not only the total cost of acquisition but also all costs associated with ongoing use and maintenance of a system.
- *Total cost of ownership (TCO) The cost of owning and operating a system, including the total cost of acquisition, as well as all costs associated with its ongoing use and maintenance.*

- **One-time costs** refer to those associated with project initiation and development and the start-up of the system. These costs typically encompass activities such as systems development, new hardware and software purchases, user training, site preparation, and data or system conversion.
- When conducting an economic cost–benefit analysis, a worksheet should be created for capturing these expenses. For very large projects, one-time costs may be staged over one or more years. In these cases, a separate onetime cost worksheet should be created for each year. This separation will make it easier to perform present value calculations.
- **Recurring costs** refer to those costs resulting from the ongoing evolution and use of the system. Examples of these costs typically include the following:

  - Application software maintenance
  - Incremental data storage expenses
  - Incremental communications
  - New software and hardware leases
  - Supplies and other expenses (e.g., paper, forms, data center personnel)

Definitions of Terms

- **Time value of money** (**TVM**): the concept that money available today is worth more than the same amount tomorrow
- **Discount rate**: the rate of return used to compute the present value of future cash flows (*the cost of capital*)
- **Present value**: the current value of a future cash flow
- **Net Present Value**
  - PVn = *present value* of Y dollars n years from now based on a *discount rate* of i.
  - NPV = sum of PVs across years.
  - Calculates *time value of money*

$$\mathrm{PV}_n = Y \times \frac{1}{(1+i)^n}$$

Break-even analysis

- The objective of the break-even analysis is to discover at what point (if ever) benefits equal costs (i.e., when breakeven occurs)
- To conduct this analysis, the NPV of the yearly cash flows are determined.
- Here, the yearly cash flows are calculated by subtracting both the one-time cost and the present values of the recurring costs from the present value of the yearly benefits.
- The overall NPV of the cash flow reflects the total cash flows for all preceding years.

$$\text{Break-Even Ratio} = \frac{\text{Yearly NPV Cash Flow} - \text{Overall NPV Cash Flow}}{\text{Yearly NPV Cash Flow}}$$

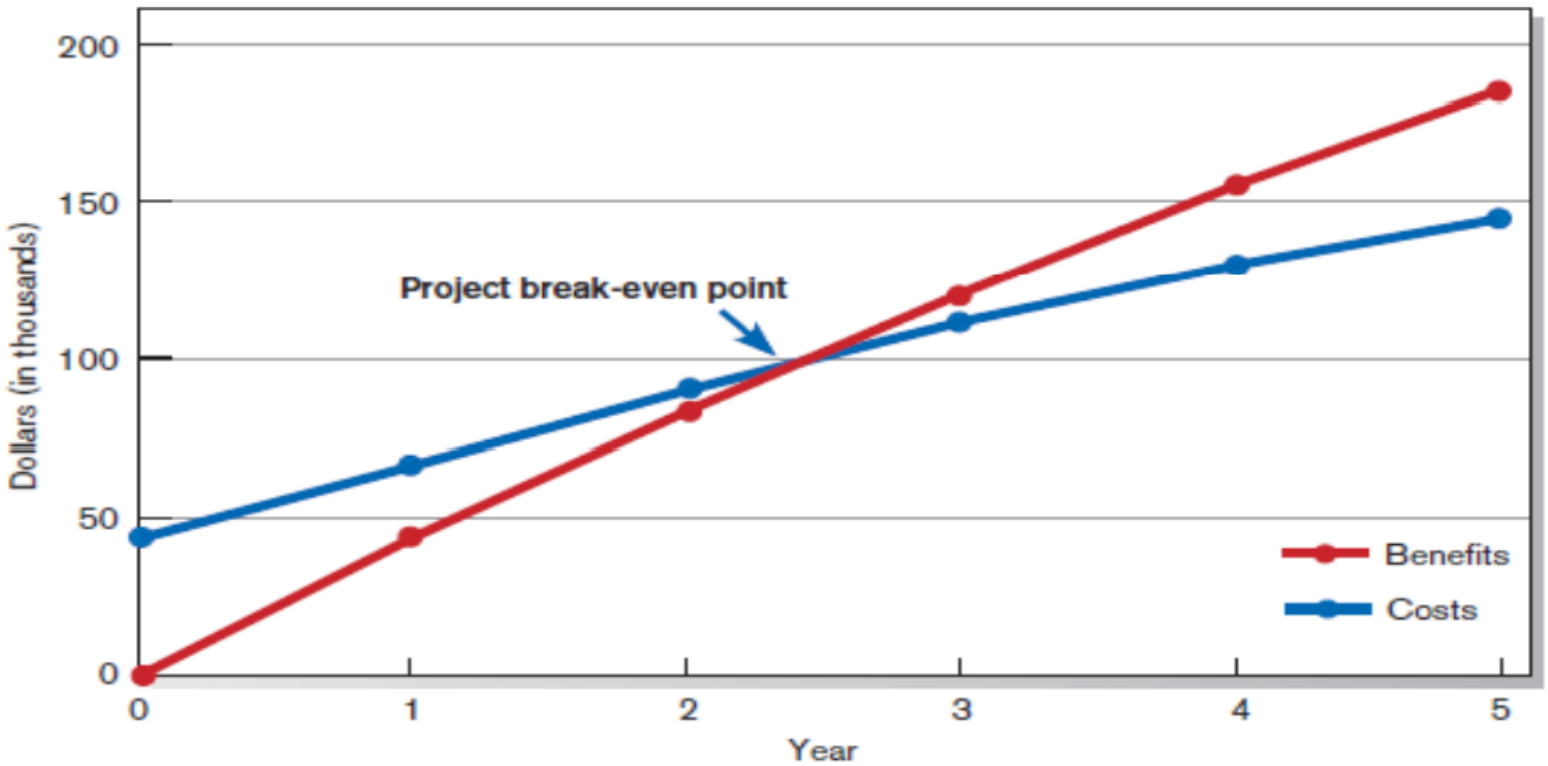


Figure
Break-even analysis for Customer Tracking System (Pine Valley Furniture)

Three Financial Measurements for Economic Feasibility

- **Net Present Value (NPV):** Use discount rate to determine present value of cash outlays and receipts
- **Return on Investment (ROI):** Ratio of cash receipts to cash outlays
- **Break-Even Analysis (BEA):** Amount of time required for cumulative cash flow to equal initial and ongoing investment

TABLE 5-6 Commonly Used Economic Cost–Benefit Analysis Techniques

Analysis Technique	Description
Net Present Value (NPV)	NPV uses a discount rate determined from the company's cost of capital to establish the present value of a project. The discount rate is used to determine the present value of both cash receipts and outlays.
Return on Investment (ROI)	ROI is the ratio of the net cash receipts of the project divided by the cash outlays of the project. Trade-off analysis can be made among projects competing for investment by comparing their representative ROI ratios.
Break-Even Analysis (BEA)	BEA finds the amount of time required for the cumulative cash flow from a project to equal its initial and ongoing investment.

Reviewing the Baseline Project Plan

- Before the next phase of the SDLC can begin, the users, management, and development group must review the BPP in order to verify that it makes sense. This review takes place before the BPP is submitted or presented to a project approval body, such as an IS steering committee or the person who must fund the project.
- The objective of this review is to ensure that the proposed system conforms to organizational standards and that all relevant parties understand and agree with the information contained in the BPP. A common method for performing this review (as well as reviews during subsequent life cycle phases) is called a **structured walk-through.**

Structured Walkthroughs

- A peer-group review of any product created during the system development process
- Experience has shown that walk-throughs are a very effective way to ensure the quality of an information system and have become a common day-to-day activity for many systems analysts
- Roles: coordinator, presenter, user, secretary, standard-bearer, maintenance oracle
- Can be applied to BPP, system specifications, logical and physical designs, program code, test procedures, manuals and documentation

Roles

- **Coordinator.** This person plans the meeting and facilitates a smooth meeting process. This person may be the project leader or a lead analyst responsible for the current life cycle step.
- **Presenter.** This person describes the work product to the group. The presenter is usually an analyst who has done all or some of the work being presented.
- **User**. This person (or group) makes sure that the work product meets the needs of the project's customers. This user would usually be someone not on the project team.
- **Secretary.** This person takes notes and records decisions or recommendations made by the group. This may be a clerk assigned to the project team or it may be one of the analysts on the team.

- **Standards bearer.** The role of this person is to ensure that the work product adheres to organizational technical standards. Many larger organizations have staff groups within the unit responsible for establishing standard procedures, methods, and documentation formats. These standards bearers validate the work so that it can be used by others in the development organization.
- **Maintenance oracle.** This person reviews the work product in terms of future maintenance activities. The goal is to make the system and its documentation easy to maintain.

3.1.7 Requirement 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.

(1) Continual User Involvement

- 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 is a key part of many Agile approaches, but it was also a key part of Rapid Application Development.

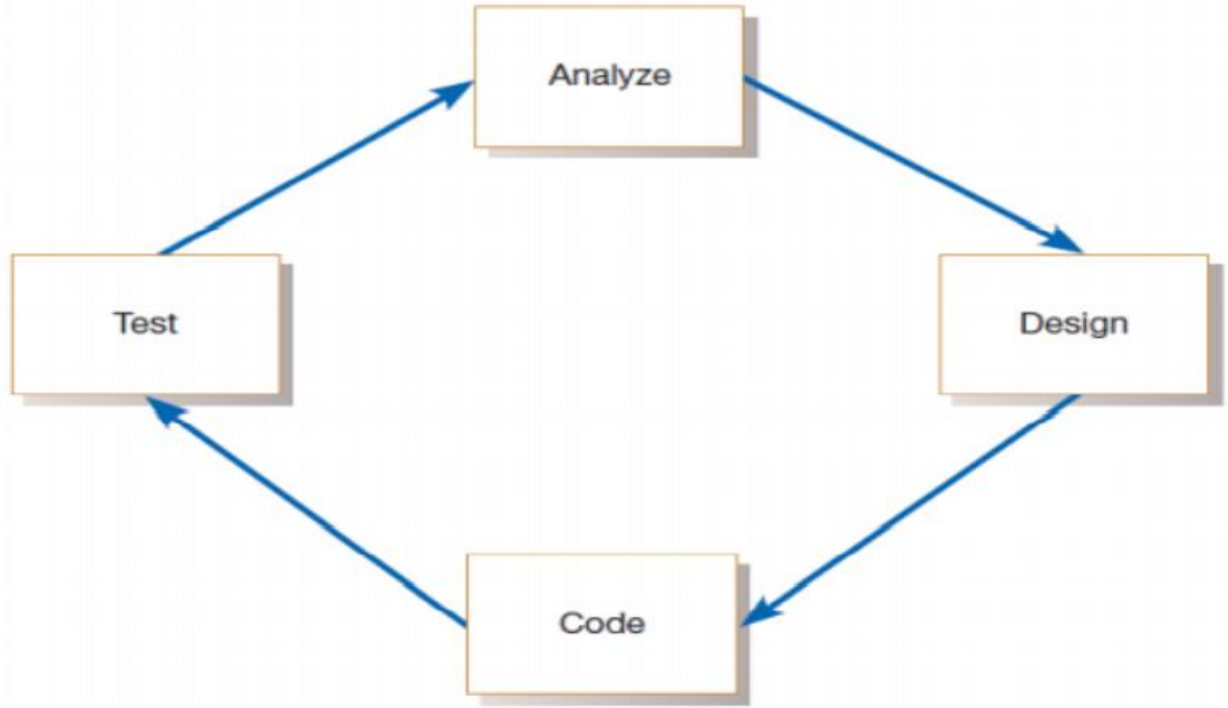


FIGURE 6-9
The iterative analysis–design–code–test cycle

(2) 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).

