UNIT-I: SYSTEM STUDY AND SYSTEM DEVELOPMENT LIFE CYCLE

CONCEPT OF SYSTEM

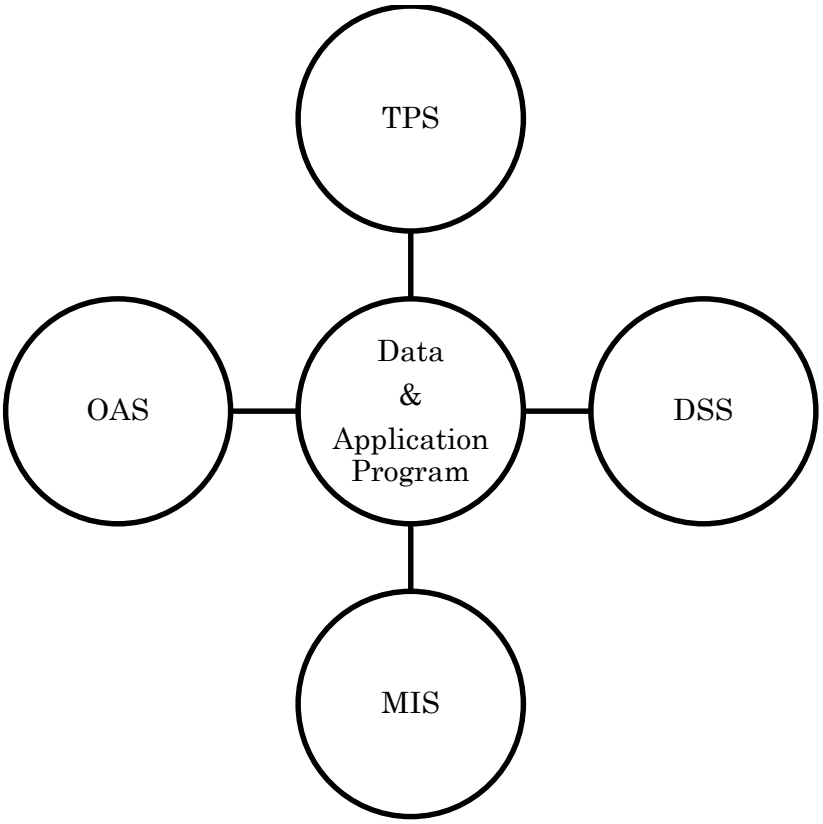
System : System is an organized relationship among the functioning units or components to achieve predefined goal. System consisted of man, machine and method.

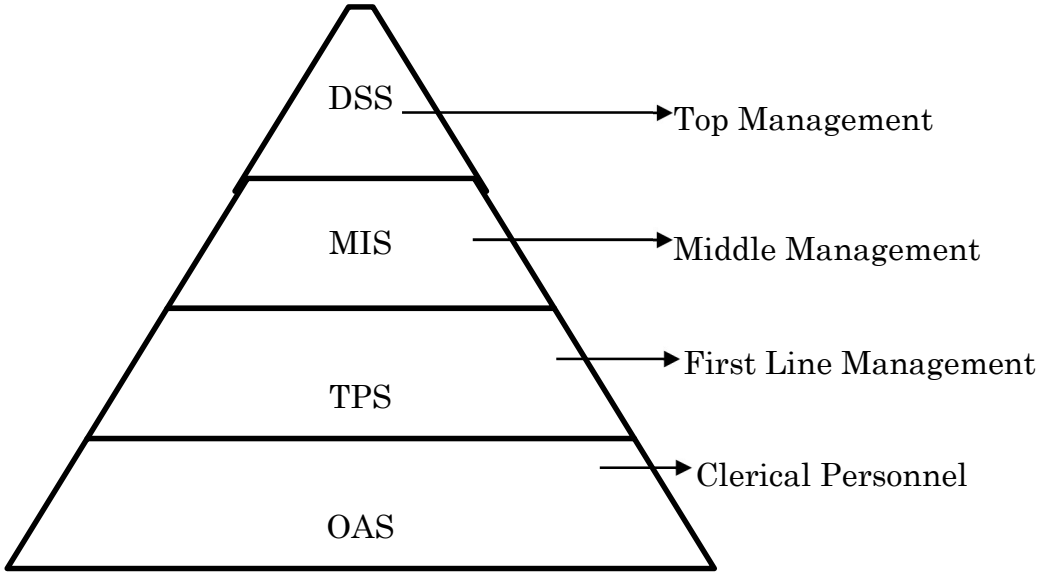
A car is an example of a system. Car is a machine. To drive it we must follow the method. It is driven by man. It is made up of different components. A computer is hardware (machine), without any software (method) we can't operate it. If we want to get benefits of computer we must required user (man). A computer is made up of different components organized in a specific structure and order. All Humans are made up of different sub system like muscular system, skeleton system, nervous system etc. it's functioning units are parts of our body. In the context of the subject software is our system.

Life blood of a system is data. System can be differentiated by manual or automated. Manual system has been introduced from many centuries. Trading between different countries, banking were already coming from very long time. Functioning of all these is possible only by implementation of a particular system. After computer has been invented all these converted to Computer Based Information System. They are classified as follows:

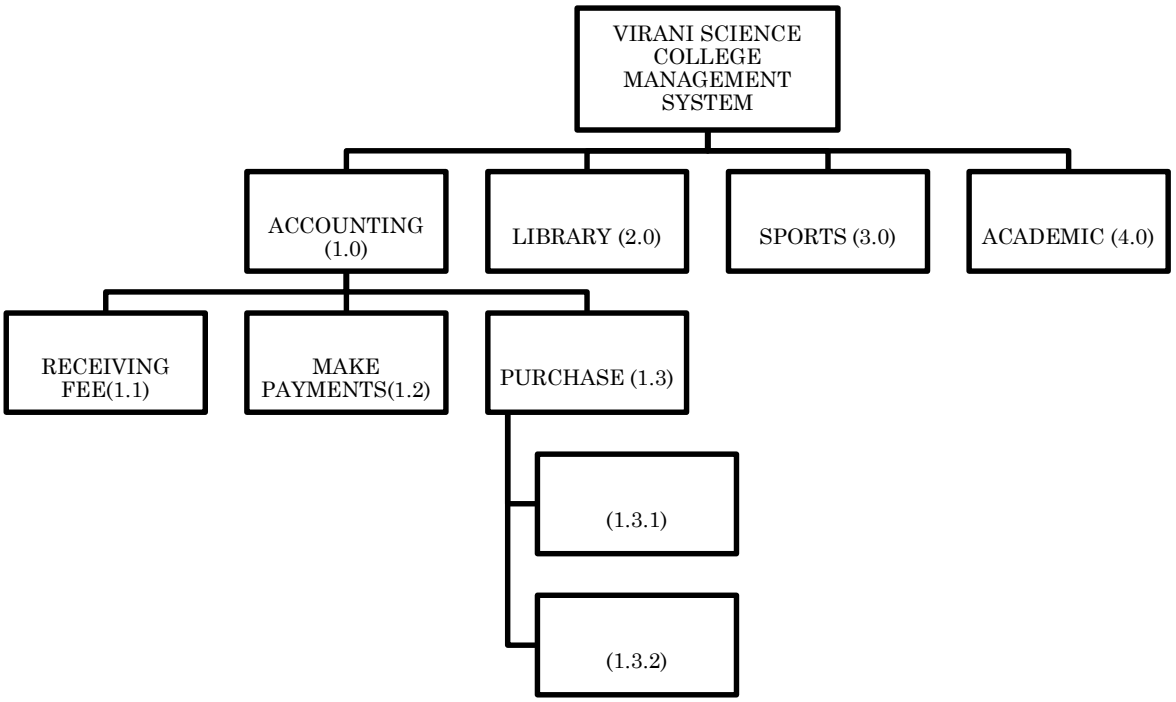
- Transaction Processing Systems (TPS)
- Management Information Systems (MIS)
- Decision Support Systems (DSS)
- Office Automation Systems (OAS).

Category of Information System	Characteristics
Decision Support System	Gives information to managers who take judgments about particular situations. Supports decision makers in situations that are unstructured.
Management Information System	Gives input to be recommended in the managerial decision process. Work with supporting well structured decision situations. Typical information requirements can be projected
Transaction Processing System	Computer-based processing for manual procedures. Includes well structured routine processes. Includes record-keeping applications.
Office Automation System	It includes devices allows to perform many office activities with electronic mode. It is a multi-function, integrated computer based system.





Subsystem : Components of a system is known as subsystem. Actually Subsystem is made up of functional units. A college may be considered as a system and its sub system is Accounting Department. Accounting department is made up of functioning units like receiving fees, payments, purchase dept. etc. This can be shown by hierarchical chart (HIPO chart). The leaf nodes of this chart are modules or functioning units which are non – decomposable.



As shown above virani Science College is a system. Its subordinate 1.0, 2.0, 3.0, 4.0 are sub system. 1.1, 1.2 are functioning components are called modules. 1.3 is called component which is further divided into 1.3.1 and 1.3.2 which are modules of 1.3.

Business System: Any business is generally established for Profit Maximization and Cost minimization. Business system may have different sub systems like accounting, marketing, purchasing, production, inventory etc.

Each sub system may have its functional units using that they works. Business is not only production of goods but it may be service providers, Consultants etc. Like any system it talks inputs and gives output to their customers or clients.

BENEFITS OF SOFTWARE

What is the role of computers in the business?

What are the benefits of using software in the Business system?

What are the reasons of project proposal?

Why system projects?

Answer of above questions in terms of benefits of using software has been given as follows.

[1] CAPABILITY :

Business activities are influenced by an organization's ability to process transactions quickly and efficiently. Information systems add capability in three ways :

(1) Improved processing speed : The inherent speed with which computers process data is one reason why organizations seek the development of systems projects.

(2) Increased volume : Provide capacity to process a greater amount of activity, perhaps to take advantage of new business opportunities.

(3) **Faster retrieval of information :** Locating and retrieving information from storage. The ability in conducting complex searches.

[2] CONTROL :

(1) **Greater accuracy and consistency :** Carrying out computing steps, including arithmetic, correctly and consistently.

(2) **Better security :** Safeguarding sensitive and important data in a form that is accessible only to authorized personnel.

[3] COMMUNICATION :

(1) **Enhanced communication :** Speeding the flow of information and messages between remote locations as well as within offices. This includes the transmission of documents within offices.

(2) **Integration of business areas :** Coordinating business activities taking place in separate areas of an organization, through capture and distribution of information.

[4] COST :

(1) **Monitor costs :** Tracking the costs of labor, goods and overhead is essential to determine whether a firm is performing in line with expectations - within budget.

(2) **Reduce costs :** Using computing capability to process data at a lower cost than possible with

Other methods, while maintaining accuracy, and performance levels.

[5] COMPETITIVENESS :

(1) **Lock in customers :** Changing the relationship with and services provided to customers in such a way that they will not think of changing suppliers.

(2) **Lock out competitors :** Reducing the chances of entering the competitors in the same market because of good information systems being used in the organization.

(3) **Improve arrangements with suppliers :** Changing the pricing, service or delivery arrangements, or relationship between suppliers and the organization to benefit the firm.

(4) **New product development :** Introducing new products with characteristics that use or are influenced by information technology.

- **Information System :**

After getting idea about business and role of software in the business it is easy to understand the definition of Information system.

"An information system is a system which usually functions to produce and communicate information (generally data is transformed into valuable information). It works by planning, controlling and monitoring with the help of devices and procedures depending on user's requirement."

1.3 Characteristics of a System

- **Organization :**

Organization means structure and order. It is the arrangement of components to achieve goal. Organization chart can be used to show structure and order.

- **Interaction :**
  Functioning units must interact with each other for specific purposes. Marketing must interact with production department for the information about products.

- **Interdependence :**
  Component depends on each other. For example engine of vehicle depends on the fuel tank.

- **Integration :**
  It describes the relationships between different components and level of bindings.

- **Central Objective :**
  It is pre-defined goal. The goal of vehicle is to reach at its destination with safety.

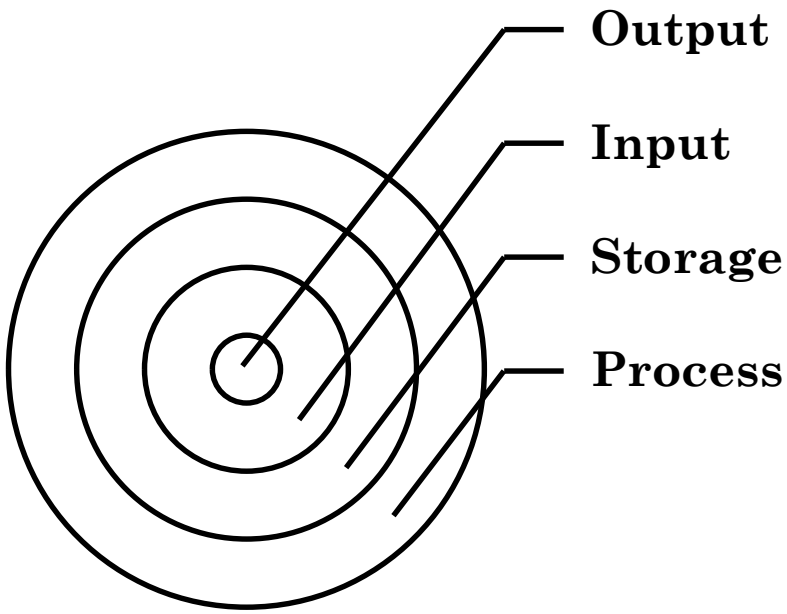
SYSTEM ANALYSIS IS PERFORMED CONSIDERING FOLLOWING ELEMENTS

Outputs : It is the goal of system. Set of User requirements.

Inputs : Input depends on output. It must be accurate. It must be taken from source. It must be provided at the time it requires. Late inputs are useless. Input must be in proper format. Format of input must be fixed in advanced. Input must not be too costly.

Files : Data can be stored in files. Files are nothing but they are tables to store records. Files can stores inputs, intermediate processed data or final results. Files can be used to store historical data.

Processes : To convert input into output we must write code and theses code uses logic. Processes are nothing but programs written in any programming language.



In the Analysis of any system we must start with output or user requirements. User

requirements are co-centered by All system development activities. After identifying requirements what user wants, we can determine what input is necessary to feed in the system. Input is raw data which can be stored in a table of database. Proper database design is necessary at this level. Rest part is writing algorithms which is design part of our program. Logic is developed in the algorithm and later in development phase it is translated in programming language by programmer. Design is known as logical design and writing code is known as physical design.

SYSTEMS ANALYST

A systems analyst performs study; he finds activities and goals and decides procedures to achieve goals.

Like an architect he designs and implements the system. He is takes interest in the profit of the business of client from computer technology. He has unique skills of communication, computer technology, mathematics, statistics, commerce, law etc. He must be flexible and intelligent.

He must have knowledge of people. He coordinates the efforts of different type of persons in an organization to achieve business goals.

A system analyst defines the problem then after he tries to solve it, he talks with user, manager to determine problem and finds the solution from them and decides best solution, he does not takes decision himself but gathers ideas from all people and chooses the best solution or workable solution. System analyst plans to achieve goals. After approval of solution he starts designing and coordinates in testing and helps to decide that new system is working as per requirement or not.

System analyst works with people so he should know each of them, their working and he should motivate them. System Analyst may develop software in any area so he must possess the knowledge of that business. How it functions? Give details. He must take interest in product and profit. System analyst must be aware with limitation and benefits of computers. He must communicate with different level peoples so he must have communication skills. Flexibility of decisions makes more communicative. Conflicting needs and results can be resolved by him. He must be well educated and sharp minded.

- **Role :**
  (1) He defines problem of business. Requirement of a company or any business may be smooth functioning of business activities with profit maximization and cost minimization.

  (2) He uses different techniques to gather information about business which can further used to find strength or weakness of business. He also communicates with different persons in organization for these.

  (3) The systems analyst analyses details and prepare a plan to solve business problem. He finds best way of solving a problem by discussing with other people and refining their ideas and implement practically feasible solution.

  (4) Systems analysts coordinate the process of developing solutions. Since many problems have number of solutions, the systems analyst must evaluate the merit

of such proposes solution before recommending one to the management.

(5) Systems analysts are often referred to as planners. A key part of the systems analyst's job is to develop a plan to meet the management's objectives.

(6) Systems analyst is responsible for designing so that profit of organization can be achieved. Design is a time consuming, complex and precise task.

(7) As per standard set prior, system analyst coordinates in testing of system.

When System Analyst is working on system having bigger size then he can allot his work to his supporters and with co-ordination of them he can be able to design the system. So he allots designing work to System Designer and the work of quality control of coding is allotted to programmer analyst.

