Systems Planning and Initial Investigation

- The primary purpose of systems planning is to identify problem's nature and its scope.
- It also includes preliminary (or initial) investigation and feasibility study.
- Initial investigation is used to understand the problem, define the project scope and constraints, identify the benefits, estimate the time and costs, and interact with managers and users. Feasibility study is used to determine some feasibility (economic, operational, technical feasibility etc) of the system.
- The purpose of this phase is twofold.
- First, it answers the question, "Is this project worth looking at?"
- Second, assuming that the problem is worth looking at, it establishes the size and boundaries of the project, the project vision, any constraints or limitations, the required project participants, and the budget and schedule.

Initial Investigation

→ This is the first phase of SDLC and is known as identification of need.
→ This is a user's request to change, improve or enhance an existing system.
→ The objective is to determine whether the request is valid or feasible
→ The user request identifies the need for change and authorizes the initial investigation.

System Planning

- Once a system is found to be feasible, software project managers undertake system planning.
- System planning is undertaken and completed even before any development activity starts.
- System planning consists of the following essential activities:
  • Estimating the following attributes of the system:
  System size: What will be problem complexity in terms of the effort and time required to develop the system?
  Cost: How much is it going to cost to develop the system? **Duration**: How long is it going to take to complete development? **Effort**: How much effort would be required?
- The effectiveness of the subsequent planning activities is based on the accuracy of these estimations.

→ Scheduling manpower and other resources
→ Staff organization and staffing plans
→ Risk identification, analysis, and abatement planning
→ Miscellaneous plans such as quality assurance plan