Agile Usage-Centered Design Steps

Gather group of programmers, analysts, users, testers, facilitator.

- Document complaints of current system.
- Determine important user roles.
- Determine, prioritize, and describe tasks for each user role.
- Group similar tasks into interaction contexts.
- Associate each interaction context with a user interface for the system, and prototype the interaction context.
- Step through and modify the prototype.

(3) The Planning Game From eXtreme Programming

- eXtreme Programming is an approach to software development put together by Kent Beck (Beck and Andres, 2004).
- 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 and

(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.

3.2 System Process Requirements

3.2.1 Introduction,

3.2.2 Process Modeling,

3.2.3 Data Flow Diagramming Mechanics,

3.2.4 Using DFD In The Analysis Process,

3.2.5 Modeling Logic With Decision Tables

3.2 System Process Requirements

3.2.1 Introduction

- We will be focusing 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.

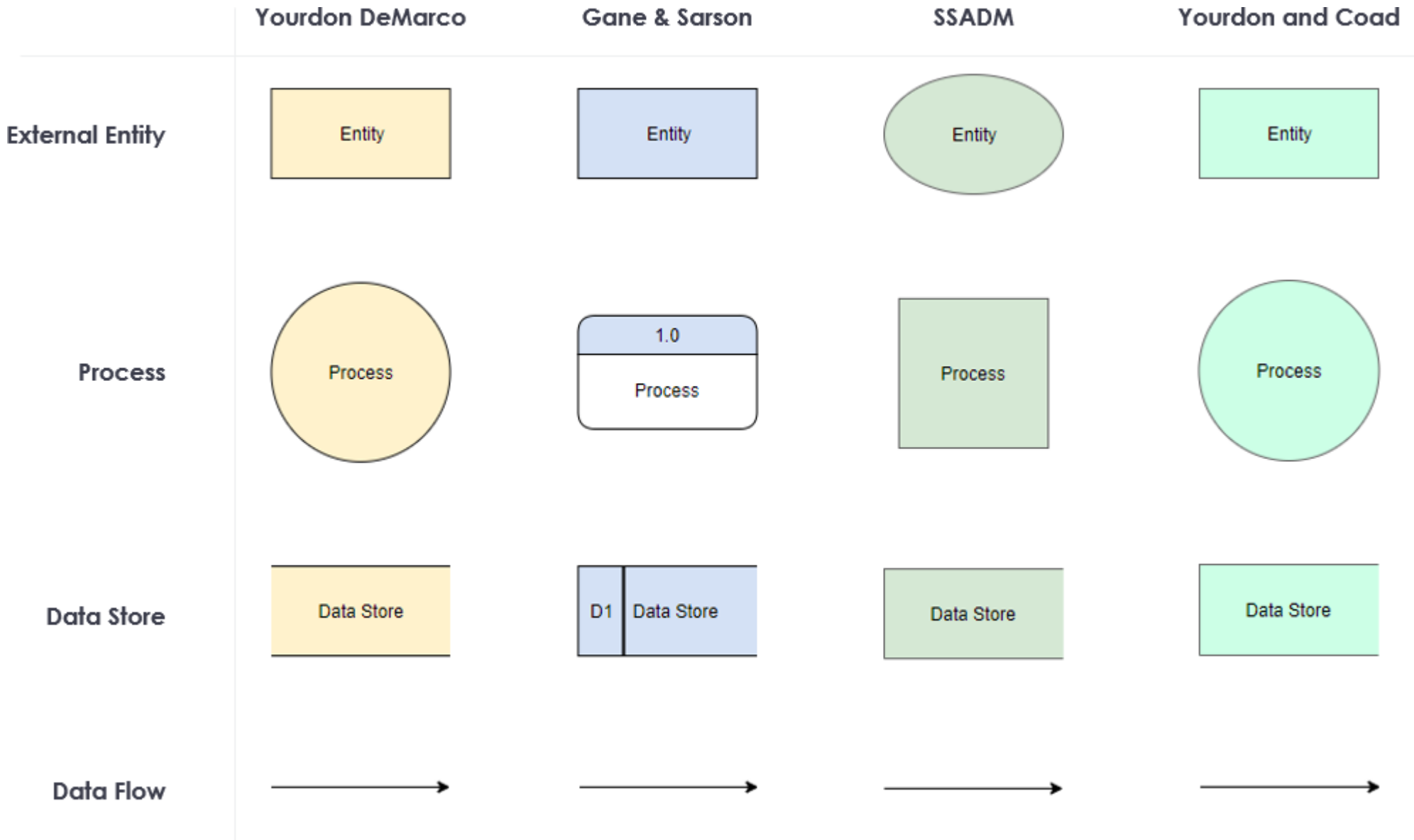
3.2.2 Process Modeling

- **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.
- Data-flow diagramming is one of several structured analysis techniques used to increase software development productivity.

3.2.3 Data Flow Diagramming Mechanics

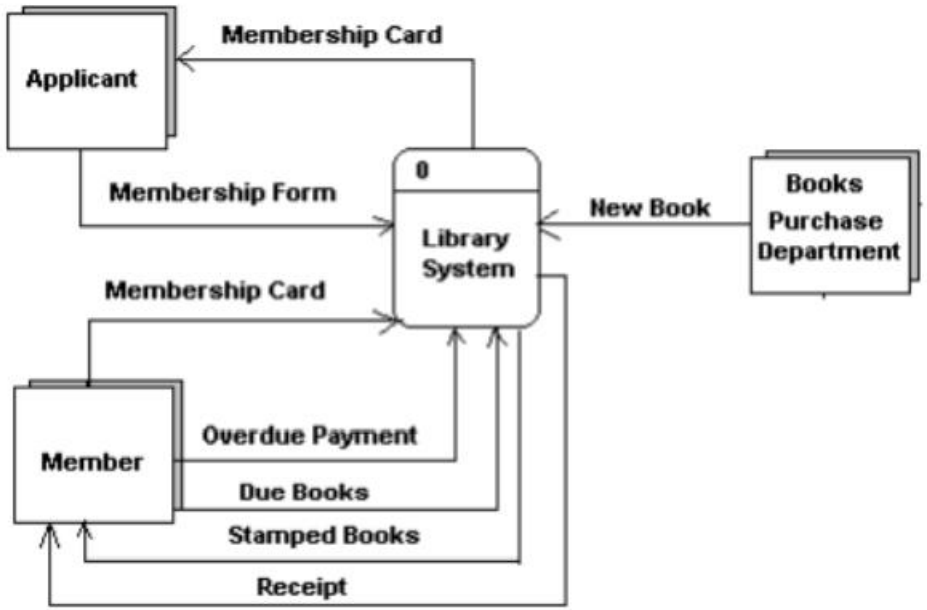
- DFDs are versatile diagramming tools. With only four symbols, you can use DFDs to represent both physical and logical information systems. DFDs are not as good as flowcharts for depicting (to represent or show something in a picture or story) the details of physical systems.
- There are two different standard sets of DFD symbols; each set consists of four symbols that represent the same things: data flows, data stores, processes, and sources/sinks (or external entities).
- 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. Sources and sinks are outside the system we are studying

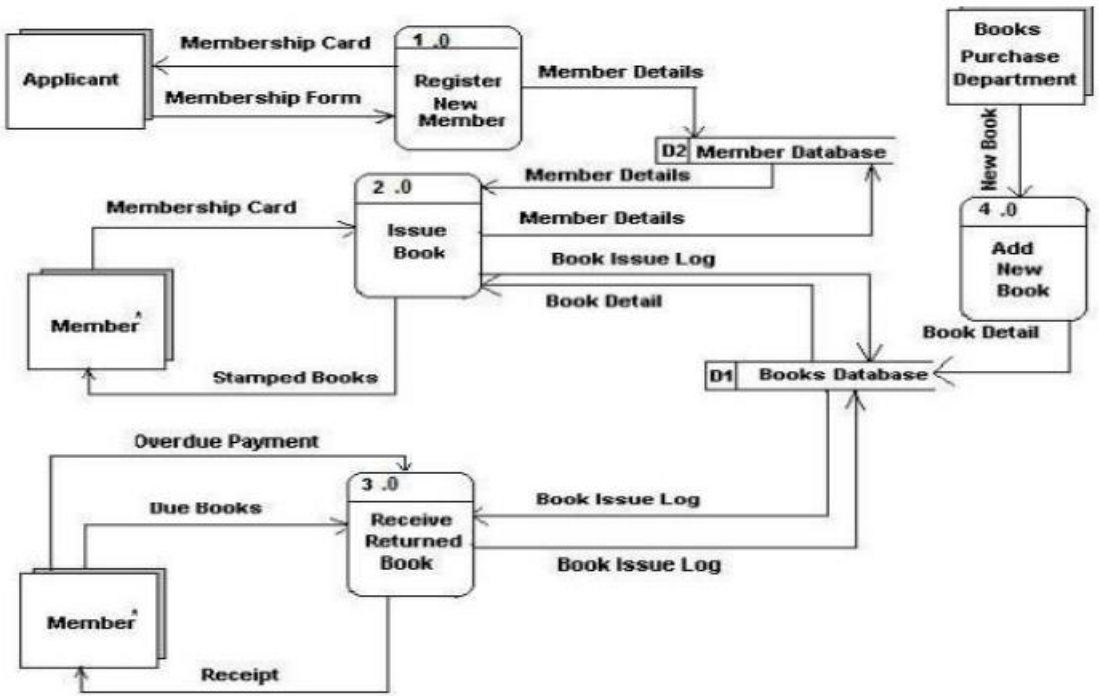

Yourdon DeMarco
Gane & Sarson
SSADM
Yourdon and Coad
External Entity
Entity
Entity
Entity
Entity
Process
Process
1.0
Process
Process
Process
Data Store
Data Store
D1
Data Store
Data Store
Data Store
Data Flow

- The name represents the aggregation of all the individual elements of data moving as part of one packet, that is, all the data moving together at the same time.
- Sources/Sinks are always outside the information system and define the boundaries of the system. Data must originate outside a system from one or more sources, and the system must produce information to one or more sinks (these are principles of open systems, and almost every information system is an open system).
- If any data processing takes place inside the source/sink, it is of no interest because this processing takes place outside the system we are diagramming.
- The figure above shows two different sets of symbols developed by DeMacro and Yourdon and Gane and Sarson. The set of symbols we will use here was deviced by Gane and Sarson.
- According to Gane and Sarson, rounded rectangles represent process or work to be done, squares represent external agents, open-ended boxes represent data store (sometimes called files or databases), and arrows represent data flows or inputs and outputs to and from the processes.

- Process is the work or actions performed on data so that they are transformed, stored or distributed. Data store is the data at rest (inside the system) that may take the form of many different physical representations. External entity (source/sink) is the origin and/or destination of data. Data flow represents data in motion, moving from one place in a system to another.



A sample Context Diagram



DFD

CONTEXT DIAGRAMS focus on how external entities interact with your system. It's the most basic form of a data flow diagram, providing a broad view of the system and external entities in an easily digestible way. Because of its simplicity, it's sometimes called a level 0 data flow diagram.

DATA FLOW DIAGRAMS contain additional information about a system that a context diagram doesn't.

Developing DFD

- The information system is represented as a DFD in Figure 7-4. The highest-level view of this system, shown in the figure, is called a **context diagram**. You will notice that this context diagram contains only one process, no data stores, four data flows, and three sources/sinks. The single process, **labeled 0,** represents the entire system; all context diagrams have only one process, labeled 0. The sources/sinks represent the environmental boundaries of the system. Because the data stores of the system are conceptually inside one process, data stores do not appear on a context diagram.