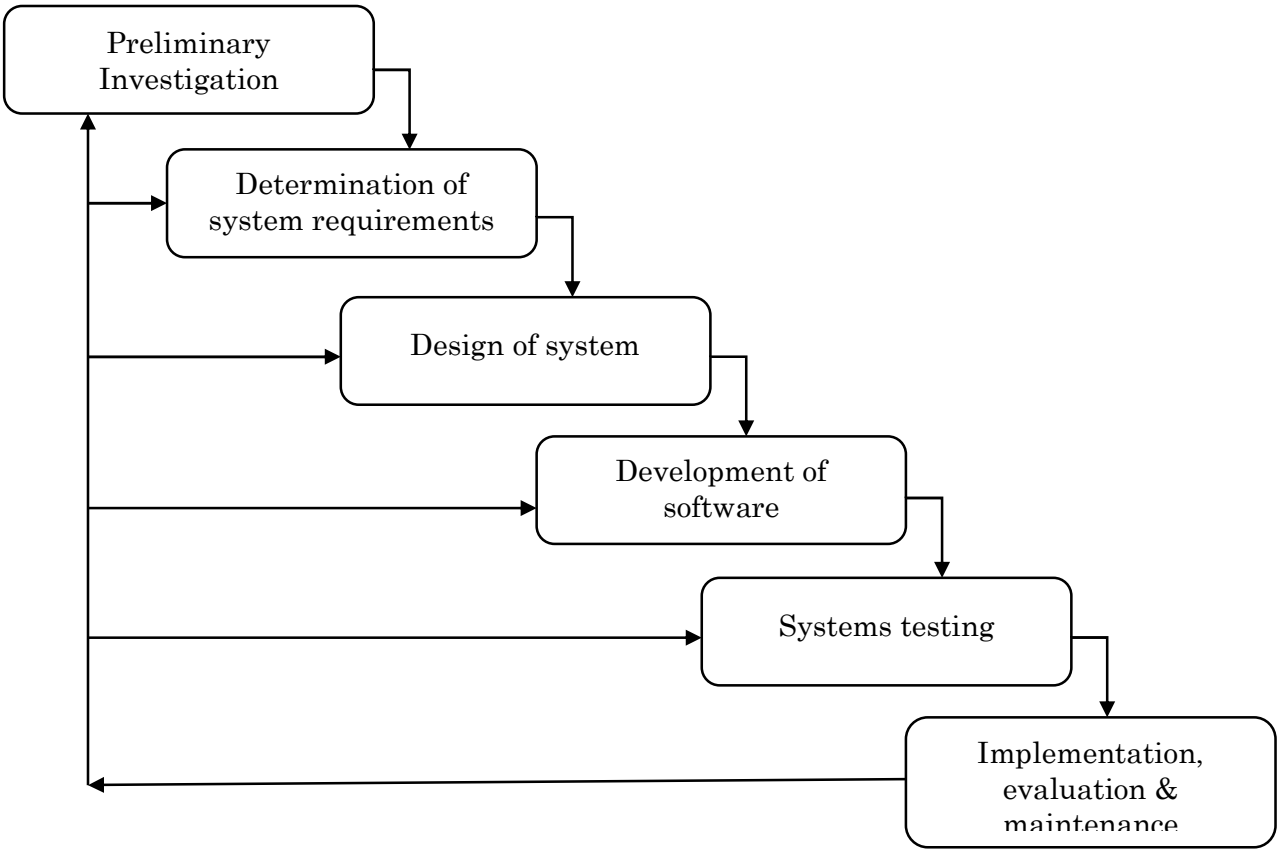
- **Information Analyst :**
  Information Analyst works upon data. He finds the source of data and he decides correct data flow for processing. He also optimizes storage and decides information receiver (external entities). He cares of Input validation. Information analyst calculates cost of each information produced by the system and make analysis for cost of inputs. Information analyst finalizes what output should be produced.

- **Systems Designer :**
  System Designer is responsible to design new system or current manual system. He performs the tasks how the system will generate the information. He consults with the information analyst for the output. System Designer generally designs the system and takes all responsibility about it. Design includes flow chart, algorithm etc.

- **Programmer Analyst :**
  Programmer Analyst performs time and cost ratio in terms of memory resource and processing time. He decides which solution (coding / algorithm) is better. To introduce a single variable may cause degradation of performance. So he always trace all memory resource and processing of each process.

SYSTEM DEVELOPMENT LIFE CYCLE

Here we focus development of software for the progress of existing business. System development indicates Software development to fulfill needs and overcome problem like high operation cost, less profit, management problem, reporting problem, problem of timeliness etc. We call this process as a life cycle because this will continue till the life of business. We develop a new system or refines existing system for improvement.



- **Preliminary Investigation :**
  - **Request Clarification :**
    Demand for software development may arise by people outside or inside from the organization. To avoid ambiguity it is necessary that project request must be clarified and examined in the terms of necessity, benefit, and beneficiary before systems investigation for development.
  - **Feasibility study :**
    Preliminary investigations examine project feasibility; the likelihood the system will be useful to the organization. Three important tests of feasibility are studied and described below.

    (1) Technical feasibility (2) Operational feasibility

    (3) Economic feasibility (4) Social feasibility

    (5) Management feasibility (6) Legal feasibility

    (7) Time feasibility

    (1) Technical feasibility :
    This is concerned with specifying equipment and software that will successfully satisfy the user requirement the technical needs of the system may vary considerably, but might include:

    - The facility to produce outputs in a given time.
    - Response time under certain conditions.

- Ability to process a certain volume of transaction at a particular speed.
- Facility to communicate data to distant location.

In examining technical feasibility, configuration of the system is given more importance than the actual make of hardware. The configuration should give the complete picture about the system's requirements: How many.

Workstations are required, how these units are interconnected so that they could operate and communicate smoothly. What speeds of input and out- put should be achieved at particular quality of printing. This can be used as a basis for the tender document against which dealers and manufacturers can later make their equipment bids. Specific hardware and software products can then be evaluated keeping in view with the logical needs.

At the feasibility stage, it is desirable that two or three different configurations will be pursued that satisfy the key technical requirements but which represent different levels of ambition and cost. Investigation of these technical alternatives can be aided by approaching a range of suppliers for preliminary discussions. Out of all types of feasibility,

Technical feasibility generally is the most difficult to determine.

(2) Operational feasibility :

It is mainly related to human organizational and political aspects. The points to be considered are:

- What changes will be brought with the system?
- What organizational structures are disturbed?
- What new skills will be required? Do the existing staff members have these skills?
- If not, can they be trained in due course of time?

Generally project will not be rejected simply because of operational infeasibility but such considerations are likely to critically affect the nature and scope of the eventual recommendations. This feasibility study is carried out by a small group of people who are familiar with in- formation system techniques, who understand the parts of the business that are relevant to the project and are skilled in system analysis and design process.

(3) Economic feasibility :

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost / benefit analysis; the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the System life cycle.

We must decide our budget and expected profit. If the project is under the

budget and profitable then only we can proceed. This decision must be taken by cost benefit analysis. This analysis can be performed using following steps.

- Calculate the value of the gross benefits of a new system to be developed.
- Calculate operating costs associated with the above mentioned gross benefits.
- Subtract operating cost from the gross benefit this gives net benefit.
- Calculate the approximate value of the development costs.

Establish time-phased relationship for cash flow between net benefits and development costs including payback, inflation etc. This gives relation between net benefits and time.

Since cost plays quite an important role in deciding the new system, it must be identified and estimated properly. Costs vary by type and consist of various distinct elements. Benefits are also of different type and can be grouped on the basis of advantages they provide to the management. The benefits of a project include four types:

- **Cost-savings benefits :** Cost-savings benefits lead to reductions in administrative and operational costs. A reduction in the size of the clerical staff used in the support of an administrative activity is an example of a cost-saying benefit.
- **Cost-avoidance benefits :** Cost-avoidance benefits are those which eliminate future administrative and operational costs. No need to hire additional staff in future to handle an administrative activity is an example of a cost-avoidance benefit.
- **Improved-service-level benefits :** Improved-service-level benefits are those where the performance of a system is improved by a new computer-based method. Registering a student in fifteen minutes rather than an hour is an example of this third type of benefit.
- **Improved-information benefits :** Improved-information benefits are where computer based methods, lead to better information for decision making. For example, a system that reports the most-improved fifty customers, as measured by an increase in sales is an improved-information. This information makes it easier to provide better service to major customers.

(4) Social feasibility :

Social feasibility is a determination of whether a proposed project will be acceptable to the people or not. This determination typically examines the probability of the project being accepted by the group directly affected by the proposed system change.

(5) Management feasibility :

It is a determination of whether a proposed project will be acceptable to management. If management does not accept a project or gives a negligible support to it, the analyst will tend to view the project as a non-feasible one.

(6) Legal feasibility :

Legal feasibility is a determination of whether a proposed project infringes on known Acts, Statutes, as well as any pending legislation. Although in some instances the project might appear sound, on closer investigation it may be found to infringe on several legal areas.

(7) Time feasibility :

Time feasibility is a determination of whether a proposed project can be implemented fully within a stipulated time frame. If a project takes too much time it is likely to be rejected.

- **Request Approval :**

  Request can approve by steering committee which consists head of each department, system analyst with owner of the organization. They can also take suggestion of information group committee. All requested projects are desirable or feasible. Projects that are feasible and desirable should be put into a schedule. In some cases, development can start immediately. After a project request is approved software engineering started by determining its cost, priority, completion time, and personnel requirements.

- **Determination of System Requirements :**

  The work product of this phase is system requirement specification document. Requirement of any organization can be decided by asking following questions so that necessary actions like modifying or adding of any operations can be performed. This will improve the quality of system that would give more benefits.

  - What is being done?
  - How is it being done?
  - How frequently does it occur?
  - How great is the volume of transactions or decisions?
  - How well is the task being performed?
  - Does a problem exist?
  - If a problem exists, how serious is it? What is the underlying cause?

  Systems analysts discuss with different category of persons to collect the facts and to study about the business process. Data are collected using fact finding techniques like, interviews, on-site observations and questionnaires but major and data specific information can be obtained first from documentation (manuals or reports).

  When preliminary investigation and Determination of system requirements carried out together sometimes it is known as information engineering.

- **Design of System :**

  The work product of this phase is module specification document. Which contains of algorithm, flow chart and data definition etc? The design describes how a system will satisfy requirements. Systems this stage is also referred as logical design.

  It produces the details that clearly describe how a system will meet the requirements

identified during systems analysis. First identify reports and other outputs system will produce. Pinpoint specific data. Design describes the data to be input, calculated or stored. Individual data items and calculation procedures are written in detail. Designers select file structures and storage devices. The documents containing the design specifications portray the design in many different ways-charts, tables, and special symbols. The detailed design information is passed on to the programming staff for the purpose of software development. Designers are responsible for providing programmers with complete and clearly out lined software specifications.

- **Development of Software :**
  The work product of this phase is software with documentation. This is also known as physical design. Software is the set of related programs. Programmers can write programs using current or required technology and languages. There are options of ready-made software, custom-made software, in-house development of system. The decision depends on available time, money and software specification. In some software development company outsourcing is also one of the options. Programmers also prepare documentation of the program, providing an explanation of how and why certain procedures are coded in specific ways. Documentation used to test the program and carry out maintenance task.

- **Systems Testing :**
  During systems testing, the system is used experimentally to ensure that the software does not fail. In other words, we can say that it will run according to its specifications and in the way users expect. Special test data are input for processing, and the results examined. There are many methods to test software including different levels. These levels are unit testing, integration testing and system testing. There are many methods in each level like white box, black box, alpha, beta, big bang etc. Primary objective is to increase quality of software with validation as per requirement specification document.

- **Implementation, Evaluation and Maintenance**
  Implementation is the process of starting work with new software and new equipment. Before we use new software we have to train users, install the new application and construct necessary data files. Training may be followed by test of employees who has participated in training. If implementation carried out by direct way in which on a specific date working on new system started and working on old system stopped. In case of parallel implementation work on both new and old system will continue up to a specific period of time and later working on old one has been stopped. This gives safety from any hazard with new system. In the case of Phase wise implementation working started with new software in a small area of organization, errors or problems are eliminated if occur then software implemented in other area gradually phase by phase.

  Evaluation of the system is performed to certify the system with its strength and weakness.

  Operational Evaluation : Assessment of the manner in which the system functions, including case of use, response time, overall reliability and level of utilization.

Organizational Impact : Identification and measurement of benefits to the organization in such areas as financial concerns, operational efficiency and competitive impact.

User Manager Assessment : Evaluation of the attitudes of senior and user manager within the organization, as well as end-users.

Development Performance : Evaluation of the development process in accordance with such yardsticks as overall development time and effort, conformance to budgets and standards and other project management criteria.

- **Maintenance :**
  There are four type of maintenance.
  - **Corrective :** Errors are removed when arise for quality improvement.
  - **Adaptive :** Externally when we change operating system, or install new device we have to modify our software to adapt these changes.
  - **Perfective :** To increase profit of the organization or to get additional benefits additional functionalities are added in the requirement specification set.
  - **Preventive :** This type of modification in software gives ease in above three type of maintenance.

  We cannot give grantee of 100% error-free software this indicates life cycle through maintenance. Maintenance is necessary to eliminate errors in the working system during its working life and to tune the system to any variations in its working environment often small system deficiencies are found as a system is brought into operations and changes are made to remove them. System planners must always plan for resource availability to carry out these maintenance functions. The importance of maintenance is to continue to bring the new system to standards.

FACT – FINDING TECHNIQUES

- **Record review :**
  Analyst first get details about the organization from documents, he can examine organization charts and study manuals of different operations which are already documented. Historical reports can be studied by him. He can learn procedure of functioning units. After study he starts on site observations. If we are going to analyze hospital then we have to review records of patient starting from case paper, pathological report, prescription, infrastructure details, timing details, and charges for OPD, operations, laboratory test etc. We must review about reports submitted to head of the hospitals monthly or weekly so that we can have idea of input, process and output. Maximum detail must be gathered this way.

- **Observing the current system :**
  Analyst can better understand by direct observation of working of any organization. Analyst will not inform prior for his site visit. If he informs prior then he cannot observe correctly because people may behave differently to impress analyst. Analyst does not disturb or interview this time to any person but he just verify actual process with documented process. He finds weak areas where improvement is required.

  Some time it is useful to visit another organization with a computerized system similar to our organization. Whenever visiting another organization, an analyst should follow the rules of etiquette: make an appointment, research the organization beforehand, know what he or she wants to see, and write a follow-up, thank you letter.

- **Questionnaires :**
  After observing the system and to confirm weakness or problems Questionnaires gather data from both large and small groups of people. They should be properly constructed. Using these data computer generated reports can be produced. Development of a questionnaire requires in depth planning, and usually more than one draft is necessary.

  These questionnaires are just like feedback form, or questioning by marketing personnel, who do door to door survey. Questions should be short, easy to understand, unbiased, and specific. Now days we can use online questionnaires to get details from people of organization involved with the system.

- **Questions can follow four formats :**
  (1) Multiple choices : This gives respondents a specific set of potential answers. The format is ideal for computer tabulating.

  (2) Open ended : Respondents must answer the question in their own words. Space is provided under each question for the response.

  (3) Rating : This is similar to multiple choices except that respondents must rate their satisfaction.

  (4) Ranking : Rank requires respondents to priorities their responses from high to low or on a percentage basis.

- **Interviewing :**
  Analyst schedules interviews with key personnel involved with the system. There should be preliminary interviews and later detailed interview with all the people who actually operate the system. Interview can be arranged with those who think that computer will replace them. Analyst detects who, what, when, why and how of business operations. The analyst should conduct the interview in such a way that people provide honest descriptions of their jobs. The following questions can help accomplish this goal.

  - Who is involved with what you do?
  - What do you do?
  - Where do you do it?
  - When do you do it?

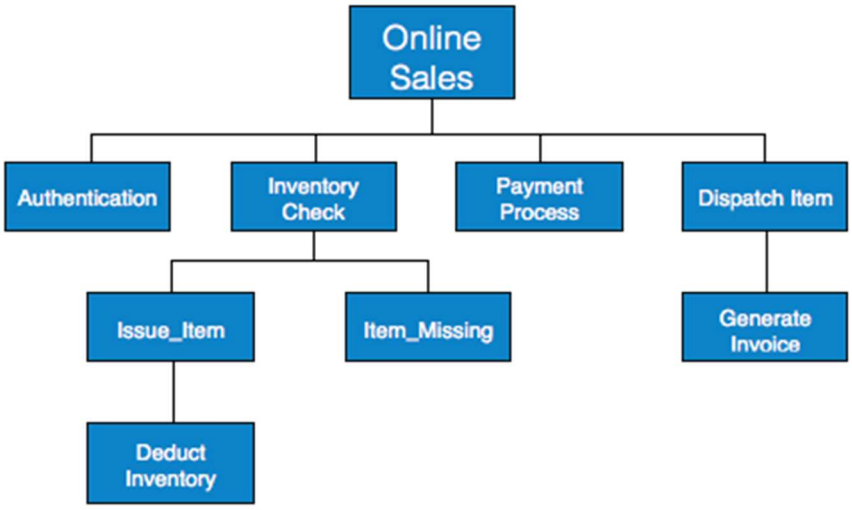
- Why do you do it the way you do?
- How do you do it?
- Do you have suggestions for change?