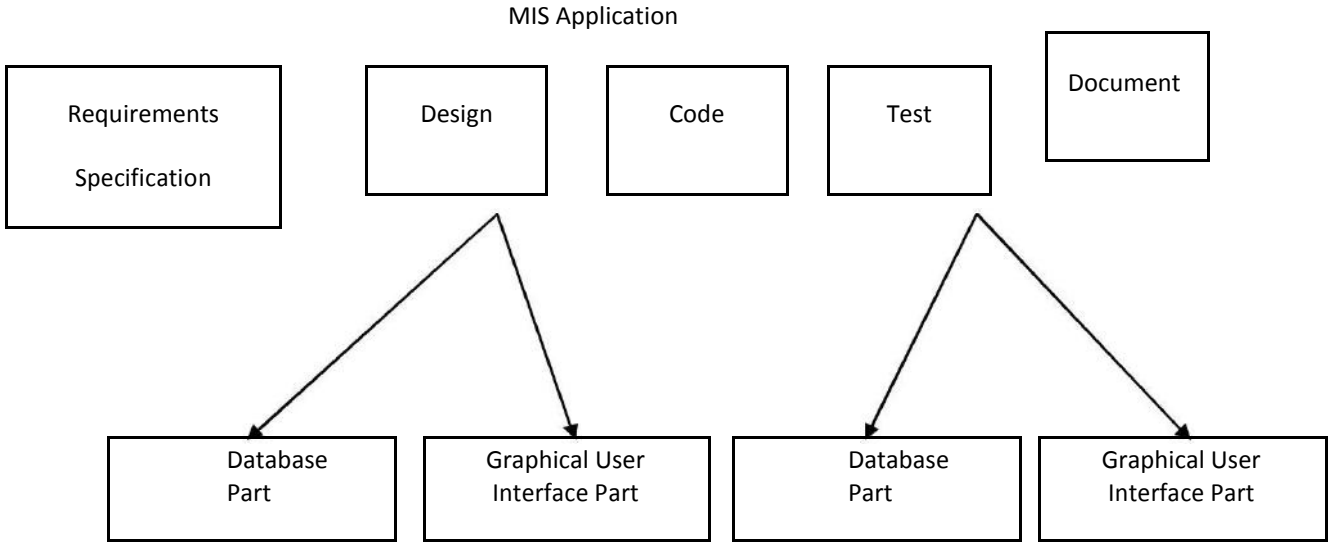
Project scheduling

- Project-task scheduling is an important project planning activity. It involves deciding which tasks would be taken up when. In order to schedule the project activities, a software project manager needs to do the following:
  - Identify all the tasks needed to complete the project.
  - Break down large tasks into small activities.
  - Determine the dependency among different activities.
  - Establish the most likely estimates for the time durations necessary to complete the activities.
  - Allocate resources to activities.
  - Plan the starting and ending dates for various activities.
  - Determine the critical path. A critical path is the chain of activities that determines the duration of the project.
- The first step in scheduling a software project involves identifying all the tasks necessary to complete the project.
- A good knowledge of the intricacies of the project and the development process helps the managers to effectively identify the important tasks of the project.
- Next, the large tasks are broken down into a logical set of small activities which would be assigned to different engineers.
- The work breakdown structure formalism helps the manager to breakdown the tasks systematically.

- After the project manager has broken down the tasks and created the work breakdown structure, he has to find the dependency among the activities.
- Dependency among the different activities determines the order in which the different activities would be carried out.
- If an activity A requires the results of another activity B, then activity A must be scheduled after activity B.
- In general, the task dependencies define a partial ordering among tasks, i.e. each tasks may precede a subset of other tasks, but some tasks might not have any precedence ordering defined between them (called concurrent task).
- The dependencies among the activities are represented in the form of an activity network.
- Once the activity network representation has been worked out, resources are allocated to each activity.
- Resource allocation is typically done using a Gantt chart. After resource allocation is done, a PERT chart representation is developed.
- The PERT chart representation is suitable for program monitoring and control. For task scheduling, the project manager needs to decompose the project tasks into a set of activities.
- The time frame when each activity is to be performed is to be determined. The end of each activity is called milestone.
- The project manager tracks the progress of a project by monitoring the timely completion of the milestones.
- If he observes that the milestones start getting delayed, then he has to carefully control the activities, so that the overall deadline can still be met.

Work breakdown structure

- Work Breakdown Structure (WBS) is used to decompose a given task set recursively into small activities.
- WBS provides a notation for representing the major tasks need to be carried out in order to solve a problem.
- The root of the tree is labeled by the problem name.
- Each node of the tree is broken down into smaller activities that are made the children of the node.
- Each activity is recursively decomposed into smaller sub-activities until at the leaf level, the activities requires approximately two weeks to develop.
- Fig. represents the WBS of an MIS (Management Information System) software.
- While breaking down a task into smaller tasks, the manager has to make some hard decisions.
- If a task is broken down into large number of very small activities, these can be carried out independently.
- Thus, it becomes possible to develop the product faster (with the help of additional manpower).
- Therefore, to be able to complete a project in the least amount of time, the manager needs to break large tasks into smaller ones, expecting to find more parallelism.
- However, it is not useful to subdivide tasks into units which take less than a week or two to execute.
- Very fine subdivision means that a disproportionate amount of time must be spent on preparing and revising various charts.



Activity Networks

- WBS representation of a project is transformed into an activity network by representing activities identified in WBS along with their interdependencies.
- An activity network shows the different activities making up a project, their estimated durations, and interdependencies (as shown in fig.).
- Each activity is represented by a rectangular node and the duration of the activity is shown along side each task.