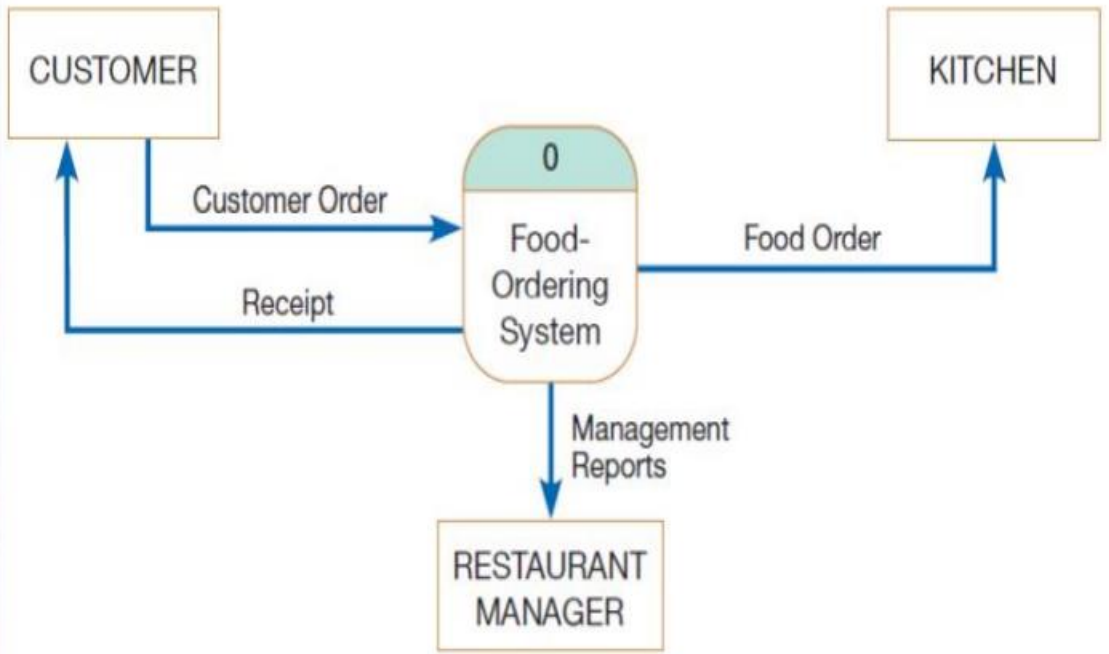


FIGURE 7-4
Context diagram of Hoosier Burger's food-ordering system

- As you can see in Figure 7-5, we have identified four separate processes. The main processes represent the major functions of the system, and these major functions correspond to actions such as the following:

1. Capturing data from different sources (e.g., Process 1.0)
2. Maintaining data stores (e.g., Processes 2.0 and 3.0)
3. Producing and distributing data to different sinks (e.g., Process 4.0)
4. High-level descriptions of data transformation operations (e.g., Process 1.0)

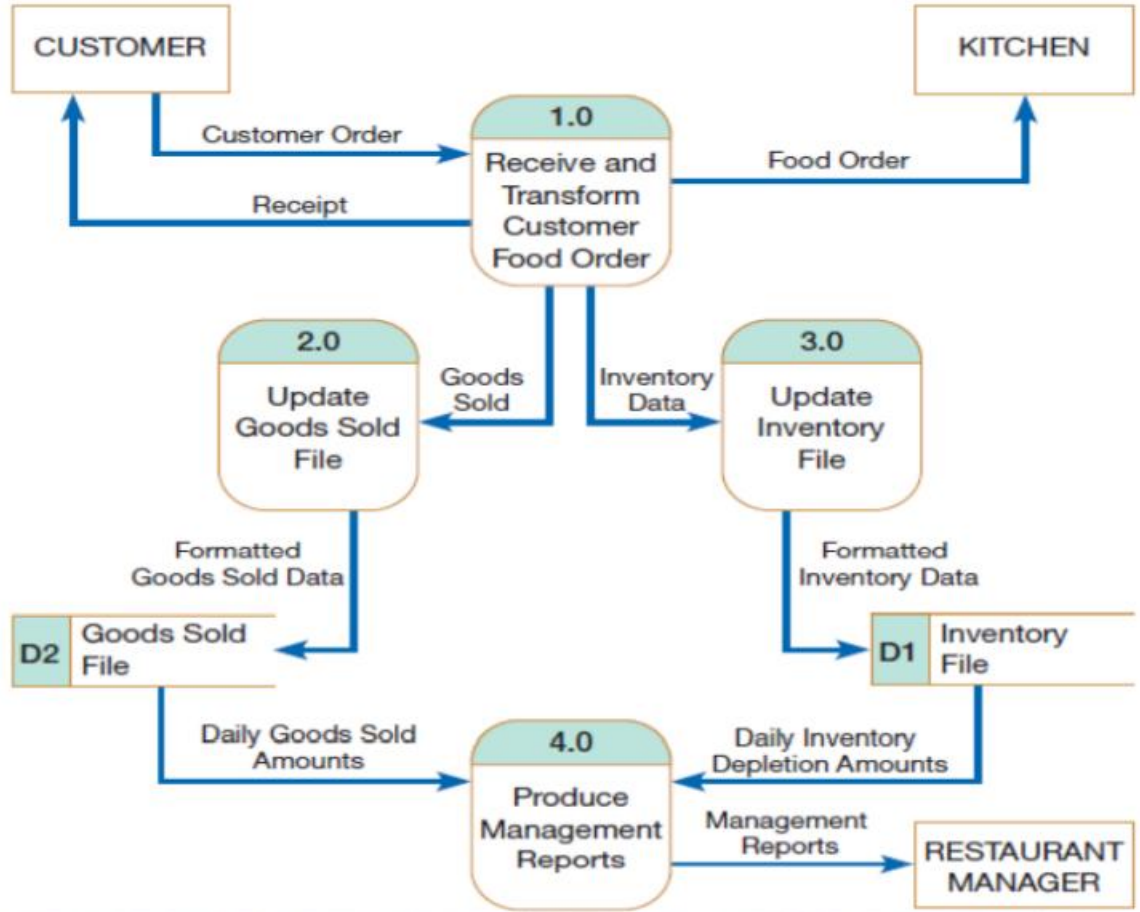


FIGURE 7-5
Level-0 DFD of Hoosier Burger's food-ordering system

DFD guidelines & rules II

TABLE 8-2 Rules Governing Data Flow Diagramming

Process:

A. No process can have only outputs. It is making data from nothing (a miracle). If an object has only outputs, then it must be a source.

B. No process can have only inputs (a black hole). If an object has only inputs, then it must be a sink.

C. A process has a verb phrase label.

Data Store:

D. Data cannot move directly from one data store to another data store. Data must be moved by a process.

E. Data cannot move directly from an outside source to a data store. Data must be moved by a process which receives data from the source and places the data into the data store.

F. Data cannot move directly to an outside sink from a data store. Data must be moved by a process.

G. A data store has a noun phrase label.

Source/Sink:

H. Data cannot move directly from a source to a sink. It must be moved by a process if the data are of any concern to our system. Otherwise, the data flow is not shown on the DFD.

I. A source/sink has a noun phrase label.

Data Flow:

J. A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however, by two separate arrows since these happen at different times.

K. A fork in a data flow means that exactly the same data goes from a common location to two or more different processes, data stores, or sources/sinks (this usually indicates different copies of the same data going to different locations).

L. A join in a data flow means that exactly the same data comes from any of two or more different processes, data stores, or sources/sinks to a common location.

M. A data flow cannot go directly back to the same process it leaves. There must be at least one other process which handles the data flow, produces some other data flow, and returns the original data flow to the beginning process.

N. A data flow to a data store means update (delete or change).

O. A data flow from a data store means retrieve or use.

P. A data flow has a noun phrase label. More than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

Adapted from Celko, 1987

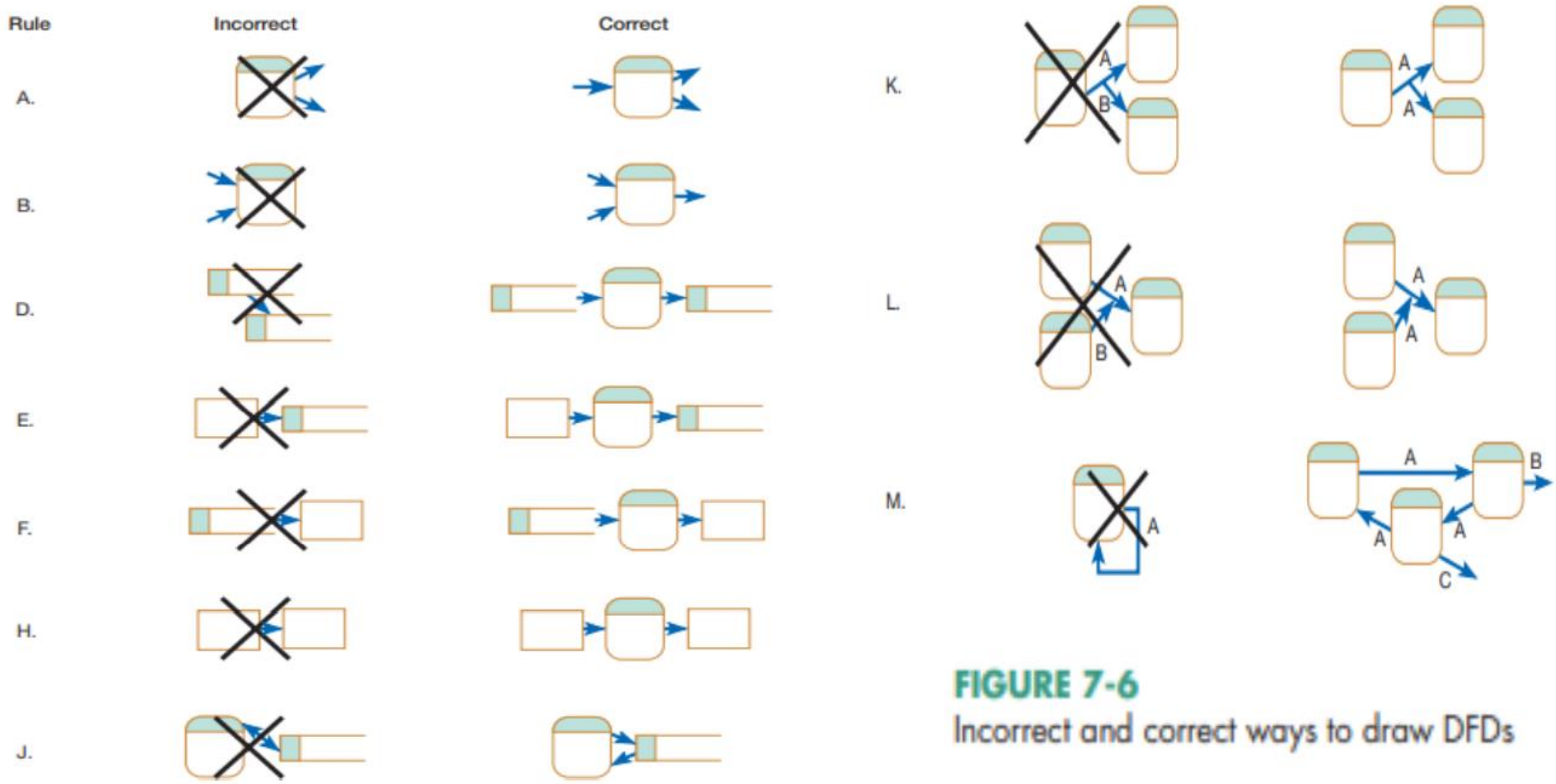


FIGURE 7-6
Incorrect and correct ways to draw DFDs

Four Different Types of DFDs

- **Current Physical**
  - Process labels identify technology (people or systems) used to process the data.
  - Data flows and data stores identify actual name of the physical media.
- **Current Logical**
  - Physical aspects of system are removed as much as possible.
  - Current system is reduced to data and processes that transform them.
- **New Logical**
  - Includes additional functions.
  - Obsolete functions are removed.
  - Inefficient data flows are reorganized.
- **New Physical**
  - Represents the physical implementation of the new system.