Interviews help gather essential facts about existing problems, such as lack of quality control or sufficient security, but they also allow the analyst to involve people in change, easing them into it.

UNIT- II: DESIGNING THE SYSTEM WITH CASE & CAD TOOLS.

HIPO

A HIPO diagram, also known as a hierarchical input-process-output diagram, is a visual tool used to analyze and improve processes within an organization. It is often used in quality management and business process improvement. The diagram helps to identify the inputs, processes, and outputs of a system or process, as well as the potential causes of issues or problems. It can be used to streamline operations, identify areas for improvement, and enhance overall efficiency and effectiveness.



HIPO diagram represents the hierarchy of modules in the software system. Analyst uses HIPO diagram in order to obtain high-level view of system functions. It decomposes functions into sub-functions in a hierarchical manner. It depicts the functions performed by system.

HIPO diagrams are good for documentation purpose. Their graphical representation makes it easier for designers and managers to get the pictorial idea of the system structure.

MS-VISIO FOR DESIGNING AND DOCUMENTATION

Microsoft Visio is a drawing tool which is useful to draw various diagrams in different areas. People use Visio to create a great variety of drawings, ranging from network diagrams to calendars and from office layouts to flowcharts.

We have two kinds of versions of Visio.

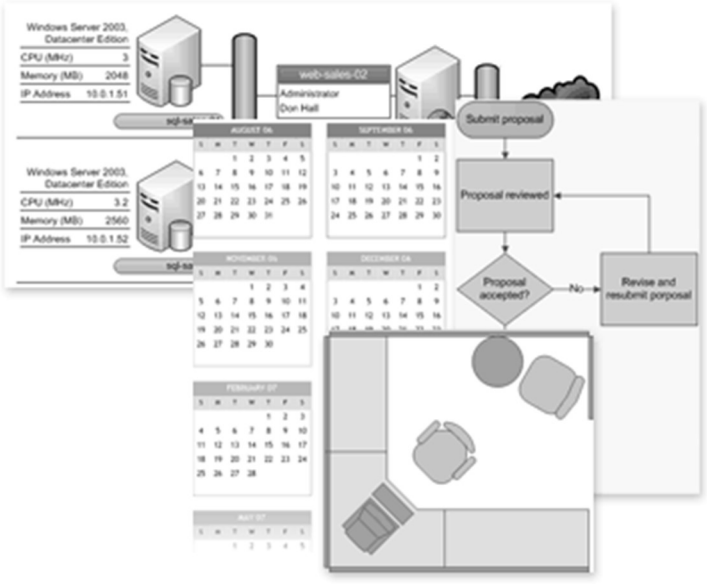
(1) Standard

(2) Professional

(1) Standard : This can be used by business professionals like product managers, financial analysts, marketing professionals. This software has ability to create business related charts and diagrams that illustrate business processes, marketing trends, organizations, project schedules, and so on. It can also used for class room arrangement in a school building or to show any kind of organization.

(2) Professional : This is superset of Visio Standard. This can be used by IT Manager, Network Administrator, electrical engineers, Web designers and software developers. It demonstrates a wide variety of technical concepts and processes. In this we can import details from any DBMS like Access and use it to create a special diagram.

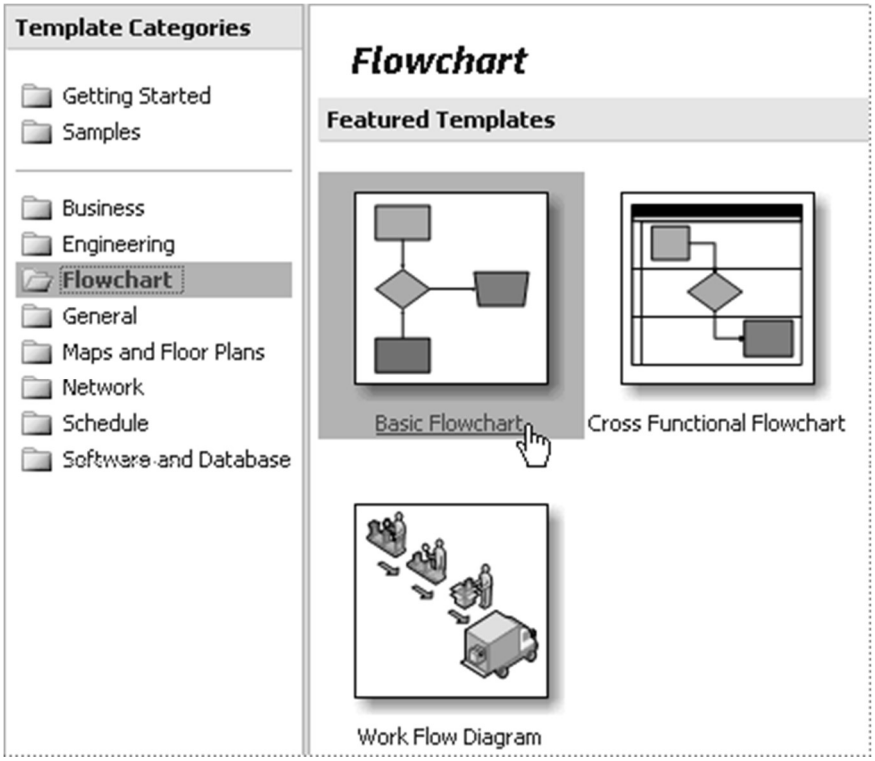
Like insert shapes in Ms Word we have different types of readymade shapes for flowchart, electrical diagram, business diagram, network chart etc. We need not to draw it from scratch. We can modify its size, appearance and easily connect them each other.



- We have to use Drag and drop method using that Visio creates drawings. We drag shapes and we drop them onto a drawing page. A *shape* stand for an object. It represents office furniture, electrical components of circuit, flow chart symbols, a road sign in a directional map, a server in a network diagram, a box on an organization chart, or a bar on a comparison chart. Visio contains literally thousands of shapes.
- Connector is a line which connects one shape to another.

We can create a drawing in 3 basic steps. There are many kinds of Visio drawings, but we can use the same three basic steps to create nearly all of them.

- Choose and open a template.
- Drag and connect shapes.
- Add text to shapes.



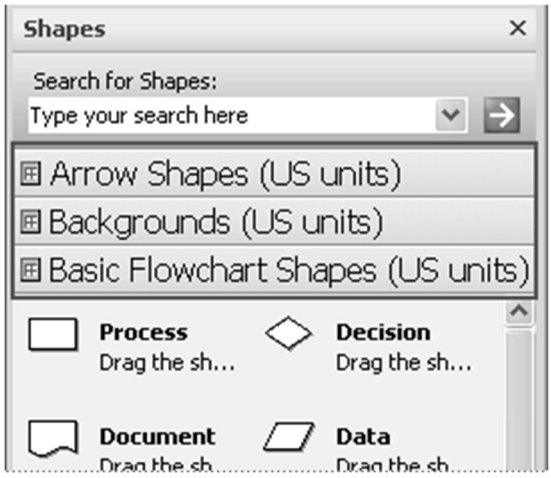
To draw a flow chart performs following steps.

Start Visio 2007.

In the Template Categories list, click Flowchart.

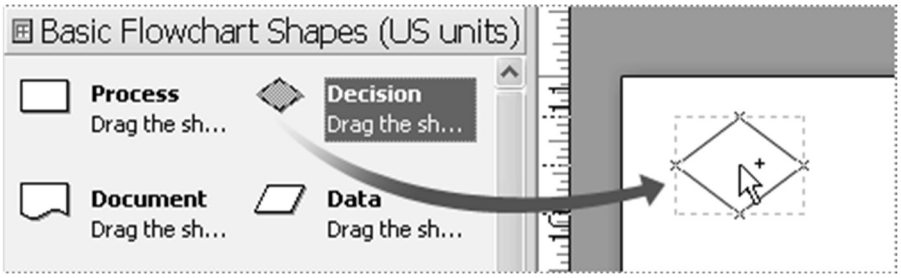
In the Flowchart window, under Featured Templates, double-click Basic Flowchart.

The shapes what we required can be dragged from stencils which is available in the left as a part of template. The stencils that open with the Basic Flowchart template are called Arrow Shapes, Backgrounds, and Basic Flowchart Shapes.



We can drag shapes from stencils onto the blank drawing page and connect them to one another. There are many ways to connect shapes one of them is that drag the shapes on top of each other to connect them automatically by using Auto Connect. For more information, see Add and glue connectors with Auto Connect.

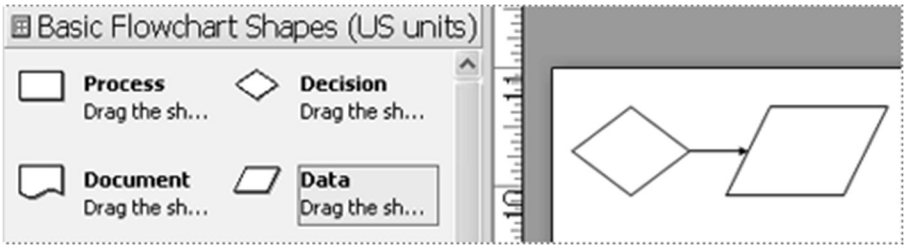
Drag our first shape from the Basic Flowchart Shapes stencil onto the drawing page, and then release the mouse button.



Drag our second shape on top of the first so that the blue arrows show, but don't release the mouse button yet.

While holding the mouse button, move our pointer on top of the blue arrow that points toward where we want to place the second shape.

Now release the mouse button. Our shapes are connected, and the first shape points to the second shape.



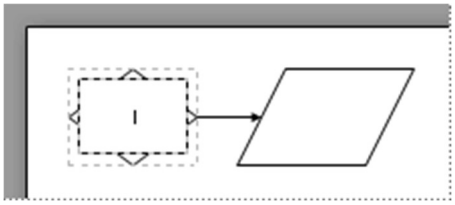
Continue to build our drawing by repeating steps.

Now Add text to shapes.

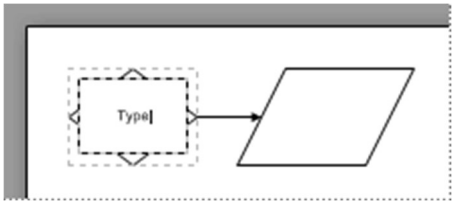
Although some drawings make a point all by themselves, it's often helpful and sometimes necessary to add text to the shapes. If we want to learn more ways to add text to shapes, see Add data to shapes and Add imported data to shapes.

Add text directly to a shape.

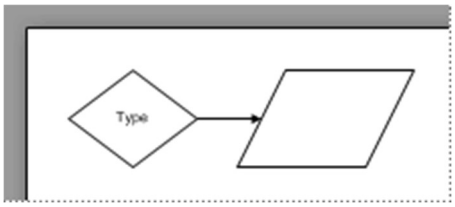
Double-click the shape.



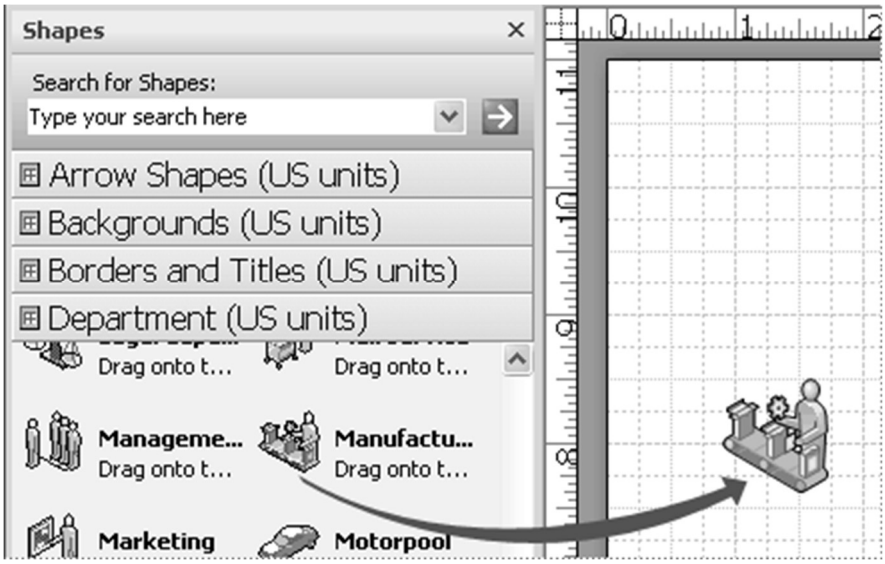
Start typing.



When we finish typing, click on a blank area of the drawing page.

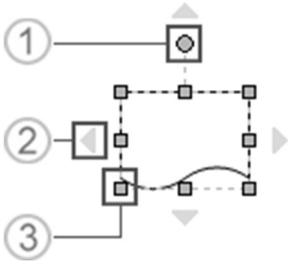


- **Shapes :**
  Visio shapes are ready-made images that we drag onto our drawing page. They are the building blocks of our drawing. When we drag a shape from a stencil onto our drawing page, the original shape remains on the stencil. That original is called a master shape. The shape that we put on our drawing is a copy called an instance of that master. From most Visio stencils, we can drag as many instances of the same shape onto our drawing as we want.



Customize shapes on the spot.

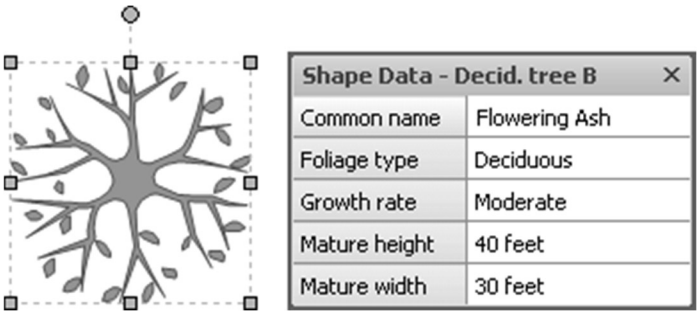
There are thousands of Visio shapes and countless ways to use and customize them. The most common things that people do with shapes involve features that are built right into the shapes. Visual cues help we find and use those features quickly.



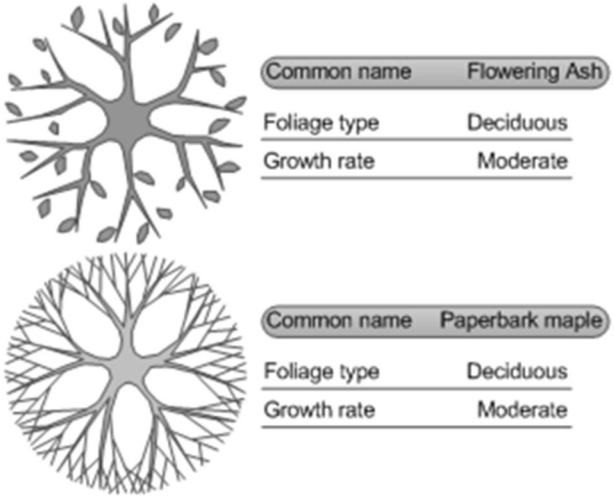
- **1 is Rotation handles :**
  The bright green dots located above shapes are called rotation handles. Drag a rotation handle right or left to rotate the shape.

- **2 is Blue connection arrows for Auto Connect :**
  The light blue connection arrows help we easily connect shapes to one another, as we saw in the previous section, Create a drawing in three basic steps. However, there is more than one way to use Auto Connect, so for more information, see Add and glue connectors with Auto Connect.
- **3 is Selection handles for resizing shapes :**
  We can use the bright green selection handles to change the height and width of our shape. Click and drag a selection handle on the corner of a shape to enlarge the shape without changing its proportions, or click and drag a selection handle on the side of a shape to make the shape taller or wider.

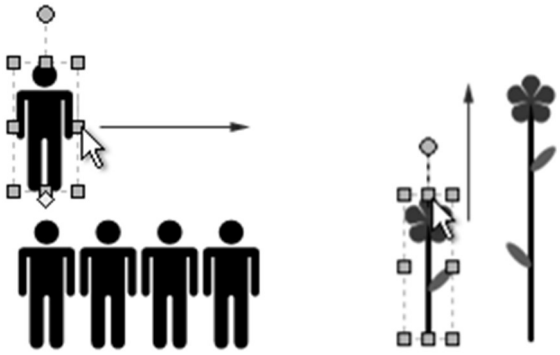
  Visio shapes are much more than simple images or symbols. These shapes can hold data. Each shape can be associated with data. We can have options like Add data to shapes and Add imported data to shapes. After data is added to a shape, it isn't displayed in the drawing by default. Using open the Shape Data window we can see data along with shapes.



  If we want to display the data for lots of shapes at once, we can use a feature called Data Graphics. The following illustration shows the data for two trees at once. For more information about using Data Graphics, see Enhance our data with data graphics.