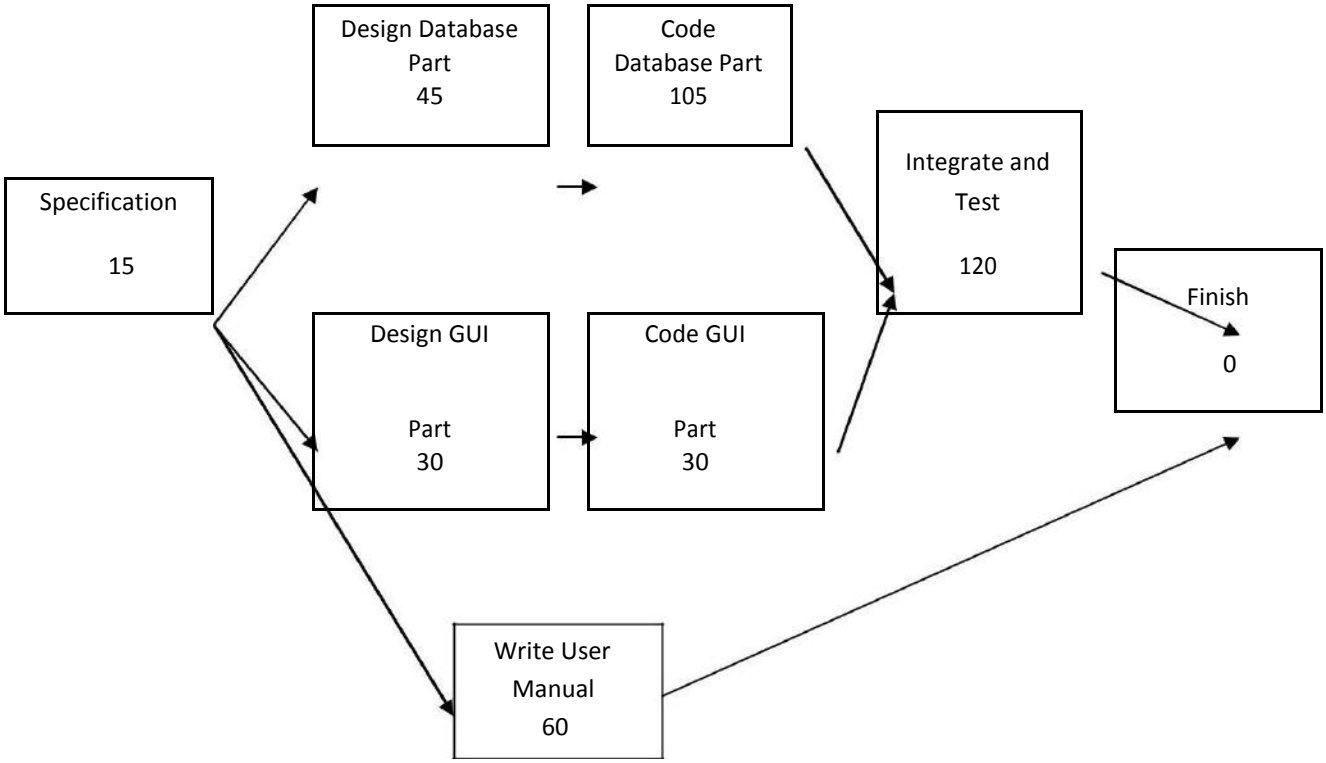


Fig. Activity network representation of the MIS problem

- Managers can estimate the time durations for the different tasks in several ways.
- One possibility is that they can empirically assign durations to different tasks.
- This however is not a good idea, because software engineers often resent such unilateral decisions.
- A possible alternative is to let engineer himself estimate the time for an activity he can assigned to.
- However, some managers prefer to estimate the time for various activities themselves.
- Many managers believe that an aggressive schedule motivates the engineers to do a better and faster job.
- However, careful experiments have shown that unrealistically aggressive schedules not only cause engineers to compromise on intangible quality aspects, but also are a cause for schedule delays.
- A good way to achieve accurately in estimation of the task durations without creating undue schedule pressures is to have people set their own schedules.

Critical Path Method (CPM)