Physical & Logical DFD

- Consider the figure 1 It is clear from the figure that orders are placed, orders are received, the location of ordered parts is determined and delivery notes are dispatched along with the order.

Fig 1

It does not however tell us how these things are done or who does them. Are they done by computers or manually and if manually who does them ? A logical DFD of any information system is one that models what occurs without showing how it occurs

- A physical DFD shows, how the various functions are performed? Who does them? Consider the following figure:

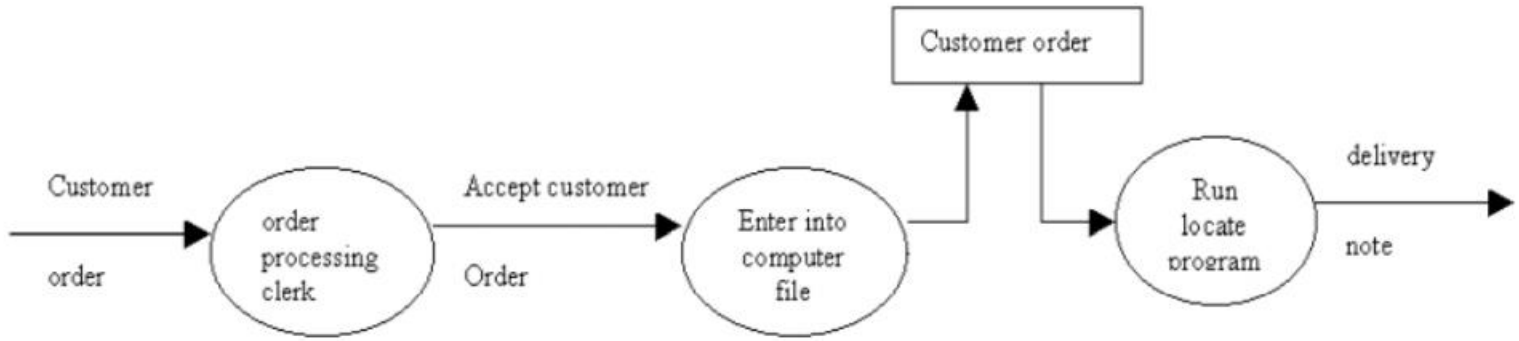
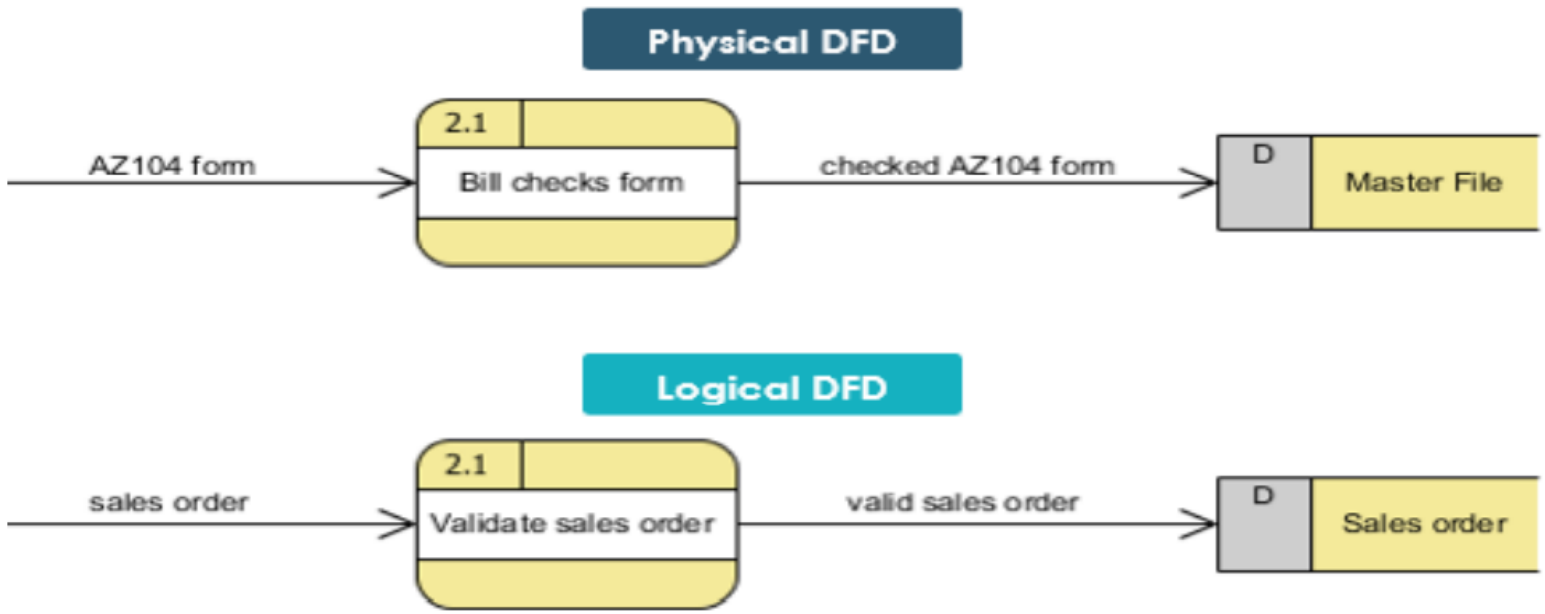


Fig 2

The figure 2 is opposite, it shows the actual devices that perform the functions. Thus there is an "order processing clerk", an "entry into computer file" process and a "run locate program" process to locate the parts ordered. DFD(s) that shows how things happen, or the physical components are called physical DFD(s). Typical processes that appear in physical DFDs are methods of data entry, specific data transfer or processing methods

Physical and Logical DFD: Example 1

- Physical DFD specifies actual flow of physical documentation, while logical DFD only focus on the information flow in business term.



3.2.4 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

1. **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.
2. **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).

3. 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.

4. 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.

5. 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.

3.2.5 Modeling Logic With Decision Tables

- Logic Modeling:

➢ 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.

➢ A logic model has four components:

1. **Needs:** are about the problem and why its important to address it. What issue are we trying to address?
2. **Inputs:** are the things that contribute to addressing the need (usually a combination of money, time, expertise and resources). What resources are we investing?
3. **Activities:** describe the things that the inputs allow to happen. What are we doing with these resources.
4. **Outcomes:** are usually expressed in terms of measures of success. What differences are we hoping to make?

Each process on the lowest level DFD will be represented by one or more of the following:

- Structured English representation of process logic.
- Decision Tables representation.
- Sequence diagram.
- Activity diagram
- Decision Trees
- State-transition diagrams

- Decision tables are a concise / short / brief visual representation for specifying which actions to perform depending on given conditions.
- They are algorithms whose output is a set of actions.
- A **decision table** is an excellent tool to use in both testing and requirements management.
- **Decision Table** is a structured exercise to formulate requirements when dealing with complex business rules.
- **Decision Tables** are used to model complicated logic.
- Lets take an example scenario for an ATM where a decision table would be of use.
- A customer requests a cash withdrawal. One of the business rules for the ATM is that the ATM machine pays out the amount if the customer has sufficient funds in their account or if the customer has the credit granted.

- This simple example of a business rule is quite complicated to describe in text. A decision table makes the same requirements clearer to understand:

Conditions	R1	R2	R3
Withdrawal amount <= balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

In a decision table, conditions are usually expressed as true (T) or false (F). Each column in the table corresponds to a rule in the business logic that describes the unique combination of circumstances that will result in the actions.

- Decision tables can be used in all situations where the outcome depends on the combinations of different choices, and that is usually very often.
- In many systems there are tons of business rules where decision tables add a lot of value.
- **Steps to create decision tables**
- **Step 1 – Analyze the requirement and create the first column**
- Requirement: “Withdrawal is granted if requested amount is covered by the balance or if the customer is granted credit to cover the withdrawal amount”.
- Express conditions and resulting actions in a list so that they are either TRUE or FALSE.
- In this case there are two conditions, “withdrawal amount <= balance” and “credit granted”.

- There is one result, the withdrawal is granted.

Conditions
Withdrawal amount <=balance
Credit granted
Actions
Withdrawal granted

- **Step 2: Add Columns**
- Calculate how many columns are needed in the table
- The number of columns depends on the number of conditions and the number of alternative for each condition.
- Number of column that is needed:

Number of conditions	Number of Columns
1	2
2	4
3	8
4	16
5	32

- **Step 2: Add Columns**
- To fill True (T) and False (F) for the conditions, the simplest way is:
- Row 1: TF
- Row 2: TTFF
- Row 3: TTTTFFFF
- For each row, there is twice as many T and F as the previous line.

Conditions				
Withdrawal Amount <= Balance	T	F	T	F
Credit granted	T	T	F	F
Actions				
Withdrawal granted				

- **Step 3: Reduce the table**
- Make insignificant values with "-". If the requested amount is less than or equal to the account balance it does not matter if credit is granted.
- In the next step, you can delete the column that have become identical.

Conditions				
Withdrawal amount<=balance	T	F	T	F
Credit granted	-	T	-	F
Actions				
Withdrawal granted				

- Check for invalid combinations.
- Invalid combinations are those that cannot happen, for example, that someone is both an infant and senior. Mark them somehow, e.g. with "X". In this example, there are no invalid combinations.
- Finish by removing duplicate columns. In this case, the first and third column are equal therefore one of them is removed.

- **Step 4: Determining actions**
- Enter actions for each column in the table. Name the columns (the rules) be named R1/ Rule 1, R2/Rule 2 and so on, but you can also give them descriptive names.

Conditions			
Withdrawal amount<=balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

Advantages and disadvantage of decision tables

- One advantage of using decision table is that they make it possible to detect combinations of decisions that would otherwise not have been found and therefore not tested or developed.
- The requirements become much clearer and you often realize that some requirements are illogical, something that is hard to see when the requirements are only expressed in text.
- A disadvantage of the technique is that a decision table is not equivalent to complete test cases containing step by step instructions of what to do in what order. When this level of data is required, the decision table has to be further detailed into test cases.

3.3 System Data Requirements

3.3.1 Introduction,

3.3.2 Conceptual Data Modeling,

3.3.3 Gathering Information For Conceptual Data Modeling,

3.3.4 Introduction To ER Modeling,

3.3.5 Conceptual Data Modeling And The ER Model,

3.3.6 Representing Super Types And Sub-Types,

3.3.7 Business Rules,

3.3.8 Role Of Packaged Conceptual Data Models- Database Patterns

3.3 System Data Requirements

3.3.1 Introduction

- Structuring system and database requirements concentrates on the definition, structure and relationships within data.
- The characteristics of data captured during data modeling are crucial in the design of databases, programs, computer screens and printed reports. This information is essential in ensuring data integrity in an information system.
- The most common format used for data modeling in entity-relationship diagramming.
- Data modeling explains the characteristics and structure of data independent of how the data may be stored in computer memories and is usually developed iteratively.