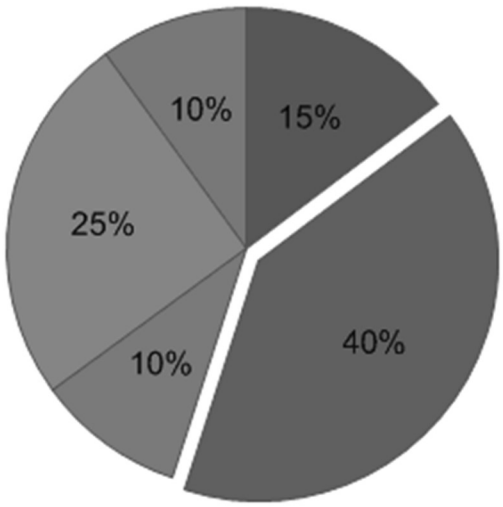


We can stretch a People shape to show more people as shown below or stretch the Growing flower shape to indicate growth.



We can measure the size of a shape on our page by using a special dimension shape that is designed for measuring other shapes.

Below is a Pie chart shape from the Charting Shapes stencil. We can right-click the shape to set the number of slices and percent of each slice.

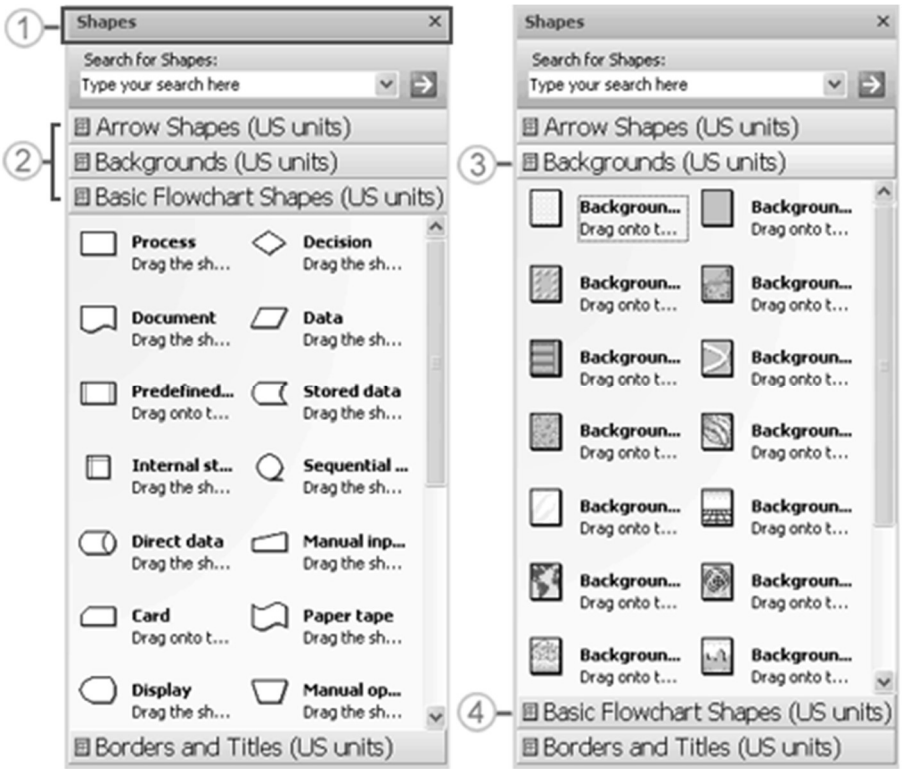


- **Stencils :**

  Visio stencils hold collections of shapes. The shapes in each stencil have something in common. The shapes can be a collection of shapes that we need to create a particular kind of diagram, or several different versions of the same shape.

  The Basic Flowchart Shapes stencil contains common flowchart shapes, and the

Backgrounds stencil contains a variety of backgrounds. We can create our own stencil of favorite shapes using Create a new stencil option.



(1) Stencils appear in the Shapes window.

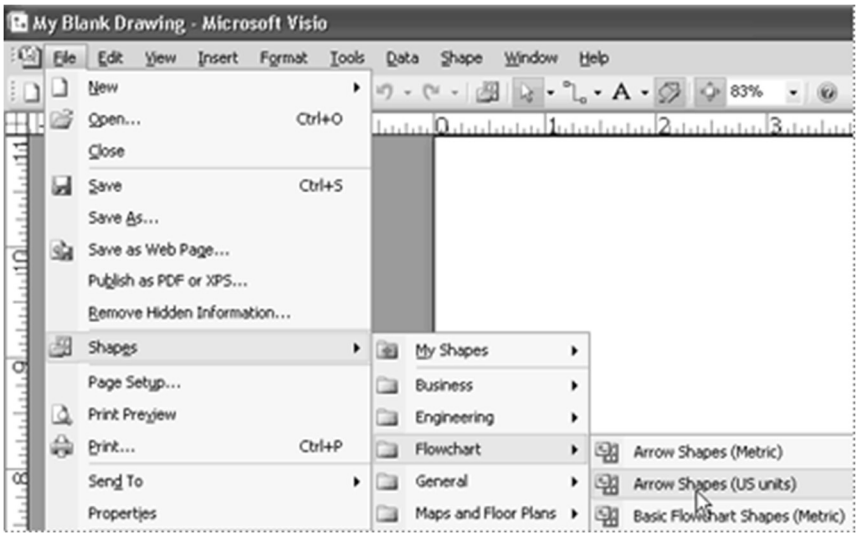
(2) When stencils open, they automatically dock themselves in the Shapes window, one on top of another.

(3) Click the title bar of a stencil to bring it to the top of the stack.

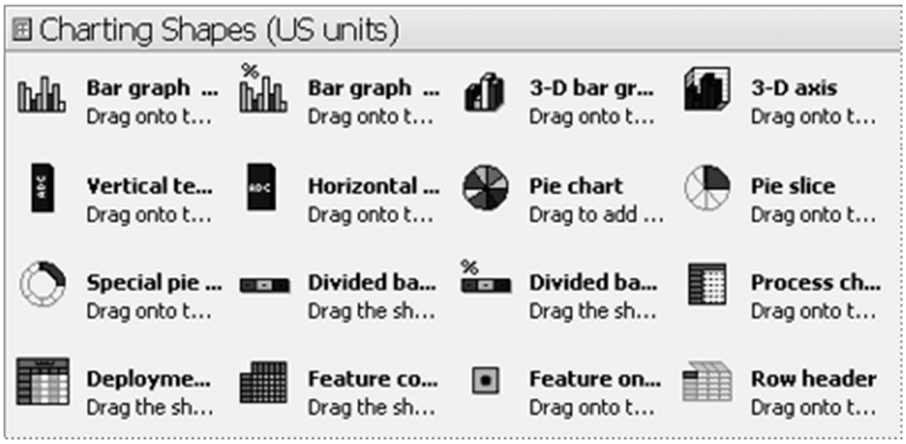
(4) The stencil that was previously on top of the stack moves to the bottom of the Shapes window.

- **Template :**

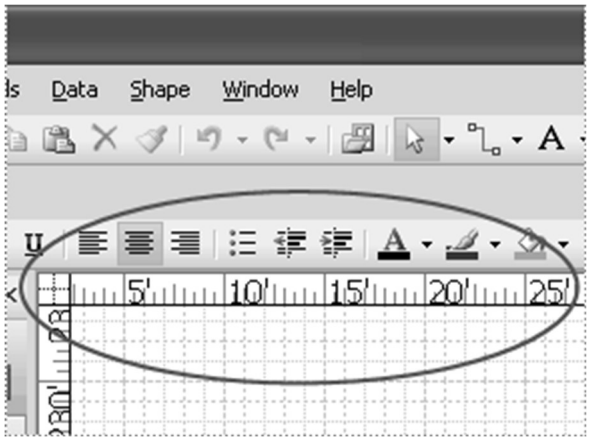
Template opens with the stencils that we need to create a particular kind of drawing. On the File menu, point to Shapes, point to the category that We want, and then click the name of the stencil that we want to use.



Visio templates are like a collection of settings. A Visio template combines a blank drawing page with any combination of Stencils full of the shapes that are needed to create a particular kind of drawing. The Charts and Graphs template, for example, opens with a stencil full of quick, easy shapes for creating charts and graphs.

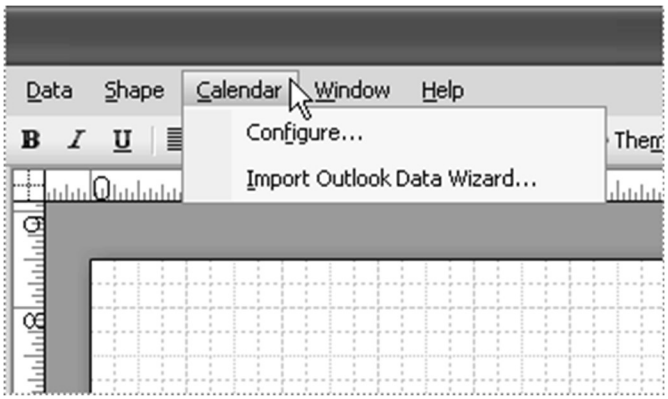


Some drawings require a special scale. For example, the Site Plan template opens with an engineering scale, where 1 inch equals 1 foot. For this need we can use appropriate grid size and ruler measurements.



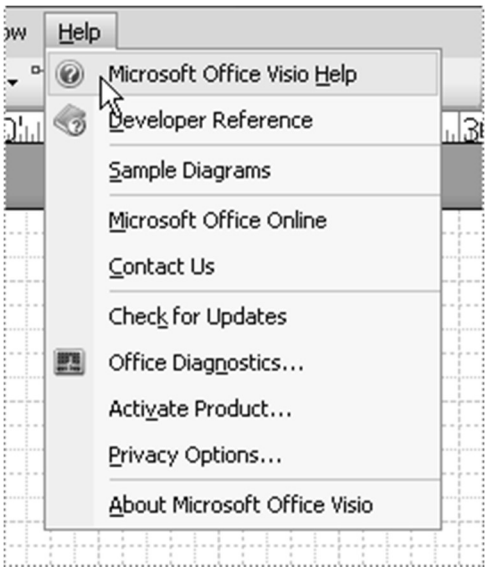
Special menus some templates have unique features that we can find on special menus.

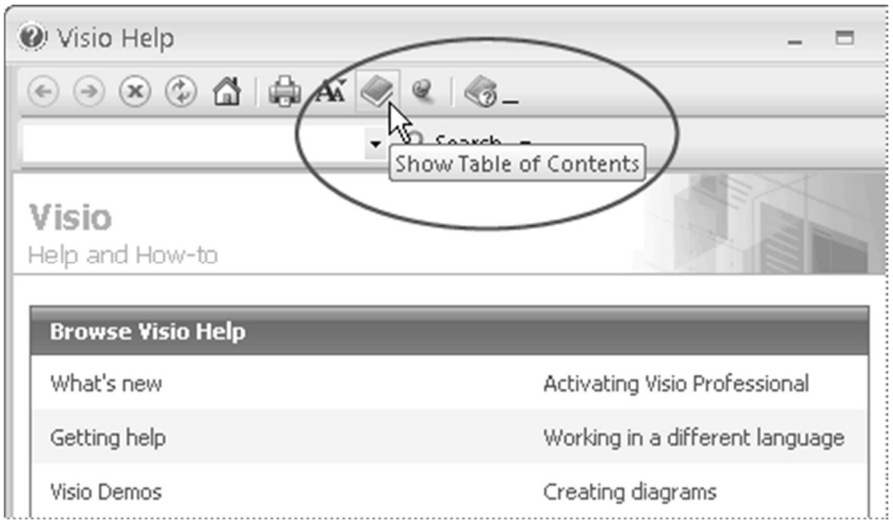
For example, when we open the Calendar template, a Calendar menu appears on the main menu bar. We can use the Calendar menu to configure our calendar or to import data from Microsoft Office Outlook into Our calendar.



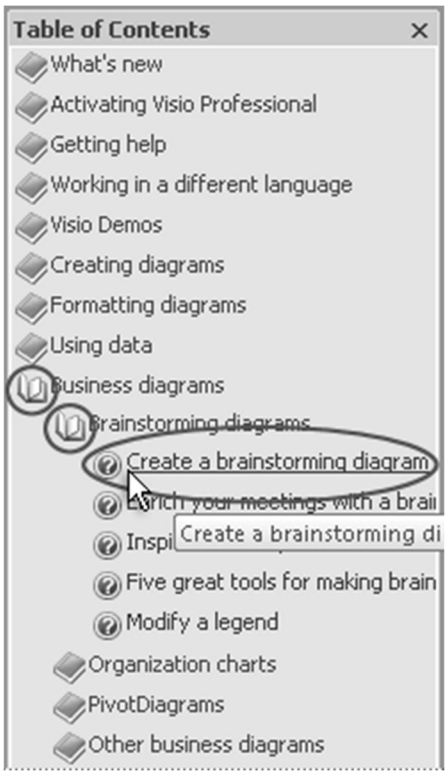
Wizards to help we with special types of drawings in some cases when we open a Visio template, a wizard helps we get the drawing off to a good start.

On the Help menu in Visio, click Microsoft Office Visio Help.

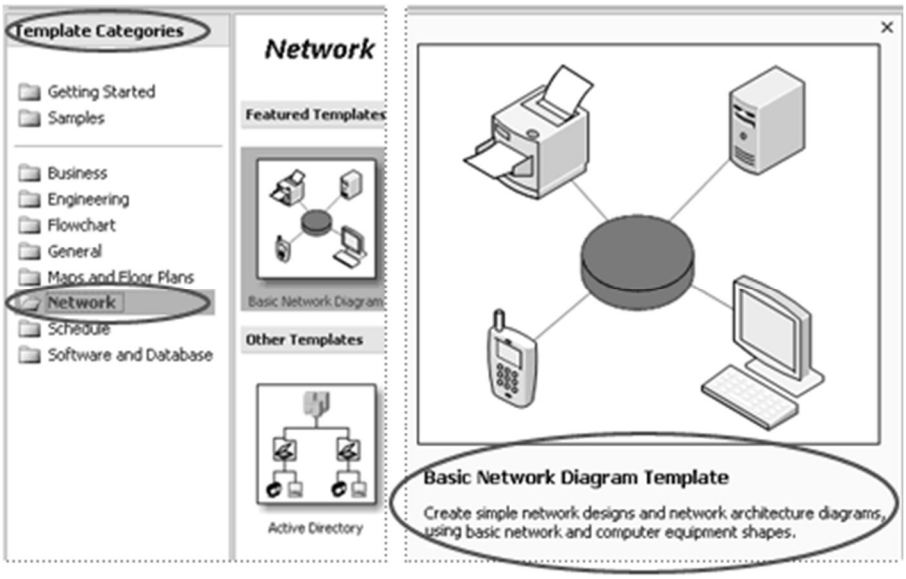




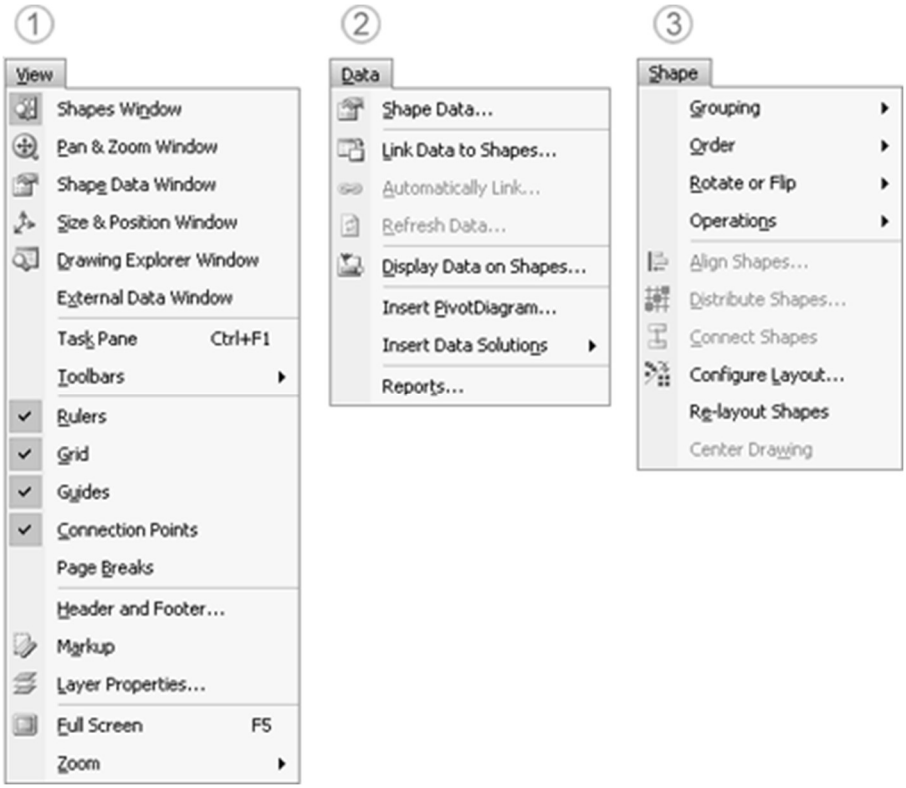
In the Table of Contents window, click the category for the drawing that we want to create.



We can find all kinds of information in Visio and we have variety of Visio templates which opens automatically when we start Visio.