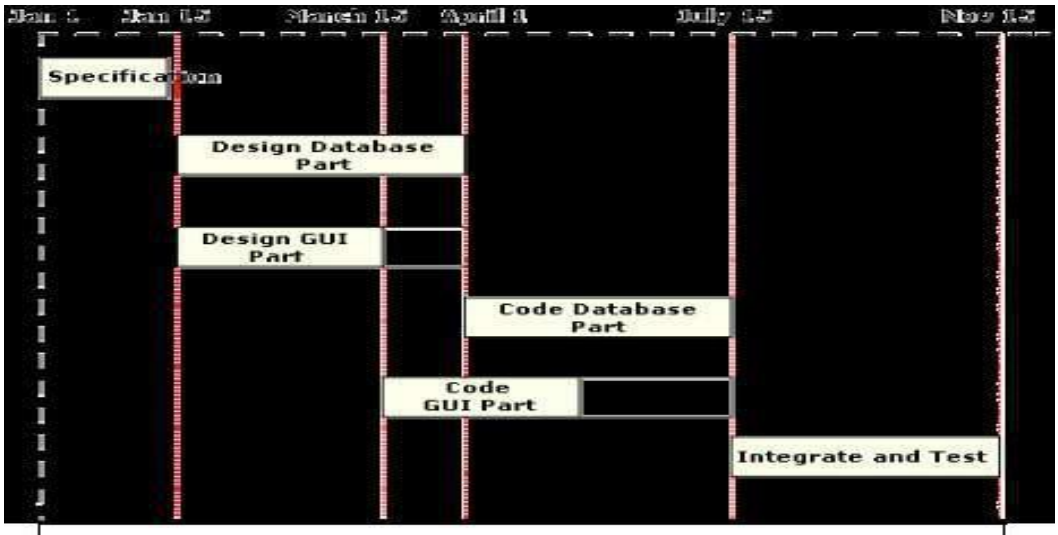
- From the activity network representation following analysis can be made.
- The minimum time (MT) to complete the project is the maximum of all paths from start to finish.
- The earliest start (ES) time of a task is the maximum of all paths from the start to the task.
- The latest start time is the difference between MT and the maximum of all paths from this task to the finish.
- The earliest finish time (EF) of a task is the sum of the earliest start time of the task and the duration of the task.

- The latest finish (LF) time of a task can be obtained by subtracting maximum of all paths from this task to finish from MT. The slack time (ST) is LS – EF and equivalently can be written as LF – EF.
- The slack time (or float time) is the total time that a task may be delayed before it will affect the end time of the project.
- The slack time indicates the "flexibility" in starting and completion of tasks.
- A critical task is one with a zero slack time.
- A path from the start node to the finish node containing only critical tasks is called a critical path.
- These parameters for different tasks for the MIS problem are shown in the following table.

Task	ES	EF	LS	LF	ST
Specification	0	15	0	15	0
Design Database	15	60	15	60	0
Design GUI Part	15	45	90	120	75
Code Database	60	165	60	165	0
Code GUI Part	45	90	120	165	75
Integrate and Test	165	285	165	285	0
Write User Manual	15	75	225	285	210

Gantt chart

- Gantt charts are mainly used to allocate resources to activities.
- The resources allocated to activities include staff, hardware, and software. Gantt charts (named after its developer Henry Gantt) are useful for resource planning.
- A Gantt chart is a special type of bar chart where each bar represents an activity.
- The bars are drawn along a time line.
- The length of each bar is proportional to the duration of time planned for the corresponding activity.
- Gantt charts are used in software project management are actually an enhanced version of the standard Gantt charts.
- In the Gantt charts used for software project management, each bar consists of a white part and a shaded part.
- The shaded part of the bar shows the length of time each task is estimated to take. The white part shows the slack time, that is, the latest time by which a task must be finished.
- A Gantt chart representation for the MIS problem is shown in the fig.



Requirement Analysis:

- Requirements analysis is the first stage in the systems engineering process and software development process.
- Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered

product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users.

- Requirements analysis is critical to the success of a development project.
- Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.
- Requirements can be functional and non-functional.

Overview

Conceptually, requirements analysis includes three types of activity:

- Eliciting requirements: the task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering.
- Analyzing requirements: determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then resolving these issues.
- Recording requirements: Requirements might be documented in various forms, such as natural-language documents, use cases, user stories, or process specifications.

- Requirements analysis can be a long and arduous process during which many delicate psychological skills are involved.
- New systems change the environment and relationships between people, so it is important to identify all the stakeholders, take into account all their needs and ensure they understand the implications of the new systems.
- Analysts can employ several techniques to elicit the requirements from the customer. Historically, this has included such things as holding interviews, or holding focus groups (more aptly named in this context as requirements workshops) and creating requirements lists.
- More modern techniques Software Development Process – activities and steps include prototyping, and use cases. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced.