- The most common format used for data modeling is **entity-relationship (E-R) diagramming.**
- A similar format used with object-oriented analysis and design methods is class diagramming, which is included in a special section at the end of this chapter on the object-oriented development approach to data modeling.
- 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. Information system (IS) planners use this preliminary data model to develop an enterprise-wide data model with very broad categories of data and little detail

3.3.2 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 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 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) drawing tools such as IBM's Rational products or Microsoft Visio are also popular.

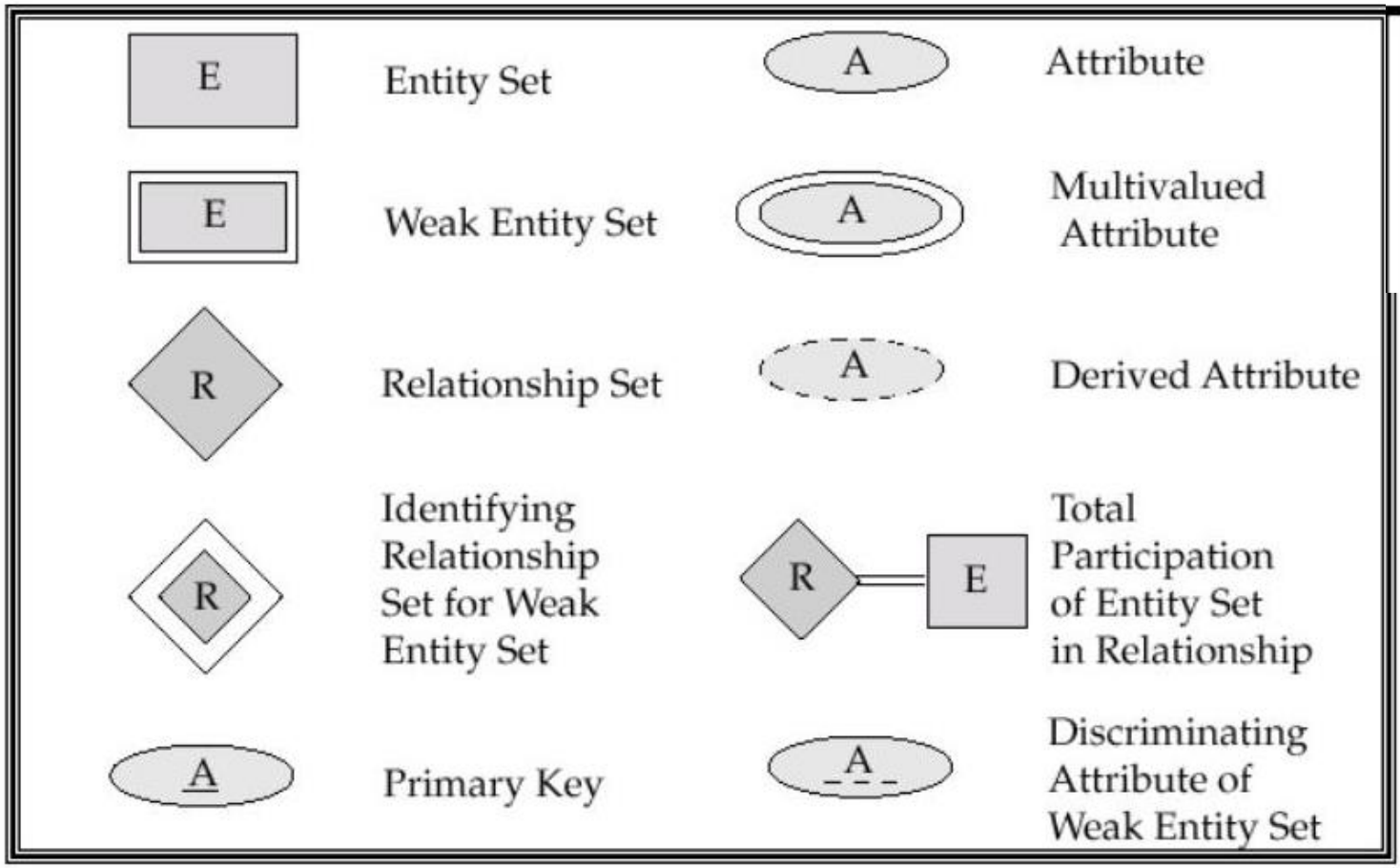
1. The primary deliverable from the conceptual data modeling step within the analysis phase is an E-R diagram.
2. The other deliverable from conceptual data modeling is a full set of entries about data objects that will be stored in the project dictionary, repository, or data modeling software. The repository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

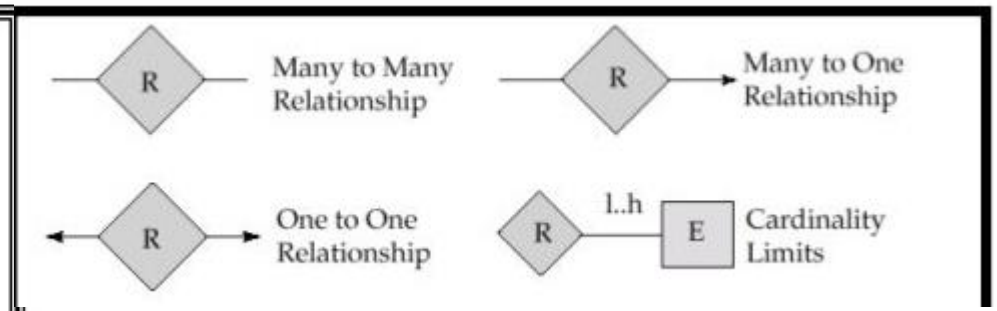
3.3.4 Introduction To ER Modeling

- An E-R data model is a detailed, logical representation of the data for an organization or for a business area.
- An entity is a person, place, object, event, or concept in the user environment about which the organization wishes to maintain data.
- An entity has its own identity, which distinguishes it from other entities.
- There is an important distinction between **entity types** and **entity instances**.
- An **entity type** is a single occurrence of an entity type.
- For example, there is usually one EMPLOYEE type but there may be hundreds of instances of this entity type stored in a database.
- Each entity type has a set of attributes associated with it.
- For example, for an entity STUDENT we have such attributes like: STUDENT NO, NAME, ADDRESS, PHONE NO.
- Every entity must have an attribute or set of attributes that distinguishes one instance from other instances of the same type – and that is called a **candidate key**.
- A **primary key** is a candidate key that has been selected to be used as the identifier for the entity type.
- For each entity the name of the primary key is underlined on an E-R diagram.

- **Entity Relationship Modeling (ER Modeling)** is a graphical approach to database design.
- It uses Entity/Relationship to represent real world objects.
- An entity is a thing or object in real world that is distinguishable from surrounding environment. For example each employee of an organization is a separate entity.
- It is a technique used in database design that helps describe the relationship between various entities of an organization. Terms used in E-R model are:
- **Entity** – It specifies distinct real world items in an application. For example: vendor, item, student course, teachers etc.
- **Relationship** – They are the meaningful dependencies between entities. For example, vendor supplies items, teacher teaches courses, then supplies and teachers are relationships.
- **Attributes** – It specifies the properties of relationships. For example, vendor code, student name.

- **Entity-relationship diagram (ERD)** are essential to modeling anything from simple to complex databases, but the shapes and notations used can be very confusing.
- There are various types of symbols used for ER diagram, some of well-defined symbols are listed below;





Elements of ER Diagram

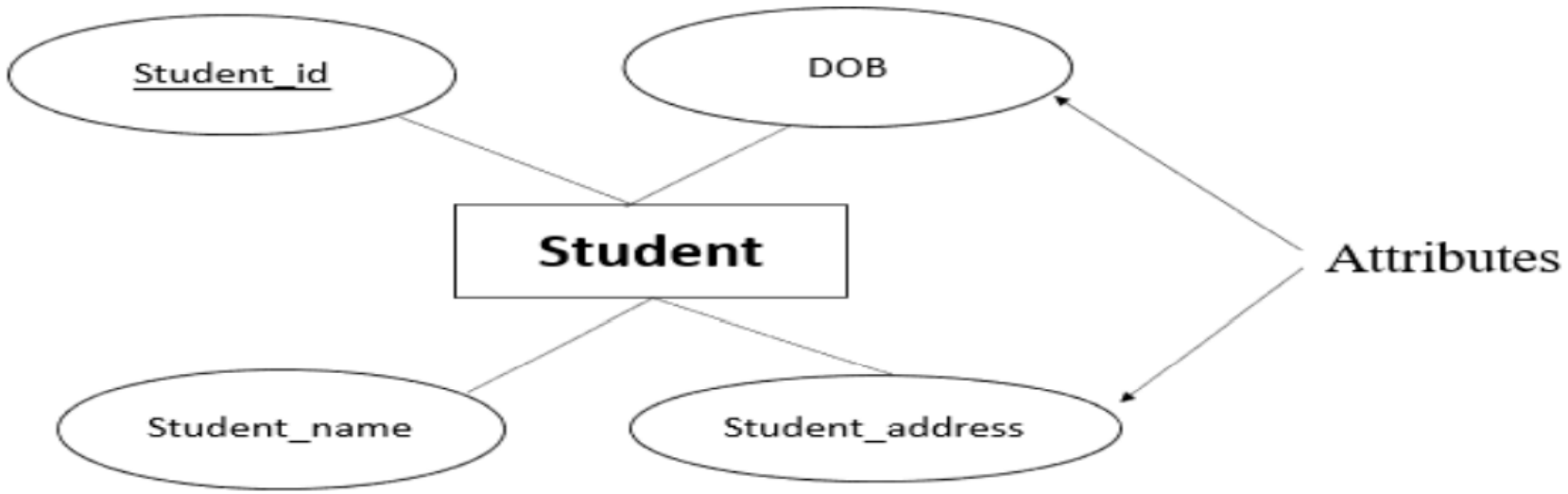
- **ENTITY:**
- An entity is a real-world thing either living or non-living that is easily recognizable and non-recognizable.
- An entity can be place, person, object, event or a concept, which stores data in the database.
- The characteristics of entities are must have an attribute, and a unique key.
- For example: Tuple1/ row1 / record1 contains information about Hari (id, name, age, address) which has existence in real world. So, the tuple1 is an entity. So, we may say each tuple is an entity. Let "Hari" is a particular member of entity type student.

Entity

- **ENTITY TYPE:**
- It is collection of entities having common attribute.
- As in student table, each row is an entity and has common attributes.
- So, Student is a entity type which contains entities having attributes id, name and age.
- Also each entity type in a database is described by a name and a list of attribute.
- For example: Collection of entities with similar properties.