Data, Shape and View menus are three of the most popular menus to get options for doing required task quickly.



(1) The View menu lists all the special windows for Visio drawings as well as the commands for turning on and off the visual guides, like the drawing grid.

(2) The Data menu lists the commands for advanced features, like importing and displaying data. These are some of the most powerful features available in Visio.

(3) The Shape menu lists the commands that orient our shapes and connectors the

way that we want them in our drawing.

INTRODUCTION OF UML

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the piece of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

- **Goals of UML :**

  The primary goals in the design of the UML are followings.

  (1) Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.

  (2) Provide extensibility and specialization mechanisms to extend the core concepts.

  (3) Be independent of particular programming languages and development processes.

  (4) Provide a formal basis for understanding the modeling language.

  (5) Encourage the growth of the OO tools market.

  (6) Support higher-level development concepts such as collaborations, frameworks, patterns and components.

  (7) Integrate best practices.

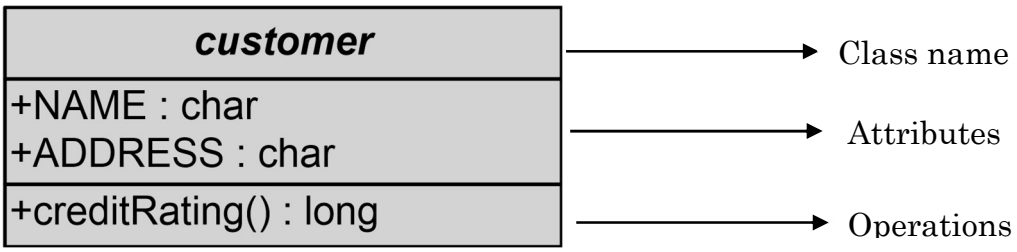
- **Why Use UML?**

  As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.

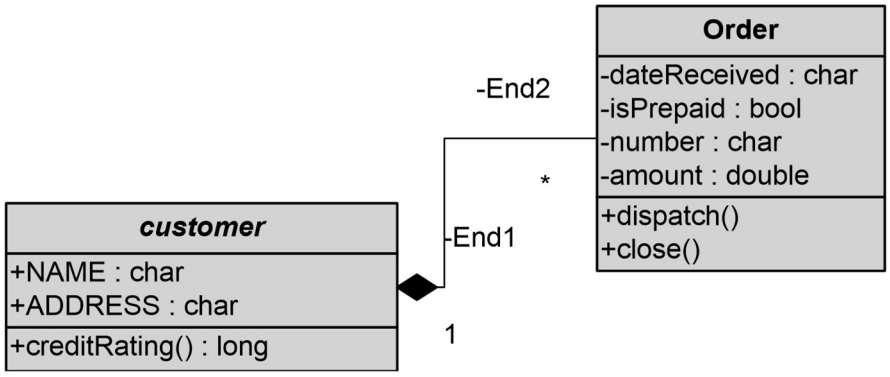
CLASS DIAGRAMS

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different

perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. This example is only meant as an introduction to the UML and class diagrams. Classes are composed of three things: a name, attributes, and operations. Below is an example of a class.

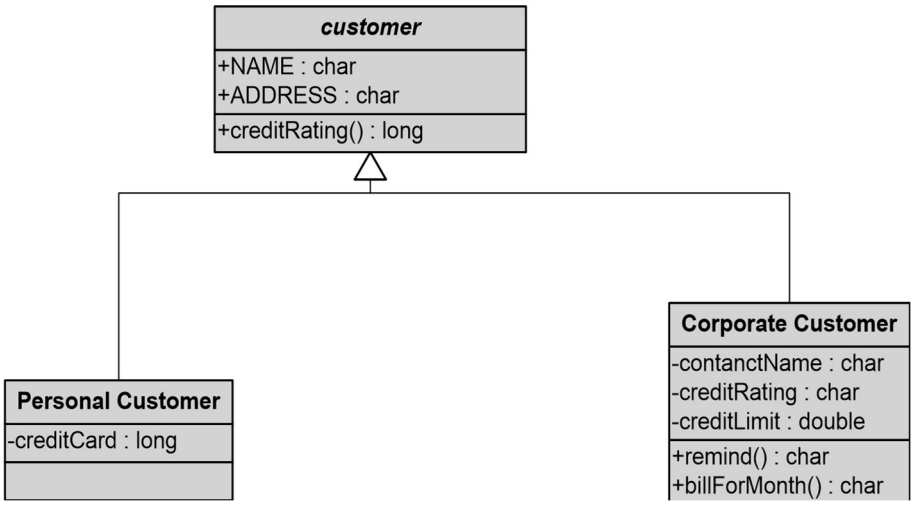


Class diagrams also display relationships such as containment, inheritance, associations and others. Below is an example of an associative relationship:



The association relationship is the most common relationship in a class diagram. The association shows the relationship between instances of classes. For example, the class Order is associated with the class Customer. The multiplicity of the association denotes the number of objects that can participate in then relationship. For example, an Order object can be associated to only one customer, but a customer can be associated to many orders.

Another common relationship in class diagrams is a generalization. A generalization is used when two classes are similar, but have some differences. Look at the generalization below.



In this example the classes Corporate Customer and Personal Customer have some similarities such as name and address, but each class has some of its own attributes and operations. The class Customer is a general form of both the Corporate Customer and Personal Customer classes. This allows the designers to just use the Customer class for modules and do not require in-depth representation of each type of customer.

- **When to Use Class Diagrams?**

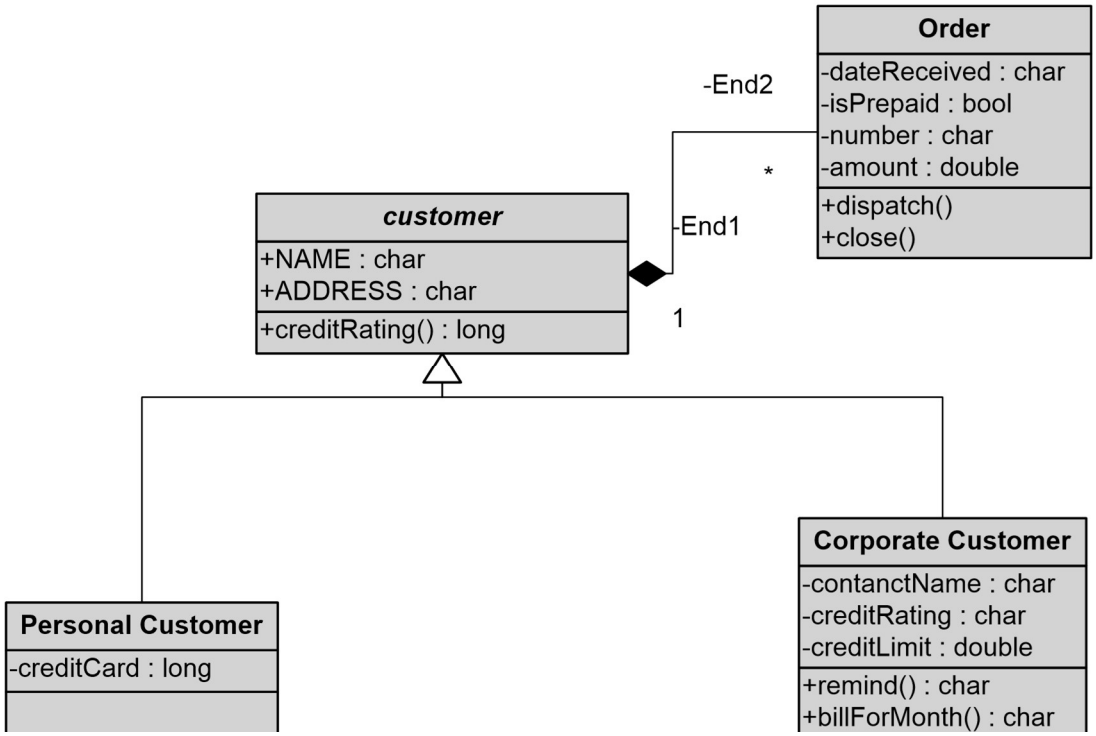
  Class diagrams are used in nearly all Object Oriented software designs. Use them to describe the Classes of the system and their relationships to each other.

- **How to Draw Class Diagrams?**

  Class diagrams are some of the most difficult UML diagrams to draw. To draw detailed and useful diagrams a person would have to study UML and Object Oriented principles for a long time. Therefore, this page will give a very high level overview of the process. To find list of where to find more information see the Resources page.

  Before drawing a class diagram consider the three different perspectives of the system the diagram will present; conceptual, specification, and implementation. Try not to focus on one perspective and try seeing how they all work together.

  When designing classes consider what attributes and operations it will have. Then try to determine how instances of the classes will interact with each other. These are the very first steps of many in developing a class diagram. However, using just these basic techniques one can develop a complete view of the software system.



This example is only meant as an introduction to the UML and use cases.

ACTIVITY DIAGRAMS

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.

- **When to Use Activity Diagrams?**

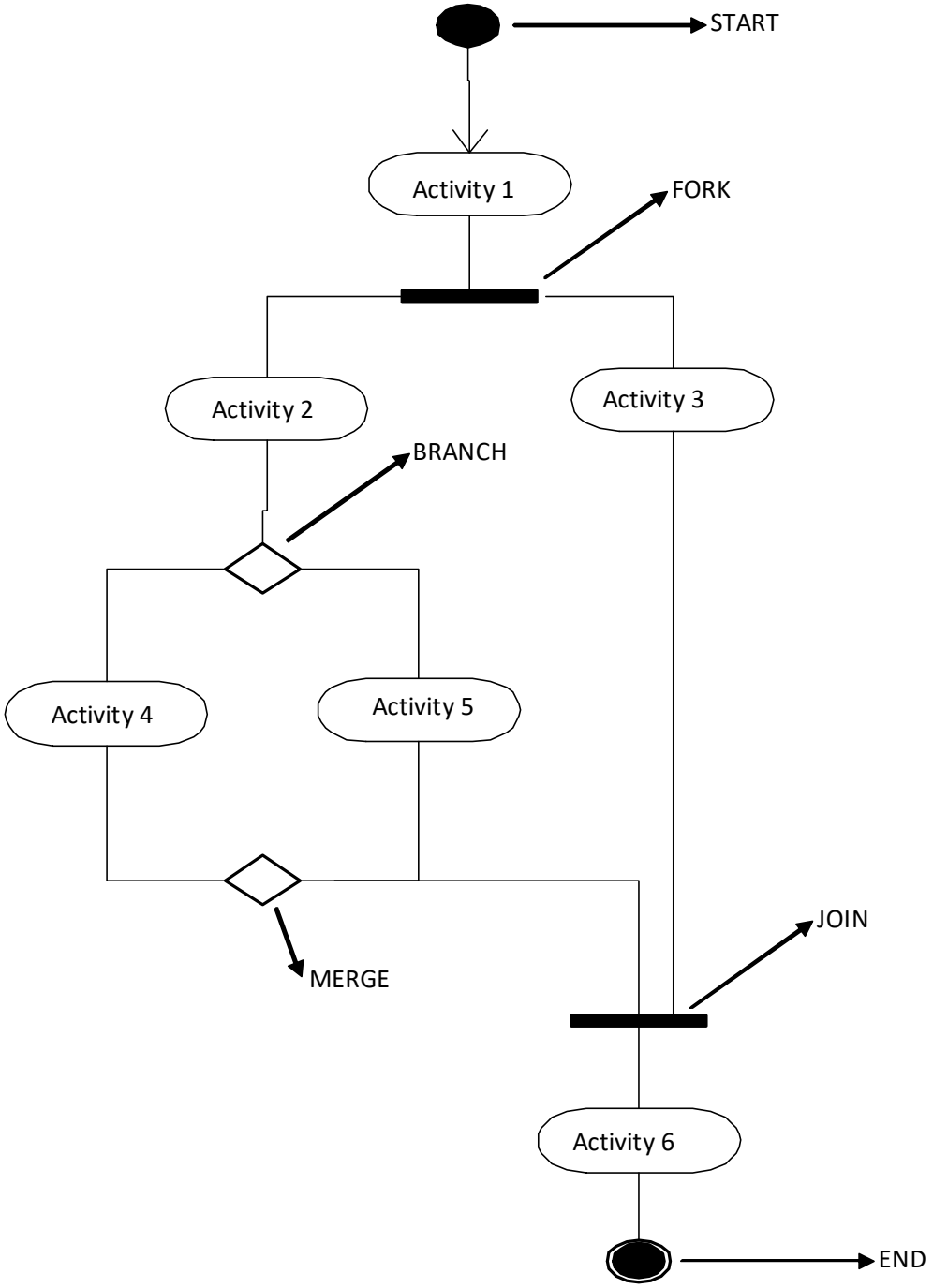
  Activity diagrams should be used in combination with other modeling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to form the workflow behind the system being designed. Activity Diagrams are also useful for analyzing a use case by describing what actions need to take place and when they should occur, describing a complicated sequential algorithm and modeling applications with parallel processes.

  However, activity diagrams should not take the place of interaction diagrams and state diagrams. Activity diagrams do not give detail about how objects behave or how objects collaborate.

- **How to Draw Activity Diagrams?**

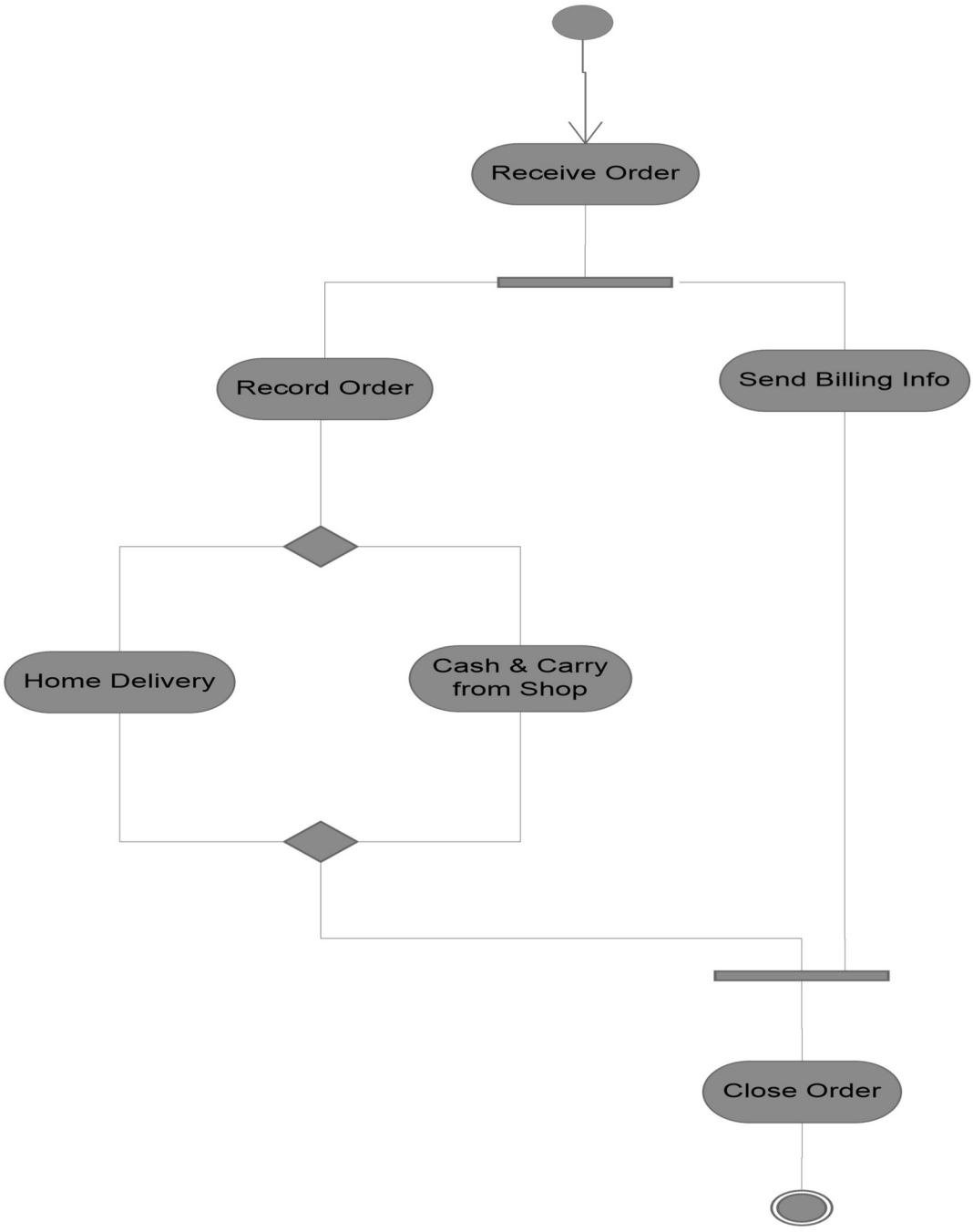
  Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities. A fork is used when multiple activities are occurring at the same time. The diagram below shows a fork after activity1. This indicates that both activity2 and activity3 are occurring at the same time. After activity2 there is a branch. The branch describes what activities will take place based on a set of conditions. All

branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch. After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.