Types of Requirements

Requirements are categorized in several ways. The following are common categorizations of requirements that relate to technical management:

Customer Requirements

- Statements of fact and assumptions that define the expectations of the system in terms of mission objectives, environment, constraints, and measures of effectiveness and suitability (MOE/MOS).
- The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer.
- Operational requirements will define the basic need and, at a minimum, answer the questions posed in the following listing:
- Operational distribution or deployment: Where will the system be used?
- Mission profile or scenario: How will the system accomplish its mission objective?
- Performance and related parameters: What are the critical system parameters to accomplish the mission?
- Utilization environments: How are the various system components to be used?
- Effectiveness requirements: How effective or efficient must the system be in performing its mission? Software Development Process – activities and steps
- Operational life cycle: How long will the system be in use by the user?
- Environment: What environments will the system be expected to operate in an effective manner?

Functional Requirements

- Functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished.
- Functional requirements analysis will be used as the top level functions for functional analysis.

Non-functional Requirements

- Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors.

Performance Requirements

- The extent to which a mission or function must be executed; generally measured in terms of quantity, quality, coverage, timeliness or readiness.
- During requirements analysis, performance (how well does it have to be done) requirements will be interactively developed across all identified functions based on system life cycle factors; and

characterized in terms of the degree of certainty in their estimate, the degree of criticality to system success, and their relationship to other requirements.

Design Requirements

- The ―build to,‖ ―code to,‖ and ―buy to‖ requirements for products and ―how to execute‖ requirements for processes expressed in technical data packages and technical manuals.

Derived Requirements

- Requirements that are implied or transformed from higher-level requirement. For example, a requirement for long range or high speed may result in a design requirement for low weight.

Allocated Requirements

- A requirement that is established by dividing or otherwise allocating a high-level requirement into multiple lower-level requirements.
- Example: A 100-pound item that consists of two subsystems might result in weight requirements of 70 pounds and 30 pounds for the two lower-level items.

Information Gathering Techniques

- To develop an information system, we first must be able to correctly identify, analyze, and understand what the users' requirements are or what the user wants the system to do.
- To know users' requirements, we use information gathering techniques. Information gathering techniques are also called **requirements discovery techniques** or **fact finding techniques** or **data collection techniques** .
- Information gathering includes those techniques to be used by system analysts to identify or extract system problems and solution requirements from the user community.
- Systems analysts need an organized method for information gathering.
- Some information gathering techniques are discussed below:

Sampling of Existing Documentation, Forms, and Files

When we are studying an existing system, we can develop a good feel for the system by studying existing documentation, forms, and files. A good analyst always gets facts from the existing documentation rather than from people.

Because it would be impractical to study every occurrence of every form or record in a file or database, system analysts normally use sampling techniques to determine what can happen in the system.

Sampling is the process of collecting a representative sample of documents, forms, and records. The most commonly used sampling techniques are **randomization** and **stratification**.

Randomization is a sampling technique in which there is no predetermined pattern or plan for selecting sample data.

Stratification is a systematic sampling technique that attempts to reduce the variance of the estimates by spreading out the sampling.

Research and Site Visits

In this technique, we perform site visits at systems they know have experienced similar problems. If these systems are willing to share, valuable information can be obtained that
may save tremendous time and cost in the development process.

Computer trade journals and reference books are also a good source of information. They can provide us with information how others have solved similar problems. Also, through the Internet we can collect immeasurable amounts of information.

Observation of the Work Environment

Observation is an effective data-collection technique for obtaining an understanding of a system. In this technique, the systems analyst either participates in or watches a person performing activities to learn about the system.

Advantages:

→ Data gathered can be highly reliable. Sometimes observations can be conducted to check the validity of data obtained directly from individuals.

→ The systems analyst is able to see exactly what is being done.

→ It is relatively inexpensive compared to other fact-finding techniques, because other techniques usually require more employees.

→ It allows the system analyst to do work measurement.

Disadvantages:

→ People usually feel uncomfortable when being watched to their work.

→ Some system activities may take place at odd times, causing a scheduling inconvenience for the system analyst.
→ It may cause interruption.
→ Some tasks may not always be performed by observation.