Entity type

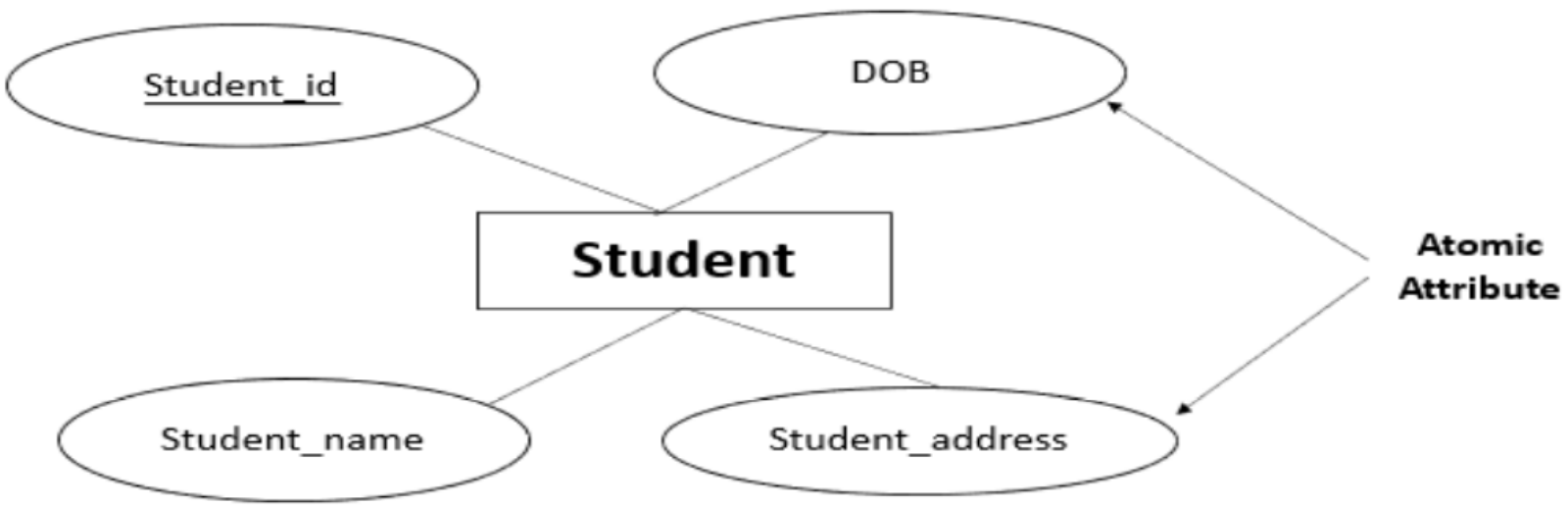
- **Attributes:**
- Attributes are the properties which define the entity type. For example, student_id, student_name, DOB, age, address, mobile_no etc. are the attributes which define entity type student.
- In ER diagram, attribute is represented by an oval.



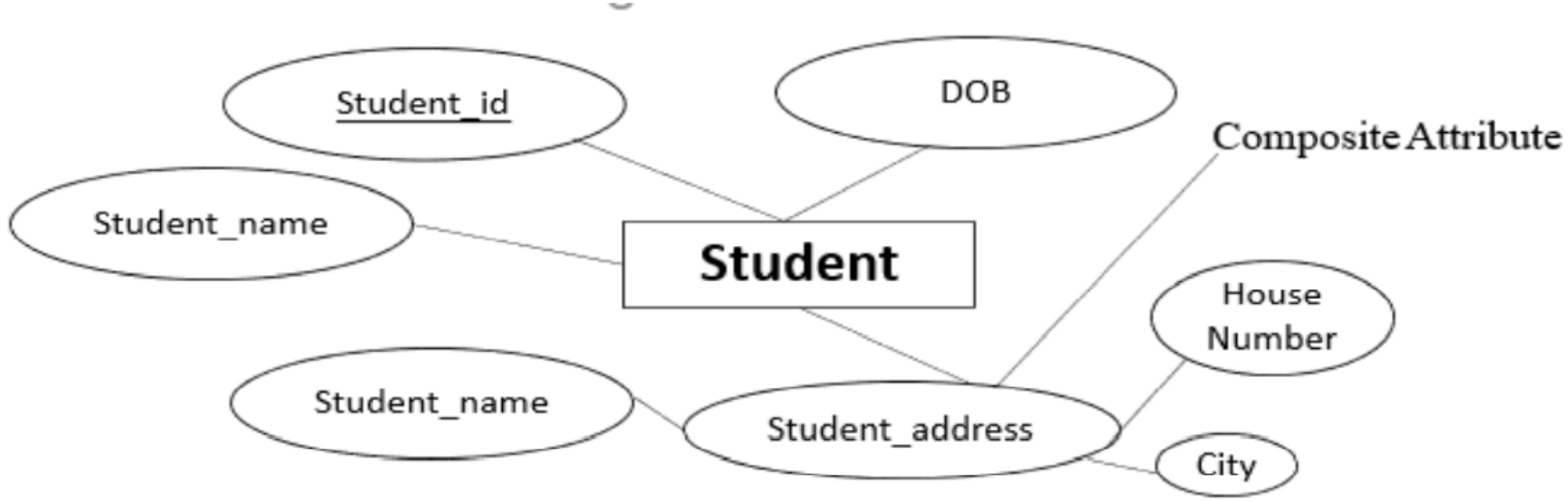
Types of Attributes

A. Atomic Vs. Composite attributes

- An attribute that cannot be divided into smaller independent attribute is known as **atomic attribute**. For example: Assume, student is an entity and its attributes are name, age, DOB, address and phone number.
- Here the student_id, DOB attribute of students (entity) cannot further divide. In this example, Student_id and DOB are atomic attribute.

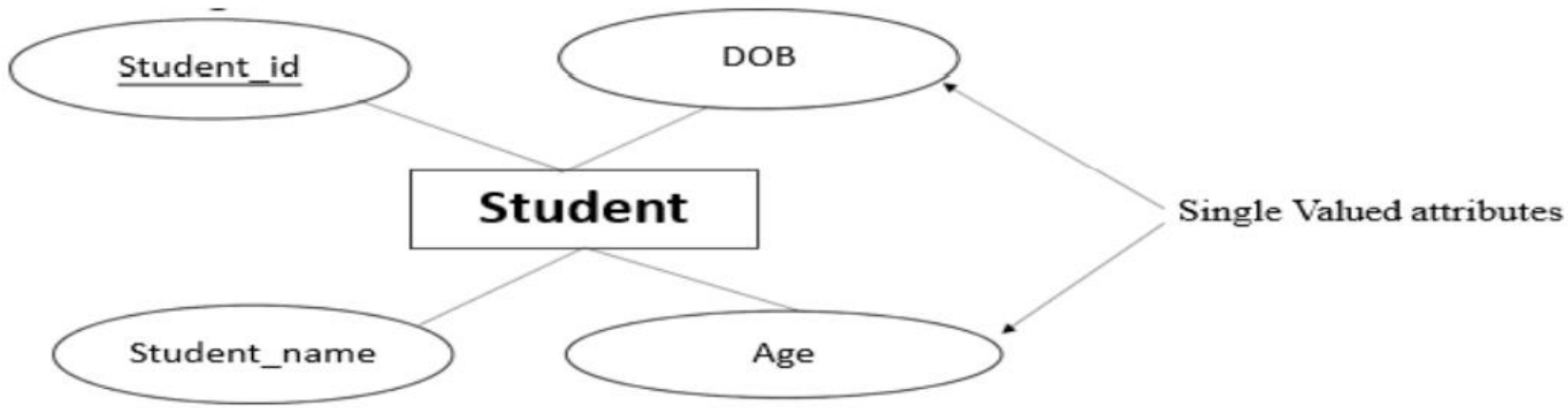


- An attribute that can be divided into smaller independent attribute is known as **composite attribute**. For example: Assume, student is an entity and its attributes are student_id, name, age, DOB, address and phone number.
- Here the address (attribute) of student (entity) can be further divided into house number, city and so on. In this example, address is composite attribute.

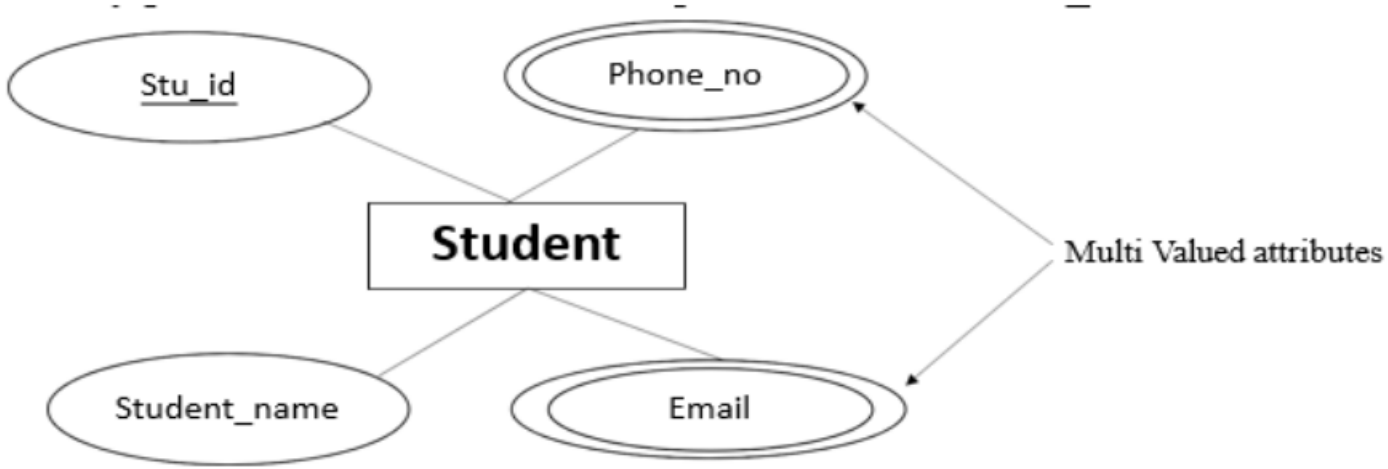


B. Single Valued Vs. Multi Valued attributes

- An attribute that has only single value for an entity is known as single valued attribute. For example: Assume, student is an entity and its attributes are Stu_id, Name, age, DOB, address and phone number.
- Here the age and DOB(attribute) of student (entity) can have only one value. In this example, age and DOB are single valued attribute.

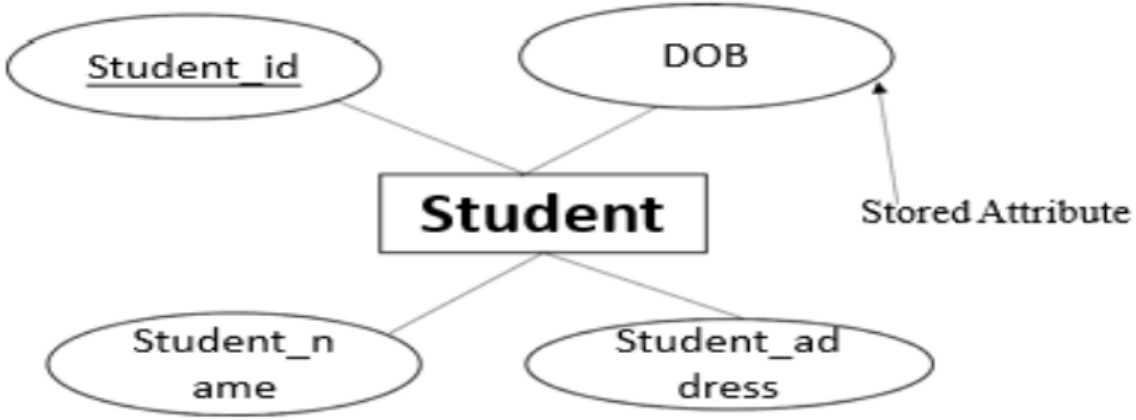


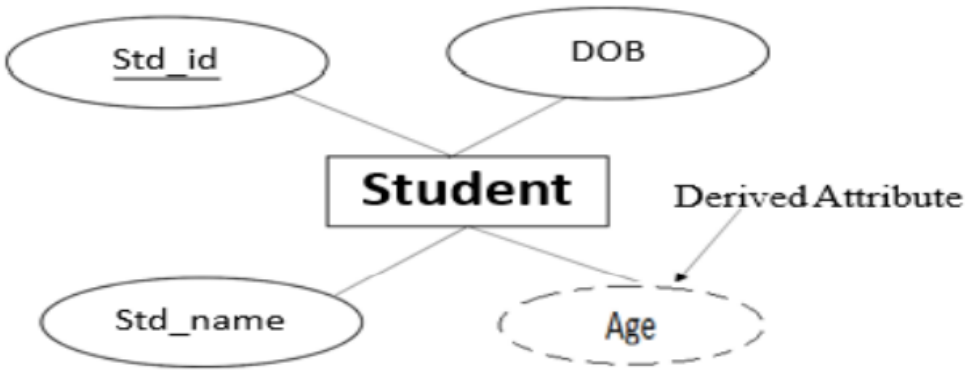
- An attribute that can have multiple values for an entity is known as **multi valued attribute.** For example: Assume, Student is an entity and its attributes are Stu_id, Name, age, DOB, address, email and phone no.
- Here the phone no and Email (attribute) of Student (entity) can have multiple values because a student may have many phone numbers. In this example, Email and Phone_no are multi valued attributes.