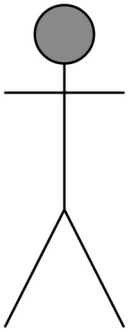
Below is a possible activity diagram for processing a Pizza order. The diagram shows the flow of actions in the system's workflow. Once the order is received the activities split into two parallel sets of activities. One side fills and sends the order while the other handles the billing. On the Fill Order side, the method of delivery is decided conditionally. Depending on the condition either the Home Delivery activity or Cash

& Carry from shop activity is performed. Finally the parallel activities combine to close the order.



USE CASE DIAGRAMS

Use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.



Actor

An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

- **When to Use "Use Case" Diagrams?**

  Use cases are used in almost every project. These are helpful in enlightening requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

- **How to Draw Use Cases Diagrams?**

  Use cases are a relatively easy UML diagram to draw, but this is a very simplified example. This example is only meant as an introduction to the UML and use cases.

  Start by listing a sequence of steps a user might take in order to complete an action. For example a user placing an order with a book seller company might follow these steps.

  (1) Browse catalog and select book.

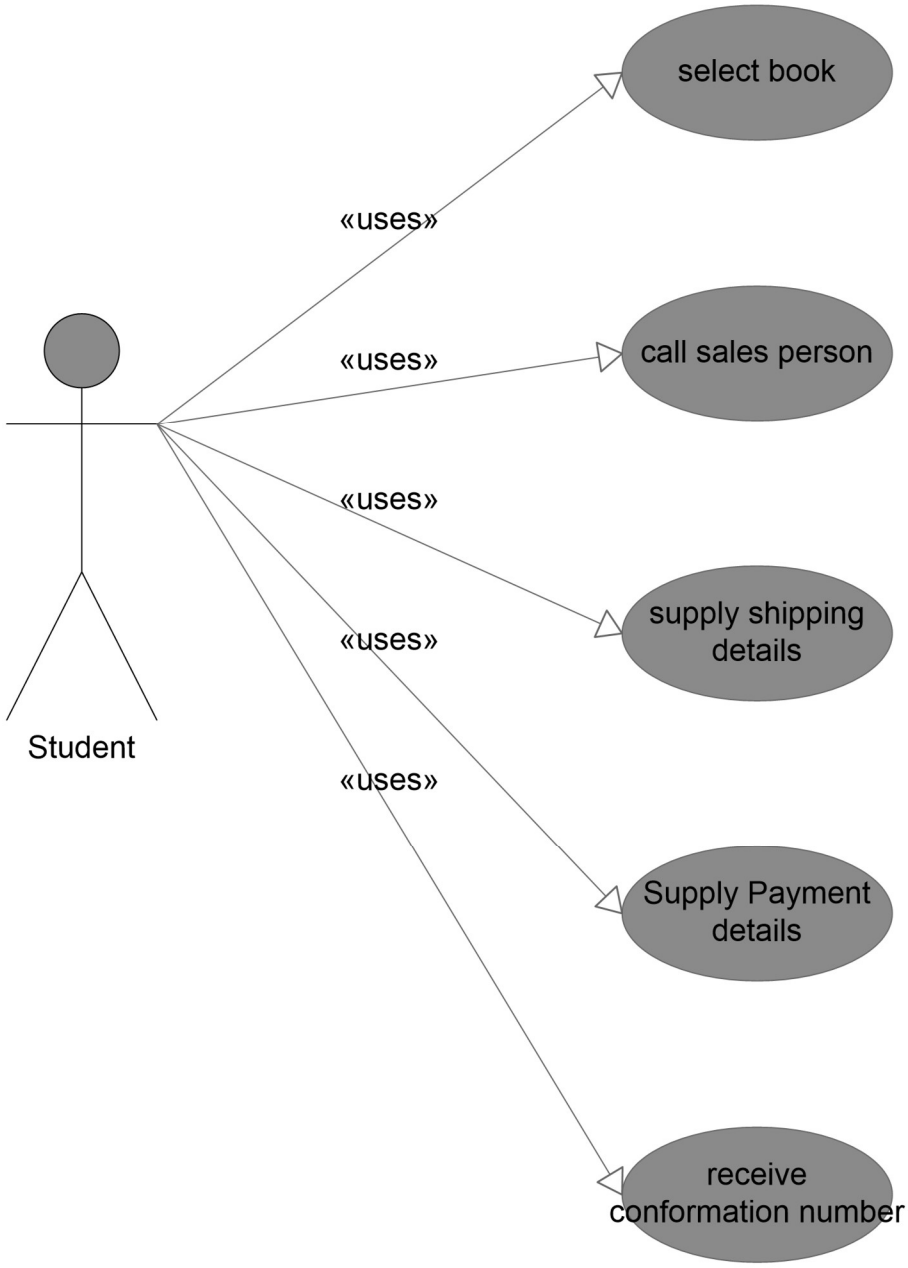
  (2) Call sales representative.

  (3) Supply shipping information.

  (4) Supply payment information.

  (5) Receive conformation number from salesperson.

  These steps would generate this simple use case diagram



This example shows the student as an actor because the student is using the book ordering system. The diagram takes the simple steps listed above and shows them as actions the student might perform. The sales person could also be included in this use case diagram because the sales person is also interacting with the book ordering system.

From this simple diagram the requirements of the book ordering system can easily be derived. The system will need to be able to perform actions for all of the use cases listed. As the project progresses other use cases might appear. The student might have a need to add a book to an order that has already been placed. This diagram can easily be expanded until a complete description of the book ordering system is derived capturing all of the requirements that the system will need to perform.

INTRODUCTION TO SOFTWARE ENGINEERING

The technology which cover a process, a set of methods – frame work, and an array of tools that we call software engineering.

Computer software is the product that software engineers design and build by applying a process (software engineering approach) that leads to a high – quality result that meets the needs of the people who will use the software.

Software is both a product and a Tool for producing a product.

- **Products (Software) :**
  It delivers the computing potential embodied by computer hardware or a network of computers that are accessible by local hardware. Whether it resides within a cellular phone or operates inside a mainframe computer, software is an information transformer – producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as a single bit or as complex as a multimedia presentation.

- **Tool for delivering a Product :**
  We develop our product (Software) using another software which includes operating system, languages, packages, database, computer aided software development tools. We also use network and network software to accomplish developing task of new software.

  To develop software Project Management required. Which includes planning, monitoring and controlling of the people, process, and events in the software industry as software evolves from a preliminary concept to an operational implementation.

  Everyone "manages" to some extent but the scope of management activities varies with the person doing it.

Person	Manages
Software Engineer	day to day activities
Project Manager	work of team of software engineers
Senior manager	Coordinates the interface between the business and the software professionals

:: EXERCISE ::

[1] Explain following terms. system, subsystem, Information Analyst, System Designer, Programmer Analyst, Module Specification Document, SRS document.

[2] Write Benefits of Using Software.

[3] Write Role of system analyst.

[4] Write Attributes of System Analyst.

[5] Write short note on SDLC.

[6] Explain Feasibility.

[7] Write short note on fact finding technique.

[8] Write rules for Data Flow Diagram.

[9] Explain types of Data Dictionary.

[10] What is Software Engineering?

DATA FLOW DIAGRAM (BUUBLE CHART)

- **What is DFD?**

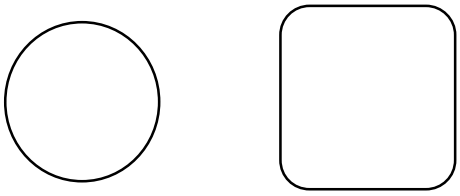
  Data is the life blood of any system. Diagram of flow of data in system and its processing which converts data into valuable information is known as Data Flow Diagram.

  It will not show logic of the algorithm. It shows only flow of the data from the process - to the process or from the table – to the table or from the external source - to the external destination.

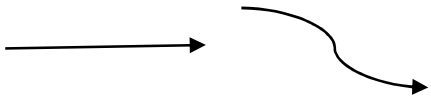
  To make data flow diagrams, we use : Arrows, circles, open-ended boxes, and squares.

  An arrow shows flow of the data. Data is in motion. Circles are used for processes that converts data/into in- formation. An open-ended box represents a temporary storage of data. A square may be an external entity or originator of data or receiver of information - reports.

- **Symbols for DFDs :**



Circle or Rounded rectangles used for process. Name of process must start with verb followed by noun.

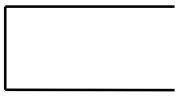


All arrows shows data flow in motion. Data name must be written on the arrow.

Example : student record, inquiry details etc.

Squares are used to show external source or destination. Give name of external entity.

Example : Manager, Customer



Open ended box is used to show table / data store. (Group of entity – records). Since it is the group of external entity. Its name must be plural.

Example : employees, students.

- **Rules :**

  Arrows should not cross each other.

  Squares, circles, and files must be properly named. Choose meaningful names. No two data flows, squares, or circles can have the same name.

  Circles (Processes) must have name started with verb and followed by noun.

  Open ended rectangles (Tables) must have names in plurals.

  Process gets input but not gives output it is an error. It is known as "infinite Sink".

  Process do not takes input but gives output it is an error. There is only one program "random number generation" which produces output without input.

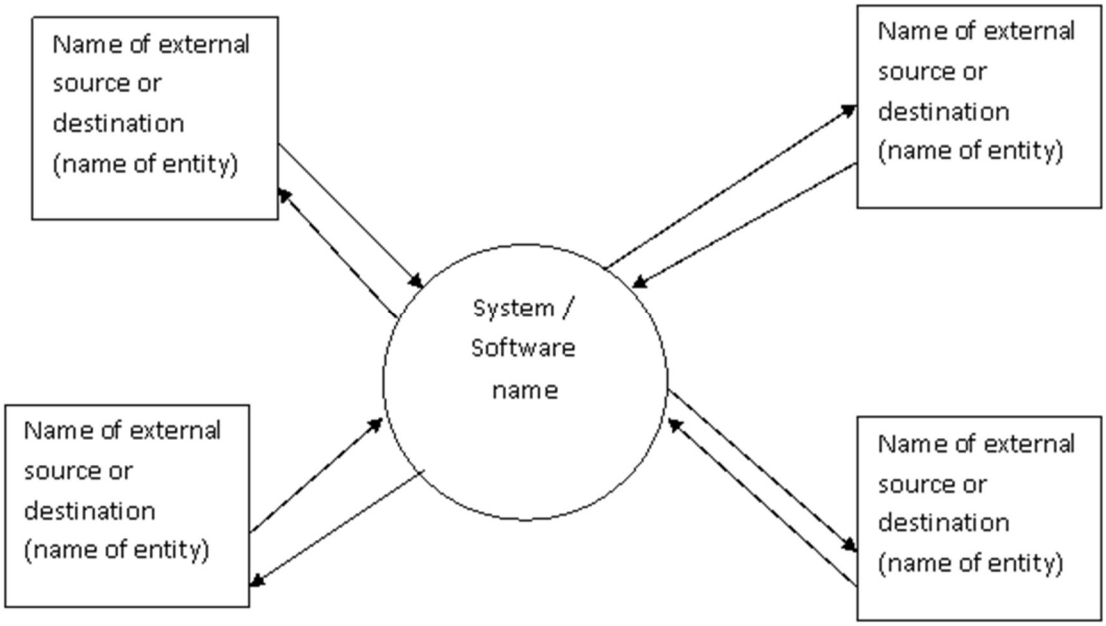
  Joining of two data store, two processes directly is not appropriate. Process gives output to temporary or permanent data store then data store provides data to another process.

  Data store, external entity (source or destination), process must be shown in the diagram only once. Repetition is not allowed.

  Decomposed DFD must be consistent with the original DFD.

- **Context Diagram :**

  Following diagram is known as context diagram. This diagram shows that for which system we are going to discuss and who are taking participate in it (external source and destination).

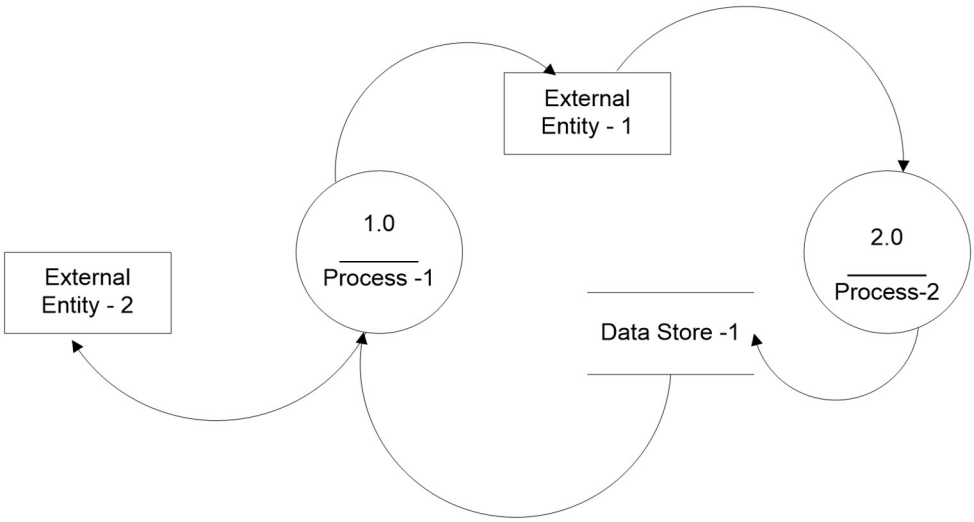


- **Zero level diagrams :**
  List out all activity in the system. These activities are processes which take input and produce outputs. In following table write name of process, input name in Data store column if input comes from table or external source name if input comes outside the system or both. Similarly we do for output. Any process gives output to data store or to external source.

  To draw 0 - level diagram can be prepared using following table.

Process No.	Activity (process)	Input From		Output To	
		Data Store	External Source	Data Store	External Destination
1.0					
2.0					
3.0					
◆					
◆					

After preparation of this table you can start drawing connecting bubble with each other by taking data store between them. Arrows will connect bubble to data store, data store to bubble, external source to bubble, bubble to external destination.



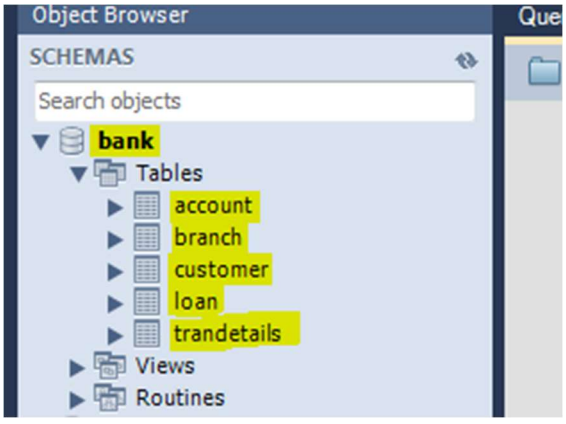
Generally in major database applications process to process connections are avoided and temporary data store inserted between them. In above figures all processes numbers are like 1.0, 2.0 etc... So this is 0 levels DFD. We can write process numbers along with the name of process in the circles.

Q. Draw a Data Flow diagram for Vishal Travelles for ticket booking, sale, and cancellation, money transactions with different types of buses with seat and sleeper facilities. If customer fails to collect booked ticket before 30 min. of the departure ticket will be automatically cancelled and can be allotted to another. Produce financial reports bus-wise, driver wise, rout wise periodically. Manager also want to know expense report, maintenance etc.

Drawing Table Structure of database and interrelation of tablesOr prepare database including primary key and foreign key concepts using GUI interface of MySQL or Microsoft SQL Server.

Create ER Diagram of a Database in MySQL Workbench

First make sure you have a **Database** and **Tables** created on the MySQL server.