Questionnaires

This technique is used to conduct surveys through questionnaires. **Questionnaires** are special purpose documents that allow the analyst to collect information and opinions from the respondents. The document can be mass -produced and distributed to respondents, who can then complete the questionnaire on their own time. Questionnaires allow the analyst to collect facts from a large number of people.

Types of Questionnaires

There are two formats of questionnaire, **free-format** and **fixed-format**. Free-format questionnaire offer the respondent to record the answer in the space provided after the questionnaire.

Fixed-format questionnaire contain questions that require selection of predefined responses. In this format, the respondent must choose from the available answer. There are three types of fixed-format questions:

→ **Multiple Choice Questions: -** The respondent is given several answers of a question. The respondent should be told if more than one answer can be selected.
→ **Rating Questions: -** The respondent is given a statement and asked to use supplied responses to state an opinion.
→ **Ranking questions: -** The respondent is given several possible answers, which are to be ranked in the order of preference or experience.

Advantages

→ Most questionnaires can be answered quickly.
→ Inexpensive for gathering data from a large number of individuals.
→ Responses can be tabulated and analyzed quickly.

Disadvantages

→ There is no guarantee that an individual will answer or expand on all the questions.
→ Questionnaires tend to be inflexible.
→ It not possible for the systems analyst to observe and analyze the respondent's body language.
→ There is no immediate opportunity to clarify a vague or incomplete answer to questions.
→ Good questionnaires are difficult to prepare.
→ The number of respondents is often low.

Interviews

The personal **interview** is generally recognized as the most often used fact-finding technique. Interviews are the fact -finding techniques whereby the systems analysts collect information from individuals through face-to-face interaction.

There are two roles assumed in an interview. The systems analyst is the **interviewer**, responsible for organizing and conducting the interview. The system user or system owner is the **interviewee**, who is asked to respond to a series of questions.

Types of Interviews

There are two types of interviews: **unstructured** and **structured** . Unstructured interviews are conducted with only a general goal or subject in mind and with few, if any, specific questions. Structured interviews on the other hand are conducted with a set of specific questions to ask the interviewee.

Types of Questions

There are two types of questions in interview: **open-ended** and **closed-ended**. Open-ended questions allow the interviewee to respond in any way that seems appropriate. But, closed-ended questions restrict answers to either specific choices or short, direct responses.

Advantages

→ Interviews give the analyst an opportunity to motivate the interviewee to respond freely and openly to questions.
→ Interviews allow more feedback from the interviewee.
→ Interviews give the analyst an opportunity to observe the interviewee's nonverbal communication.

Disadvantages

→ Interviewing is a very time-consuming and therefore costly fact-finding approach.
→ Success of interviews is highly dependent on the systems analyst's human relational skills.
→ Interviewing may be impractical due to the location of interviewees.

Discovery Prototyping

Discovery prototyping is the act of building a small-scale, representative or working model of the users' requirements to discover or verify the requirements.

Usually only the areas where the requirements are not clearly understood are prototyped. Creating discovery prototypes enables the developers as well as the users to better understand and refine the requirements involved with developing the system.

Advantages

→ Allows users and developers to experiment with the software and develop an understanding of how the system might work.
→ Aids in determining the feasibility and usefulness of the system before high development costs are incurred.
→ Serves as a training mechanism for users.
→ Aids in building system test plans and scenarios to be used last in the system testing process.
→ May minimize the time spent for fact-finding and help to define more stable and reliable requirements.

Disadvantages

→ Users and developers may need to be trained in the prototyping approach.
→ Doing prototyping may extend the development schedule and increase the development costs
→ Users may develop unrealistic expectations.

Joint Requirements Planning (JRP)

It is a process whereby highly structured group meeting is conducted to analyze problems and define requirements. JRP is a subset of a more comprehensive ***joint application development (JAD)***. The JRP participants are:

→ **Sponsor: -** This person is normally an individual who is in top management and has authority that spans the different departments and users who are to be involved in the systems project.
→ **Facilitator: -** The JRP facilitator or leader is usually responsible for leading all sessions that are held for a systems project.
→ **Users and Managers: -** Users devote themselves to the JRP sessions to effectively communicate business rules and requirements, review design prototypes and make acceptance decisions. Managers approve project objectives, establish project priorities, approve schedules and costs, and approve identified
training needs and implementation plans.
→ **Scribe(s): -** Scribes are responsible for keeping records pertaining to everything discussed in the meeting.
→ **IT Staff: -** IT personnel listen and take notes regarding issues and requirements voiced by the users and managers. Normally, IT personnel do not speak unless invited to do so.