C. Stored Vs. Derived attributes

- An attribute that cannot be derived from another attribute and we need to store their value in database is known as **stored attribute**. For example, DOB cannot derive from age of student.
- An attribute that can be derived from another attribute and we do not need to store their value in the database due to dynamic nature is known as **derived attribute.**
- It is denoted by dotted oval. For example, age can be derived from birth date of student.

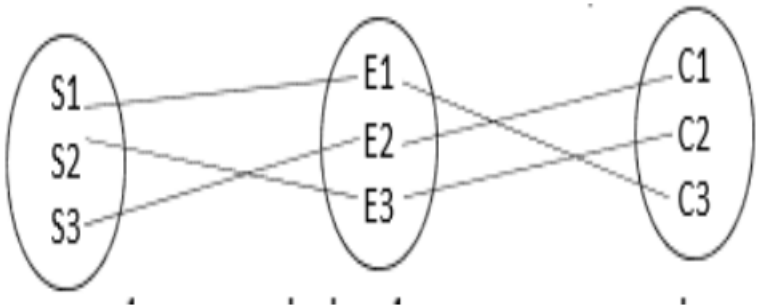




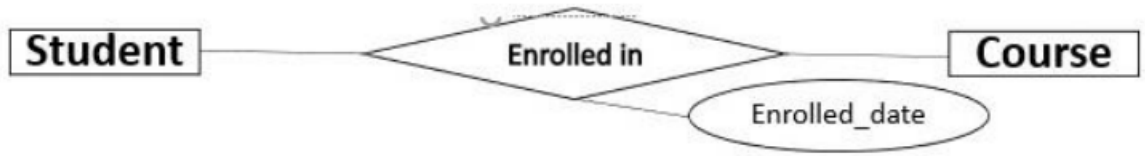
Relationship Type and Relationship Set

- A relationship type represents the association between entity types. For example, "Enrolled in" is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.

- A set of relationships of same type is known as relationship set. The following relationship set depicts / represents / illustrates S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled C3.



In another way, we can say that association between two entity sets is called relationship set. A relationship set may also have attributes called descriptive attributes. For example, the enrolled_in relationship set between entity sets student and course may have the attribute enrolled_date.

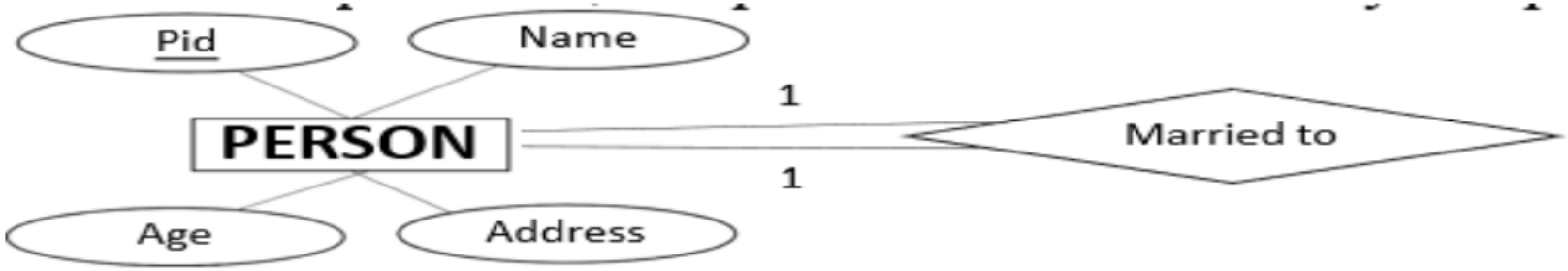


3.3.5 Conceptual Data Modeling And The E-R Model: Degree of a Relationship

- 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-ray Relationship
- **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

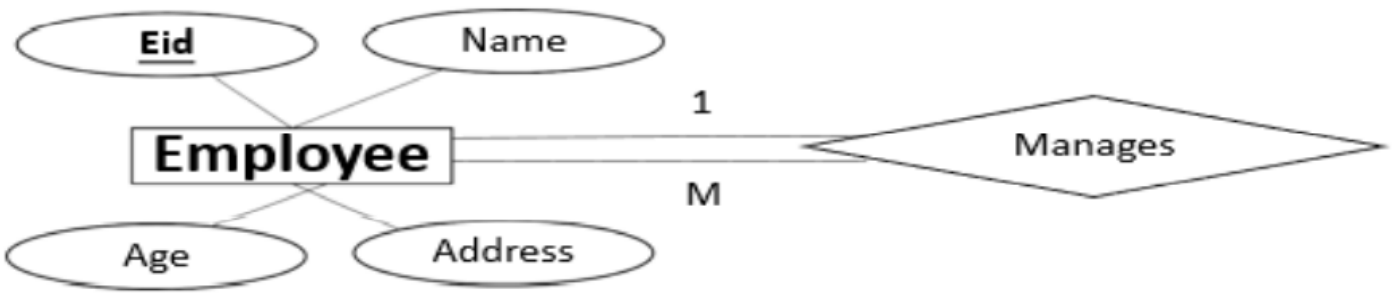
One to One (1:1) Unary Relationship

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



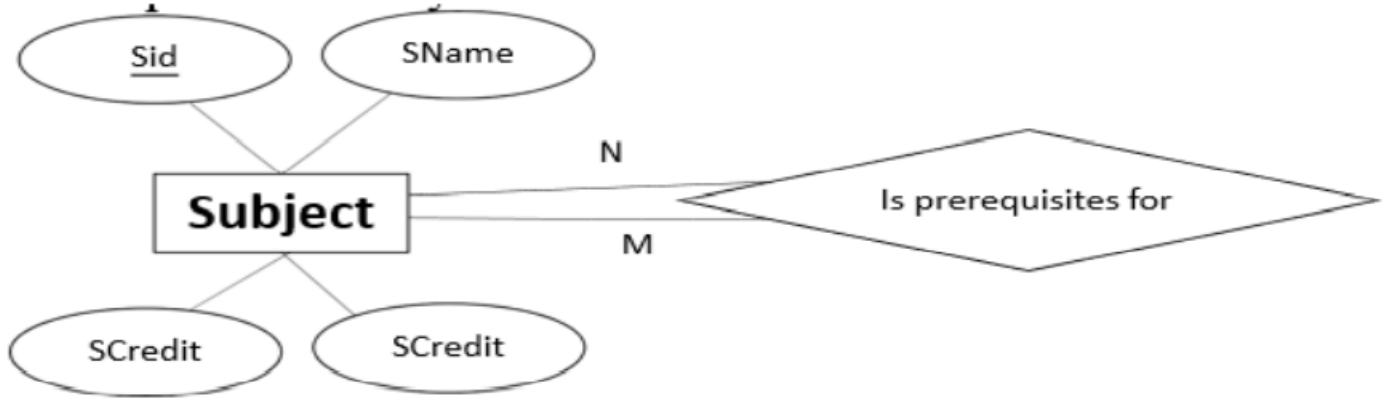
One to Many (1:M) Unary Relationship

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



Many to Many (M:M) Unary Relationship

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

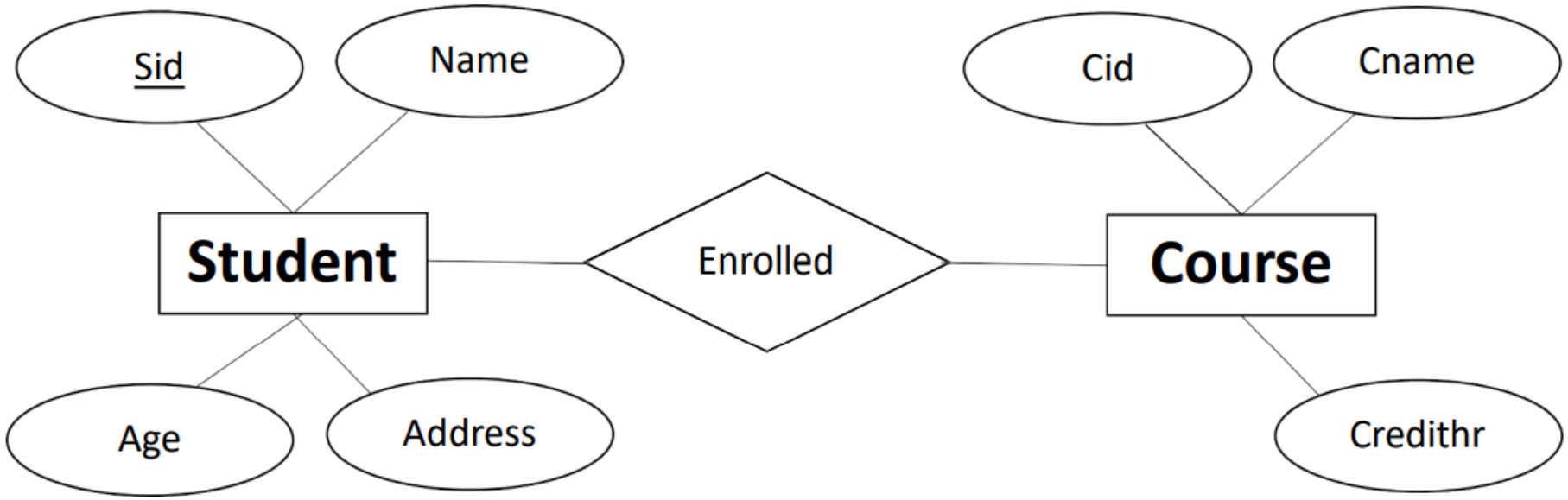


Binary Relationship

- 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.