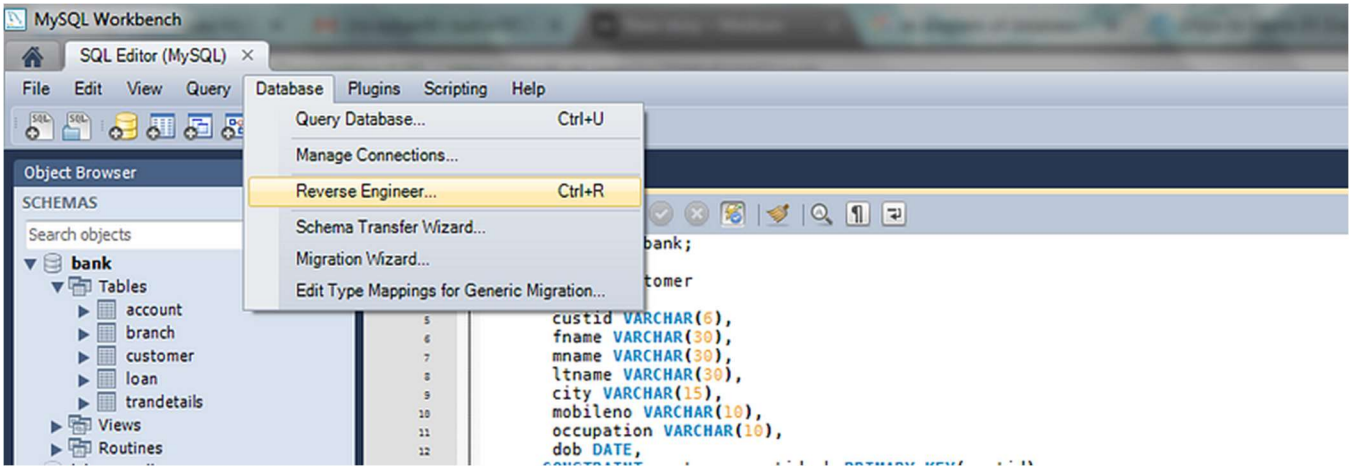


Example :-

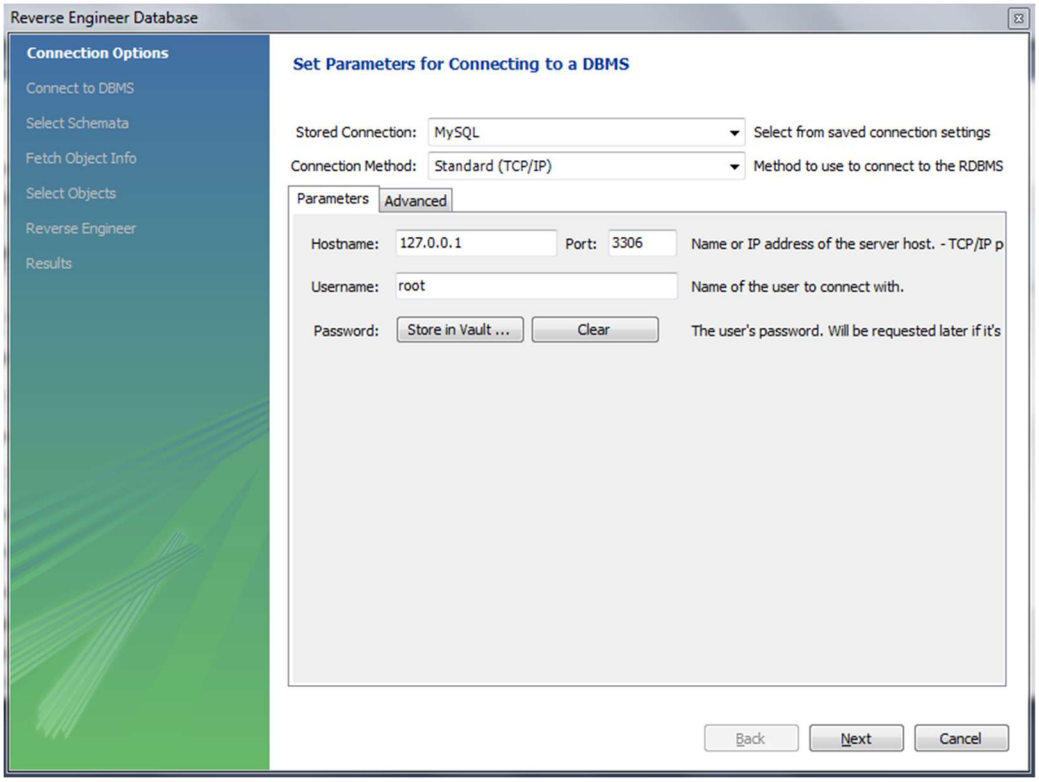
Database - *bank.*

Tables - *account, branch, customer, loan, trandetails.*

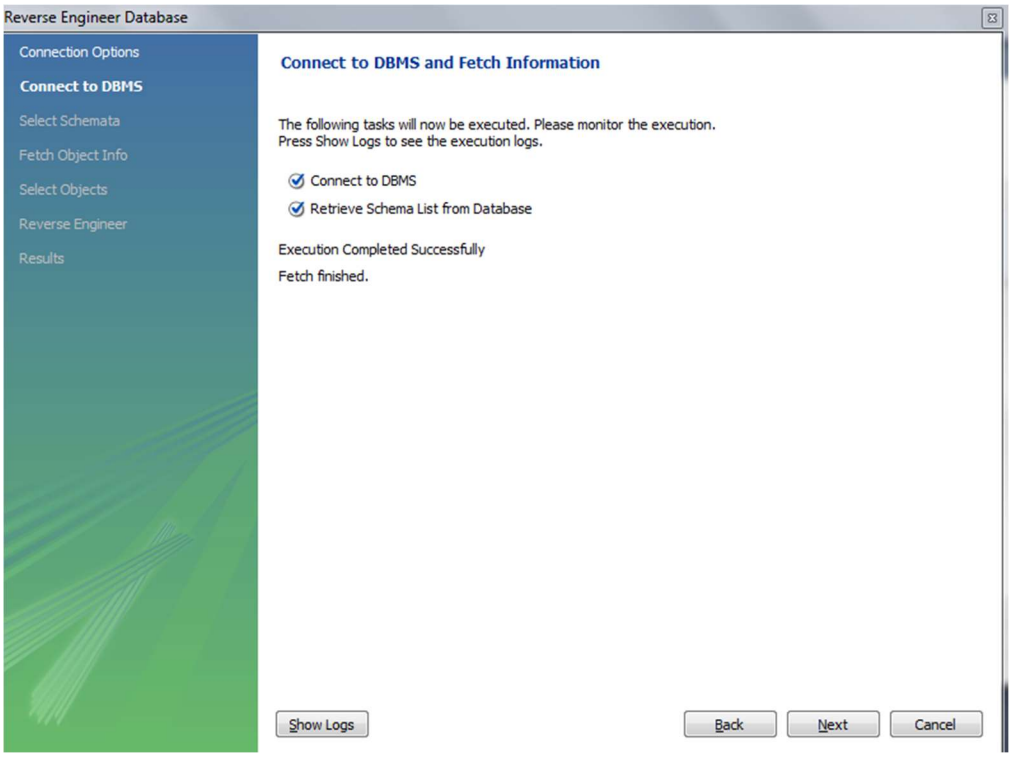
2. Click on **Database** -> **Reverse Engineer**.



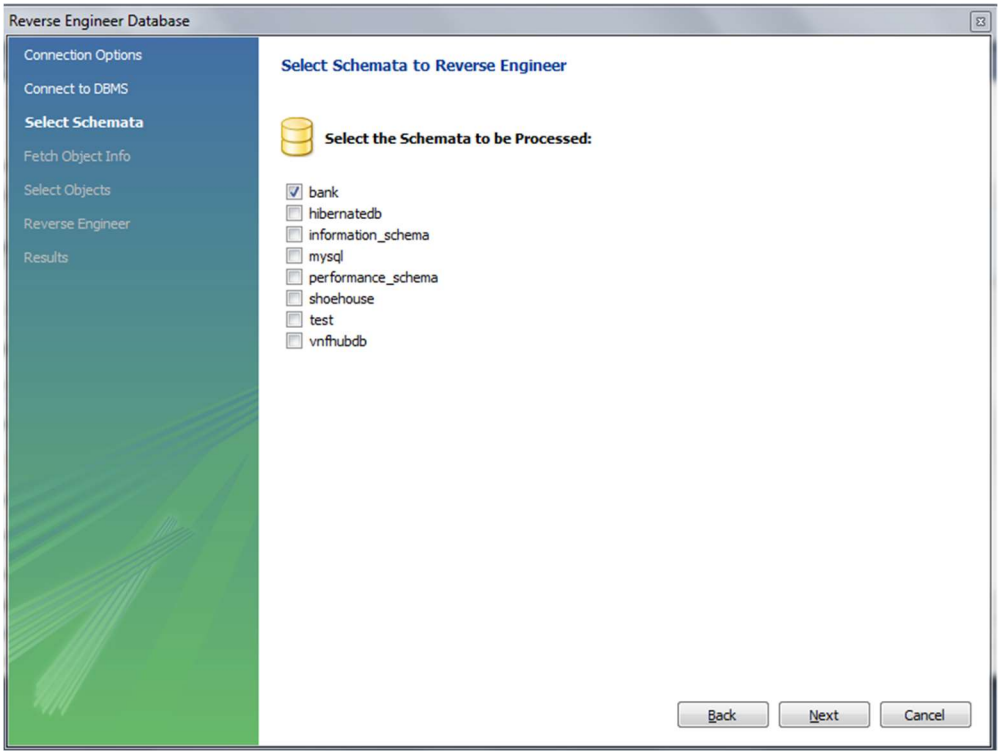
3. Select your **stored connection** *(for connecting to your MySQL Server in which database is present)* from the dropdown. Then click **Next**.



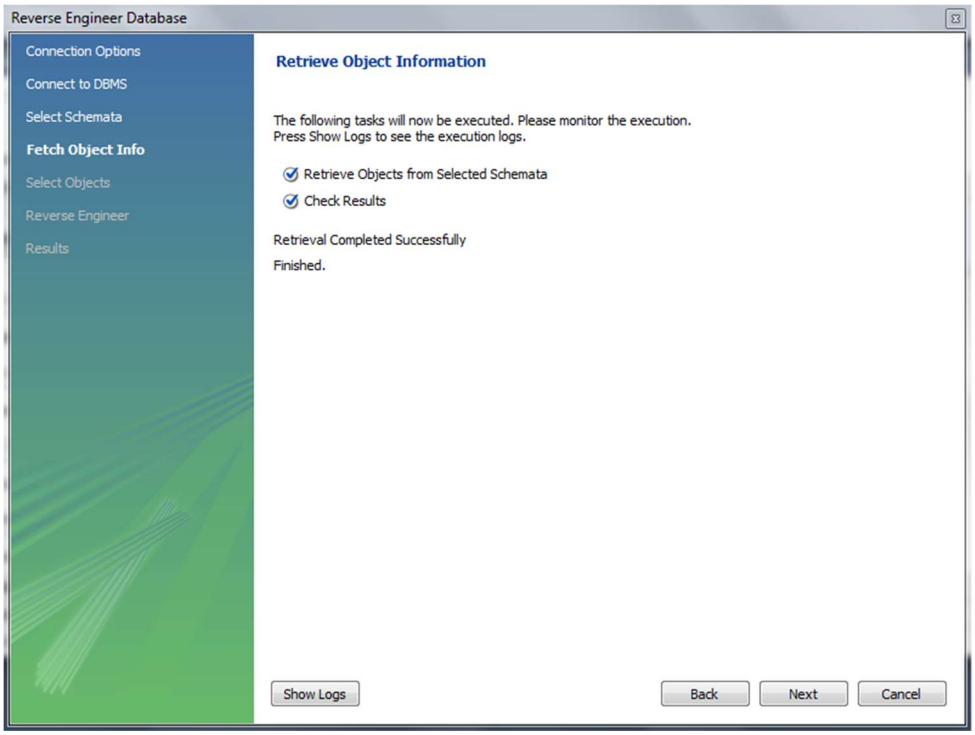
4. After the execution gets completed successfully *(connection to DBMS)*, click **Next.**



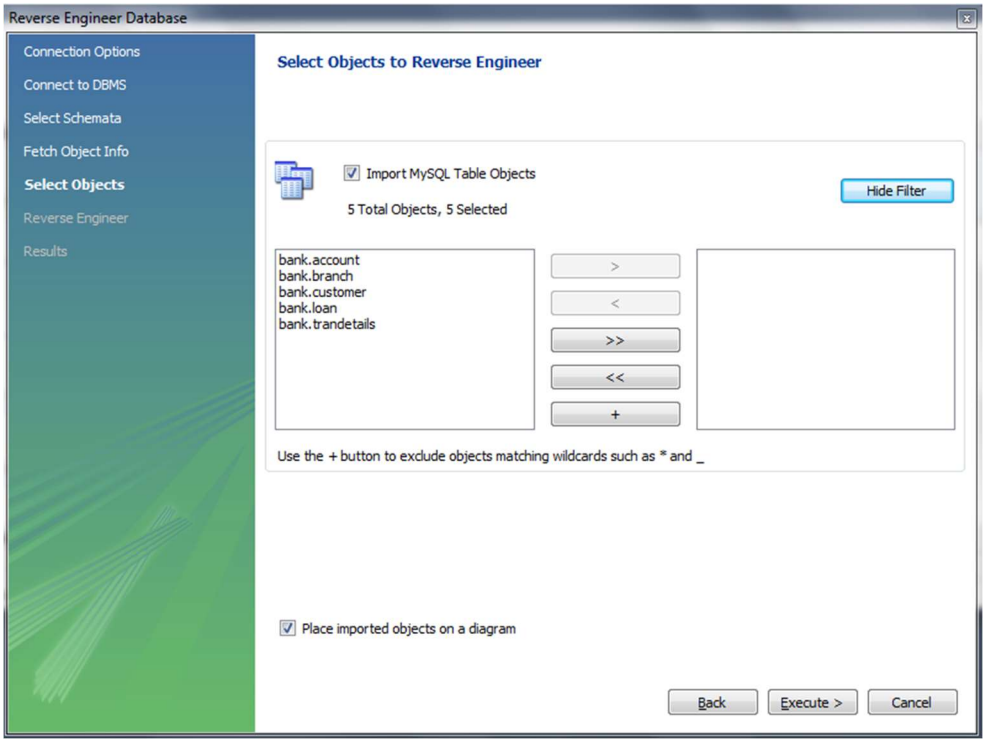
5. Select your Database from the MySQL Server for which you want to create the ER Diagram (*in our case the database name is "bank"),* then click **Next.**



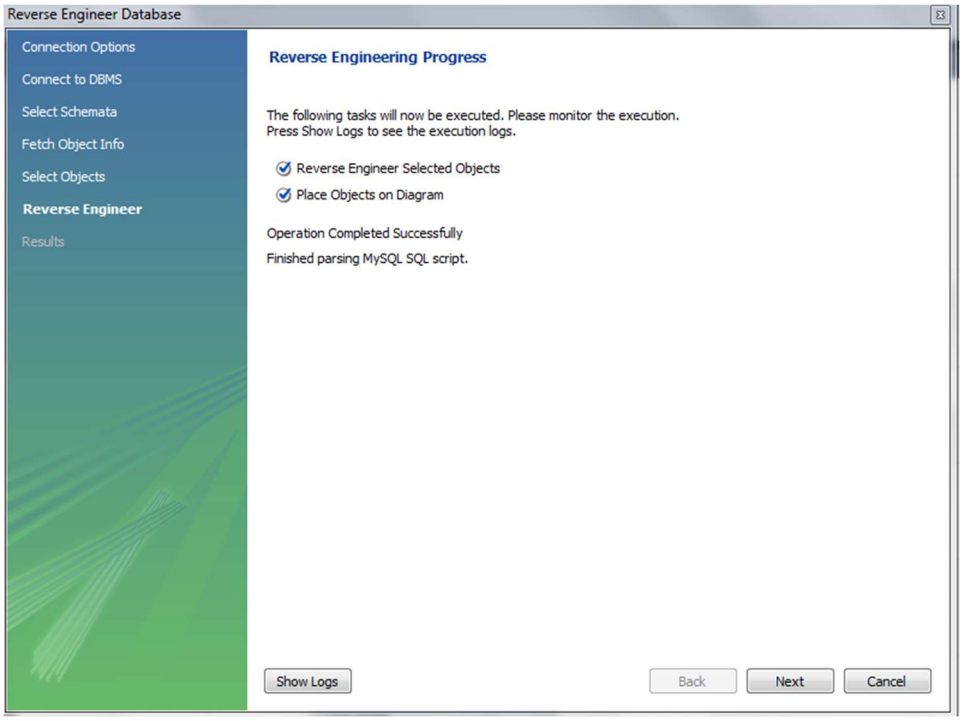
6. After the retrieval gets **completed** successfully for the selected Database, click **Next.**



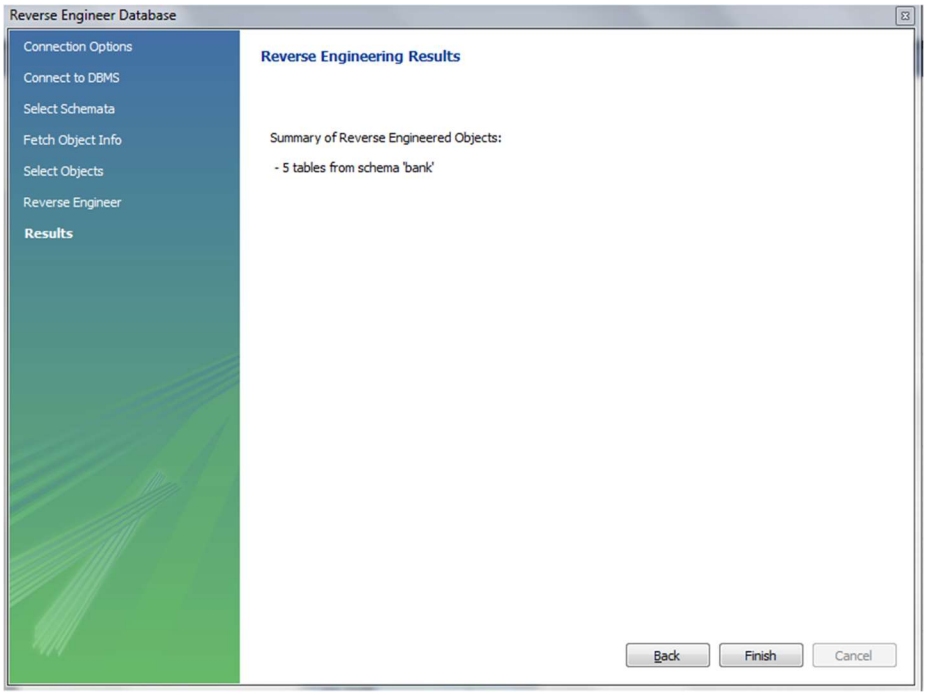
7. Select the Tables of the Database which you want to be visible on the ER Diagram *(In this case I am importing all the tables of the DB),* then click **Execute>.**