Benefits

→ It actively involves users and managers in the development project.
→ It reduces the amount of time required to develop systems.
→ When JRP incorporates prototyping as a means for conforming requirements and obtaining design approvals, the benefits of prototyping are realized.

Structured Analysis Tools

Structured analysis includes different tools like DFD (data flow diagram), ERD (entity relationship diagram), data dictionary, structured English, decision table, and decision tree.

Feasibility Study

- **Feasibility** is the measure of how beneficial or practical the development of information system will be to an organization.
- **Feasibility study** is the process by which feasibility is measured. Feasibility analysis is appropriate to the systems analysis phase.
- A feasibility study is an analysis of the viability of an idea through a disciplined and documented process of thinking through the idea from its logical beginning to its logical end.
- A feasibility study provides an Investigating function that helps answer "Should we proceed with the proposed project idea? Is it a viable business venture?"

- A feasibility study should be conducted to determine the viability of an idea before proceeding with the development of a business.

Why feasibility study?

Objectives:

- To find out if an system development project can be done:
  - ...is it possible?
  - ...is it justified?
- To suggest possible alternative solutions.
- To provide management with enough information to know:
  - Whether the project can be done
  - Whether the final product will benefit its intended users
  - What the alternatives are (so that a selection can be made in subsequent phases)
  - Whether there is a preferred alternative

Four Tests for Feasibility

During systems analysis phase, the system analyst identifies different alternate solutions and analyzes those solutions for feasibility. To analyze different alternative solutions, most analysts use four categories of feasibility tests:

- Operational feasibility
- Technical feasibility
- Schedule feasibility and
- Economic feasibility

1. **Operational Feasibility:** It is a measure of how well the solution will work in an organization. It is also a measure of how people feel about the system/project. So, this feasibility is people oriented. Operational feasibility addresses two major issues:
   a. *Is the problem worth solving, or will the solution to the problem work?*
   b. *How do end users and management feel about the problem (solution)?*

   When determining operational feasibility, *usability analysis* is often performed with a working prototype of the proposed system. **Usability analysis** is a test of system's user interfaces and is measured in how easy they are to learn and to use and how they support the desired productivity levels of the users. Usability is measured in terms of *ease of learning, ease of use*, and *satisfaction*.
2. **Technical Feasibility:** It is a measure of practically of a specific technical solution and availability of technical resources and expertise. Technical feasibility is computer oriented. This feasibility addresses three major issues:
   a. *Is the proposed technology or solution practical?*
   b. *Do we currently possess the necessary technology?*
   c. *Do we possess the necessary technical expertise, and is the schedule reasonable?*
3. **Schedule Feasibility:** It is a measure of how reasonable the project timetable is. Schedule feasibility is the determination of whether the time allocated for a project seems accurate. Projects are initiated with specific deadlines. It is necessary to determine whether the deadlines are mandatory or desirable. If the deadlines are desirable rather than mandatory, the analyst can propose alternative schedules.
4. **Economic Feasibility:** It is the measure of the cost-effectiveness of a project or solution. This feasibility deals with costs and benefits of the information system. The bottom-line in many projects is economic feasibility. During the early phases of the project, economic feasibility analysis amounts to little more than judging whether the possible benefits of solving the problem are worthwhile. However, as soon as specific requirements and alternative solutions have been identified, the analyst can determine the costs and benefits of each alternative.

Some other feasibility tests are also possible. These are *legal and contractual feasibility* and *political feasibility*. **Legal and contractual feasibility** is the process of assessing potential legal and contractual ramifications due to the construction of a system. **Political feasibility** is the process of evaluating how key stakeholders within the organization view the proposed system.