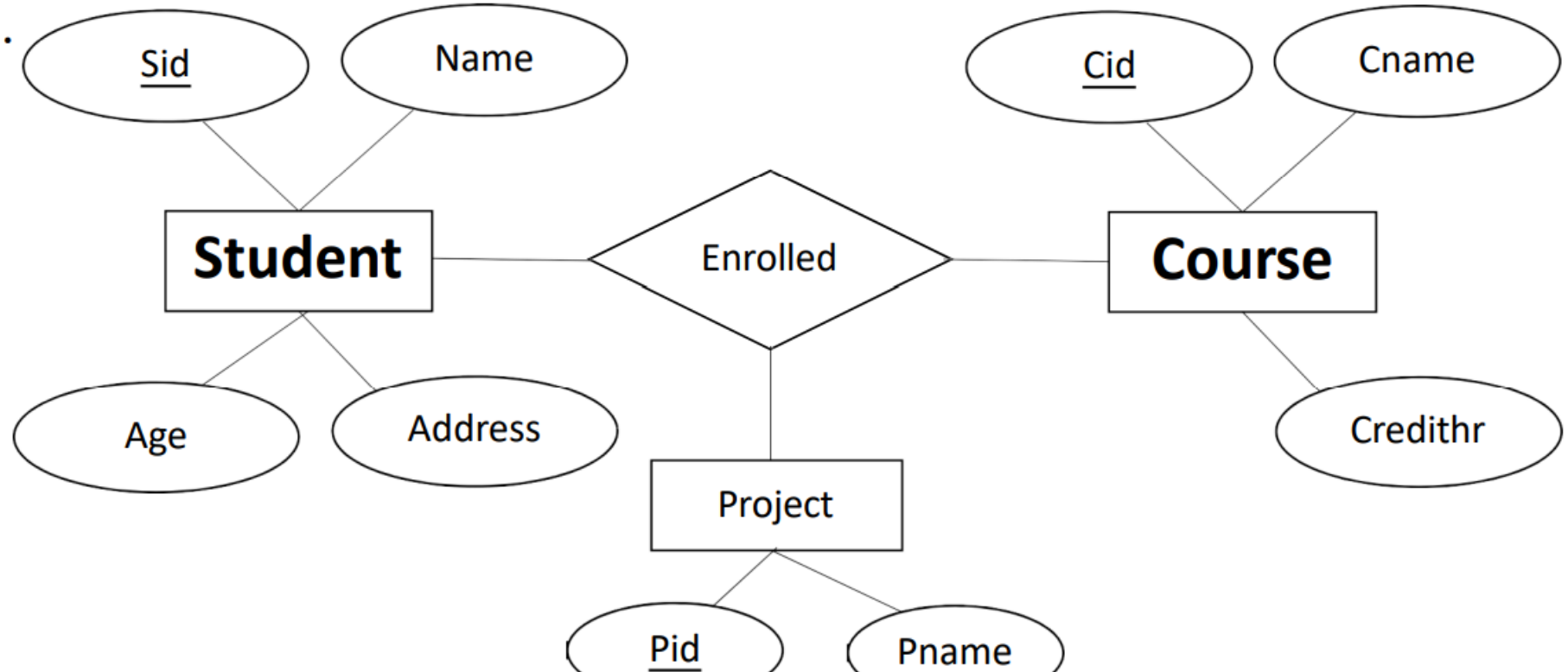
Binary Relationship

- 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.



N-ary relationship

- When there are n entities set participating in a relation, the relationship is called n-ary relationship. For example, 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 then two entities participating with a single relationship i.e. n>2.

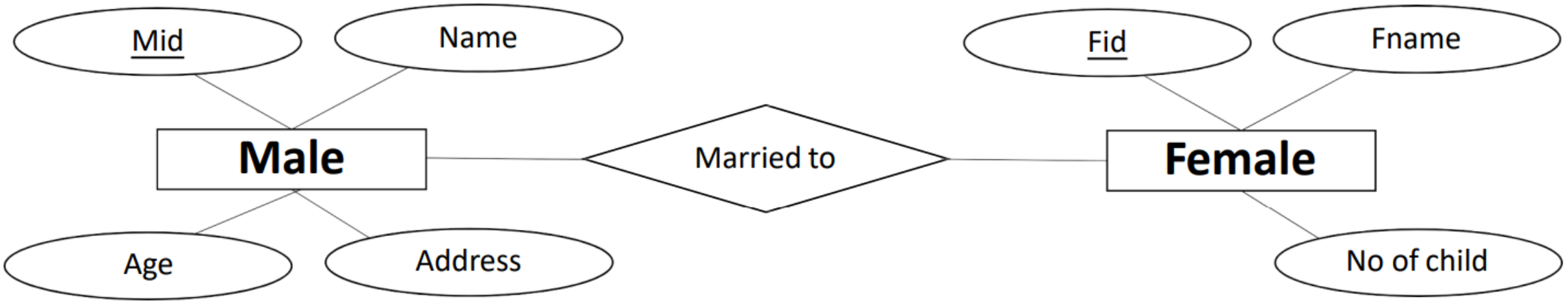


Constraints on ER Model/Structural Constraints in ER

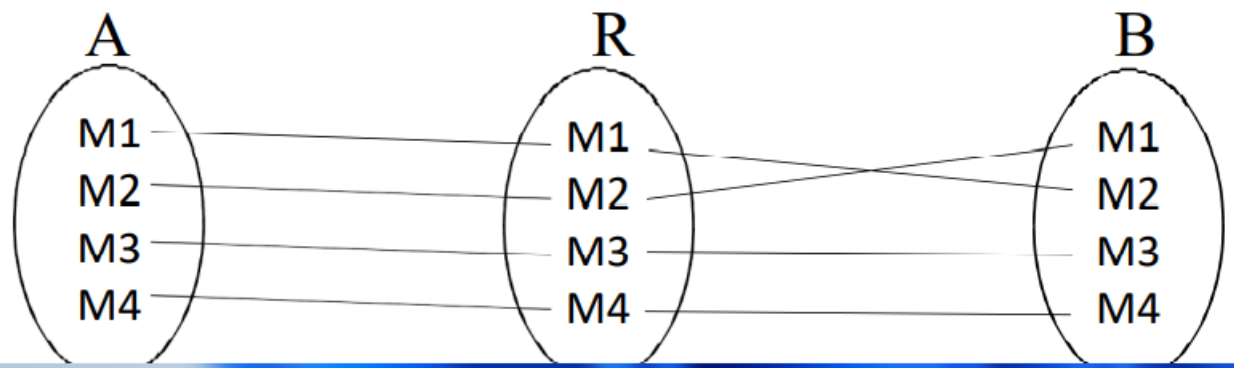
- Relationship sets in ER model usually have certain constraints that limit the possible combinations of entities that may involve in the corresponding relationship set. Database content must confirm these constraints. The most important structural constraints in ER are listed below:
- Mapping cardinalities and
- Participation constraints
- **Mapping Cardinality Constraints**
- The number of times an entity of an entity set participated in a relationship set is known as cardinality. Cardinality can be of different types:
- One-to-One
- One-to-Many
- Many-to-One
- Many-to-Many

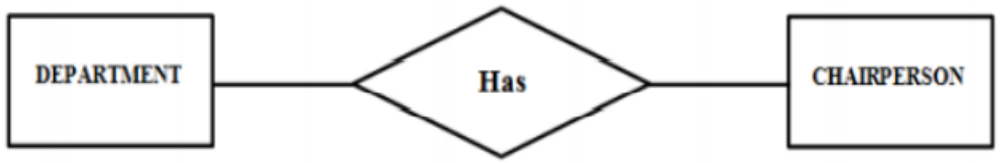
One-to-One Mapping Cardinality Constraints

- In One-to-One mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.

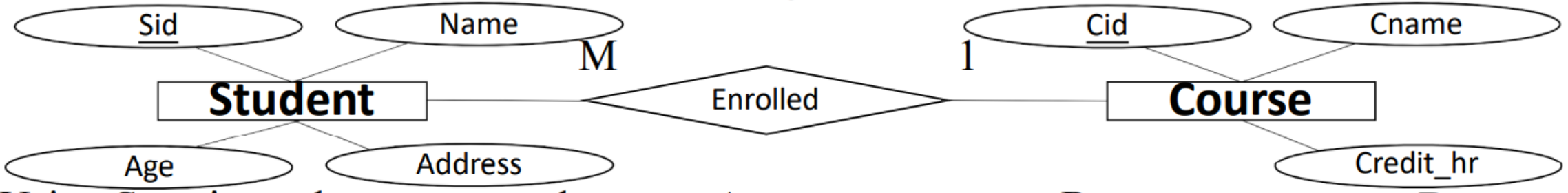


- Using Sets, it can be represented as:

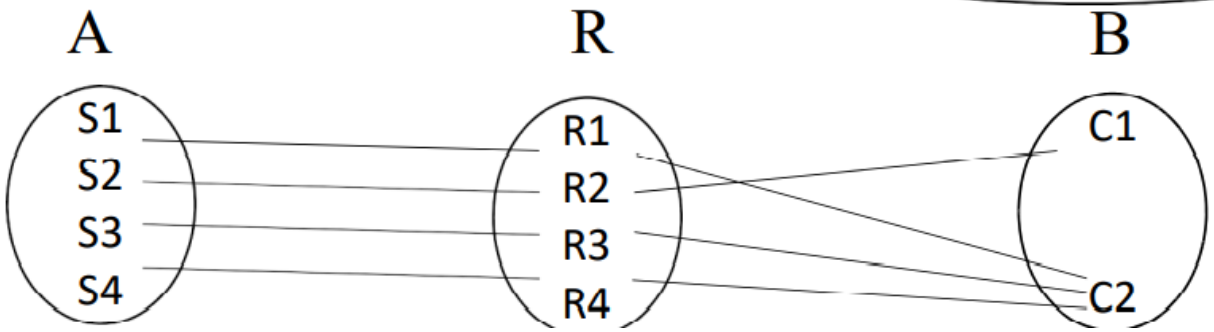




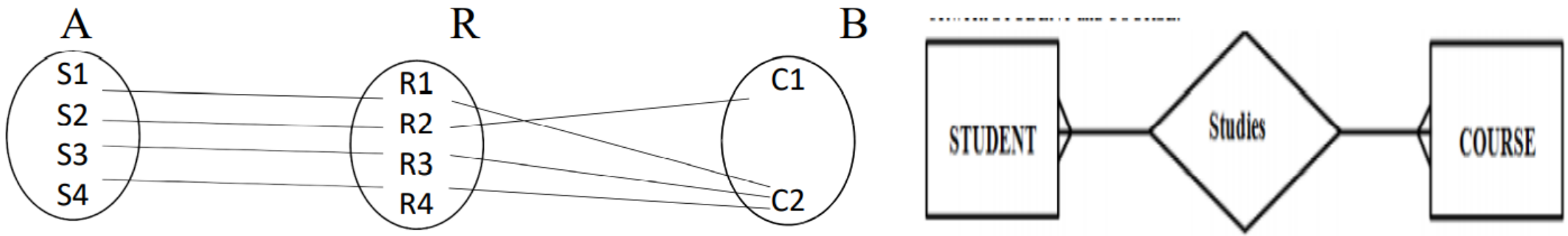
One-to-Many or Many-to-One Mapping Constraints

- In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.
- In this mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.
- Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.
- 

- Using Sets, it can be represented as:



Many-to-Many Mapping Constraints

- In Many-to-Many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1. Let us assume that a student can take more than one course and one course can be taken by many students. So, the relationship will be many to many.
- Using Sets, it can be represented as:
- 

- In this example, student S1 is enrolled in C1 and C3 and Course C2 is enrolled by S1, S2 and S4. So, it is many to many relationships.

Participation Constraints

- Participation Constraint is applied on the entity participating in the relationship set. Constraint on ER model that determines whether all or only some entity occurrences participate in a relationship is called participation constraint.
- It specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- There are two types of participation constraints:
- Total Participation Constraints and
- Partial Participation Constraints.
- **Total Participation Constraints**
- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set. That is why; it is also called as mandatory participation. Total participation is represented using a double line between the entity set and relationship set in ER diagram. If each student must enroll in a course, the participation of

Participation Constraints

- **Partial Participation Constraints**
- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set. That is why; it is also called as optional participation. Partial participation is represented using a single line between the entity set and relationship set in ER diagram. If some courses, are not enrolled by any of the student, the participation of course will be partial.
- The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.
-