Cost-Benefit Analysis Techniques

Economic feasibility has been defined as a cost-benefit analysis. Most schools offer coerces like *financial management, financial decision analysis*, and *engineering economics and analysis* for cost benefit analysis. The cost benefit analysis techniques include:

- *How much will the system costs?*
- *What benefits will the system provide?*
- *Is the proposed system cost-effective?*

How much will the system costs?

Costs fall into two categories: *costs associated with developing the system* and *costs associated with operating the system*. The former costs can be estimated from the start of the project and should be refined at the end of each phase of the project. The later can be estimated only after specific computer-based solution has been defined.

System development costs are usually onetime costs that will not recur after the project has been completed. Many organizations have standard cost categories that must be evaluated. In the absence of such categories, we use the following list:

- **Personnel cost –** The salaries of systems analysts, programmers, consultants, data entry personnel, computer operators, secretaries, and the like who work on the project.
- **Computer usage –** The cost in the use of computer resources.
- **Training –** Expenses for the training of computer personnel or end-users.
- **Supply, duplication, and equipment costs**.
- **Cost of any new computer equipment and software**.

The operating costs tend to recur throughout the lifetime of the system. The costs in this case can be classified as *fixed* or *variable*.

- **Fixed costs –** Fixed costs occur at regular intervals but at relatively fixed rates. Some examples include: lease payments and software license payments, salaries of IS operators and support personnel etc.
- **Variable costs –** Variable costs occur in proportion to some usage factor. Some examples include: costs of computer usage (e.g., CPU time used, storage used), supplies (e.g., printer paper, floppy disks), overhead costs (e.g., utilities, maintenance, and telephone service) etc.

What benefits will the system provide?

Benefits normally increase profits or decrease costs. Benefits can be classified into two categories: *tangible* and *intangible*.

- **Tangible benefits –** Tangible benefits are those that can be easily quantified. These benefits are usually measured in terms of monthly or annual savings or of profit to the firm. Alternatively, these benefits might be measured in terms of unit cost savings or profit. Some examples of tangible benefits are: fewer processing errors, increased throughput, decreased response time, elimination of job steps, increased sales, reduced credit losses, and reduce expenses.
- **Intangible benefits –** Intangible benefits are believed to be difficult or impossible to quantify. Unless these benefits are at least identified, it is entirely possible that many projects would not be feasible. Some examples of intangible benefits are: improved customer goodwill, improved employee morale, better service to community, and better decision making.

Unfortunately, if a benefit cannot be quantified, it is difficult to accept the validity of an associated cost-benefit analysis that is based on incomplete data.

Is the proposed system cost-effective?

There are three popular techniques to assess economic feasibility, also called cost-effectiveness: *payback analysis*, *return to investment*, and *net present value*.

One concept that is shared by all three techniques is the **time value of money** – a dollar today is worth more than a dollar one-year from now.

Some of the costs of the system will be accrued in after implementation. Before cost-benefit analysis, these costs should be brought back to the current dollars. **Present value** is the current value of a dollar at any time in the future. It is calculated using the formula:

$$PVn = 1/(1 + i)^n$$

Where PVn is the present value of $1.00 n years from now and i is the discount rate.

- **Payback analysis –** It is a technique for determining if and when an investment will pay for itself. Because system development costs are incurred long before benefits begin to occur, it will take

some time for the benefits to overtake the costs. After implementation, there will be additional operating expenses that must be recovered. *Payback analysis* determines how much time will lapse before accrued benefits overtake accrued and continuing costs. This period of time is called **payback period**, that is, the period of time that will lapse before accrued benefits overtake accrued costs.

- **Return-on-investment analysis –** This technique compares the lifetime profitability of the solution. It is a percentage rate that measures the relationship between the amounts the business gets back from an investment and the amount invested. It is calculated as follows:

  Lifetime ROI = (Estimated lifetime benefits – Estimated lifetime costs)/Estimated lifetime costs

- **Net present value –** It is an analysis technique that compares costs and benefits for each year of the system's lifetime. Many managers consider it the preferred cost-benefit analysis technique.