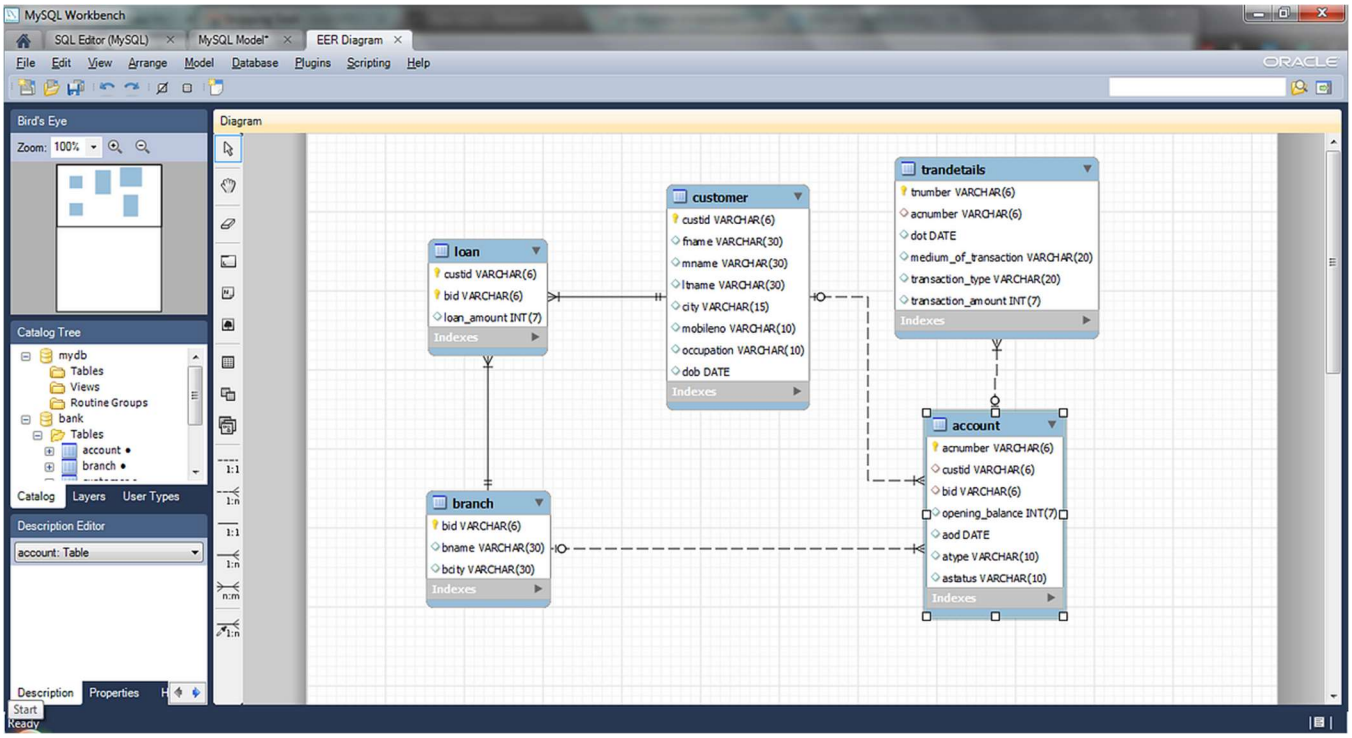
8. After the Reverse Engineering Process gets completed successfully, click **Next.**



9. Click **Finish.**

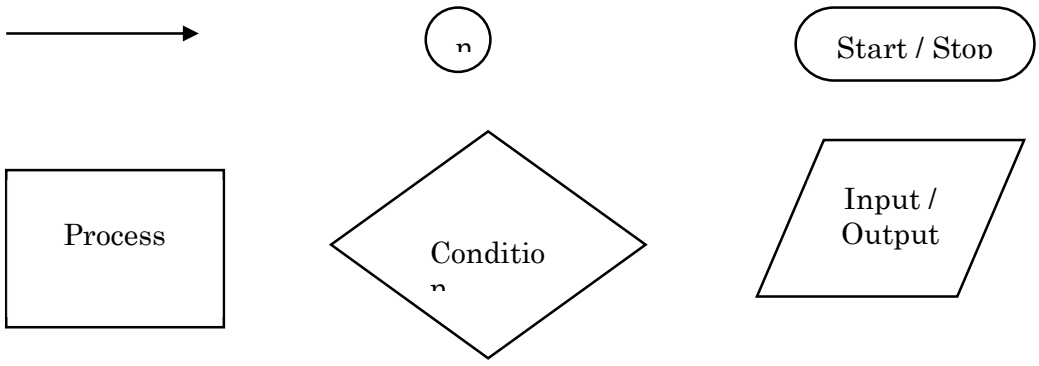


Now you can see the ER Diagram of the Database.

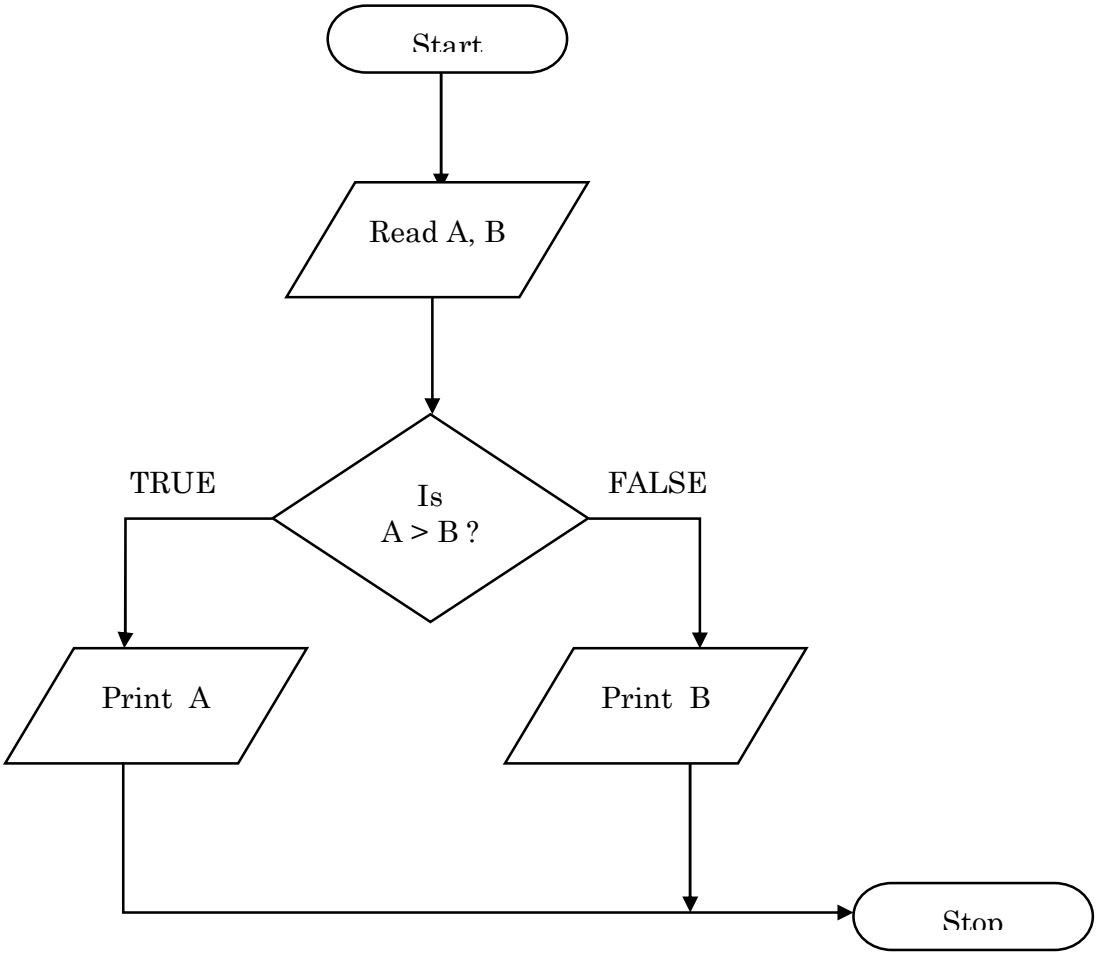


Flow Chart :

We can represent solution of users requirement in terms of flow chart. Following are symbols used in flow chart construct. This diagram shows flow of processing from top to bottom. Arrows are used to indicate flow of processing. Circles are used to connect flow chart from one page to another page they contains number to link connection. Rectangles are used to show process. Diamonds are used to show condition and parallelogram is used to show input and output. Oval is used to start or stop the flow of processing. They are used to show start or stop.



Following is the follow chart find greatest among two values.



Problem : Draw a flow chart to find greatest among given three values.

Algorithm :

Algorithm is representation of solution (logic) of user requirement in the form of English like language. We write each step in concise way step by step giving step number so normally this can be easily understood by everybody.

Following is the algorithm to find greatest among given three values.

Step 1 : Read values of A and B.

Step 2 : Compare A and B. jump to 5 if A is greater.

Step 3 : Print B is greater.

Step 4 : Jump to 6.

Step 5 : Print A is greater.

Step 6 : Stop.

Problem : Write an algorithm to find greatest among given three values.

UNIT-III: INTRODUCTION OF SOFTWARE ENGINEERING AND SOFTWARE DEVELOPMENT PARADIGM

INTRODUCTION TO SOFTWARE ENGINEERING

The technology which cover a process, a set of methods – frame work, and an array of tools that we call software engineering.

Computer software is the product that software engineers design and build by applying a process (software engineering approach) that leads to a high – quality result that meets the needs of the people who will use the software.

Software is both a product and a Tool for producing a product.

- **Products (Software) :**
  It delivers the computing potential embodied by computer hardware or a network of computers that are accessible by local hardware. Whether it resides within a cellular phone or operates inside a mainframe computer, software is an information transformer – producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as a single bit or as complex as a multimedia presentation.

- **Tool for delivering a Product :**
  We develop our product (Software) using another software which includes operating system, languages, packages, database, computer aided software development tools. We also use network and network software to accomplish developing task of new software.

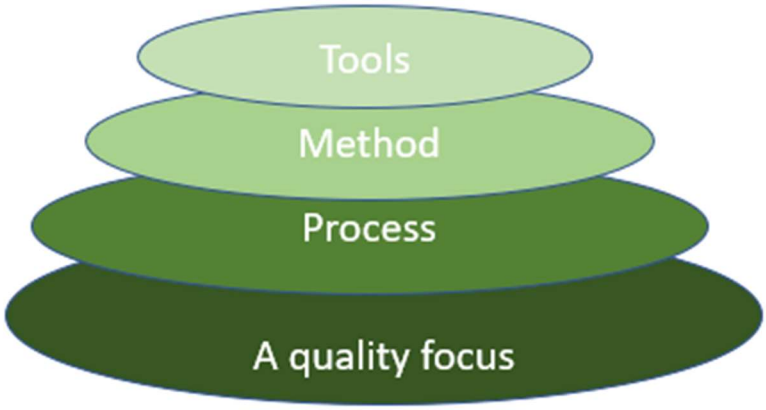
  To develop software Project Management required. Which includes planning, monitoring and controlling of the people, process, and events in the software industry as software evolves from a preliminary concept to an operational implementation.

  Everyone "manages" to some extent but the scope of management activities varies with the person doing it.

Person	Manages
Software Engineer	day to day activities
Project Manager	work of team of software engineers
Senior manager	Coordinates the interface between the business and the software professionals

LAYERED TECHNOLOGY

Software engineering is a fully layered technology, to develop software we need to go from one layer to another. All the layers are connected and each layer demands the fulfillment of the previous layer.



Layered technology is divided into four parts:

1. A quality focus: It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.

2. Process: It is the foundation or base layer of software engineering. It is key that binds all the layers together which enables the development of software before the deadline or on time. Process defines a framework that must be established for the effective delivery of software engineering technology. The software process covers all the activities, actions, and tasks required to be carried out for software development. Process activities are listed below:-

- Communication: It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.
- Planning: It basically means drawing a map for reduced the complication of development.
- Modeling: In this process, a model is created according to the client for better understanding.
- Construction: It includes the coding and testing of the problem.
- Deployment:- It includes the delivery of software to the client for evaluation and feedback.

3. Method: During the process of software development the answers to all "how-to-do" questions are given by method. It has the information of all the tasks which includes communication, requirement analysis, design modeling, program construction, testing, and support.

4. Tools: Software engineering tools provide a self-operating system for processes and methods. Tools are integrated which means information created by one tool can be used by another.

SOFTWARE PROCESS AND SOFTWARE PROCESS MODELS.

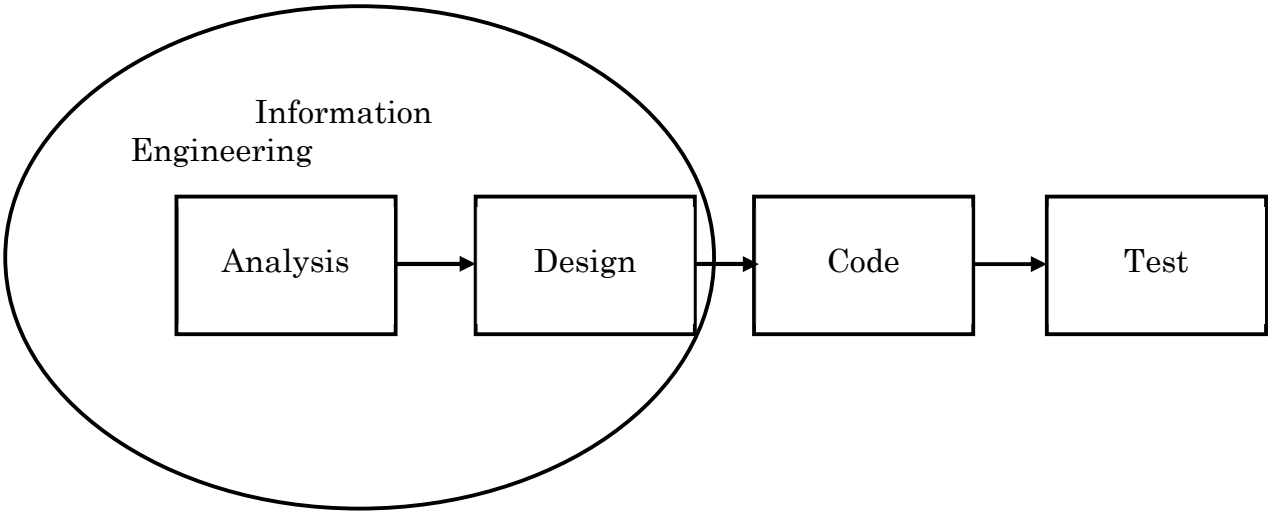
WATERFALL MODEL

Another name of this model is *classic life cycle* or the *waterfall model,* the *linear sequential model.* This proposes a systematic, sequential approach to software development.

This works on system level having phase analysis, design, coding, testing, and support. Figure shows the linear sequential model. The classic life cycle widely used as a procedural model for software engineering. It is considerably better than a chaotic approach to software development.

The linear sequential model encompasses the following tasks.

- **System/information engineering and modeling :**
  Software reflects part of a larger system or business; we start by setting all requirements within software. This is necessary because software must work together with other elements such as hardware, people, and databases. System analysis includes requirements gathering at the system level with Design and analysis. Information engineering includes requirements gathering.



- **Software requirements analysis :**
  The requirements gathering process makes stronger and focused specifically on business activities. Understanding requirement, an analyst can have clear idea about the nature of the software including function, behavior, performance, and interface. Requirements for the system recorded and evaluated with the customer.

  - **Design :**
    Software design shows following four distinct components of a program.

    (1) Database Design

    (2) Software architecture

    (3) Interface Design

(4) Algorithm

The design process converts requirements into a symbolic representation of the software. That can be used for static testing before coding. Like requirements, the design is documented and turns out to be part of the software configuration.

- **Code generation :**
  The design must be converted into a machine program. The code generation step does this task. If design is performed in a correctly, code generation can be done speedy and with more efficiency.
- **Testing :**
  Once code developed, program testing can be started. The testing process covered by static and dynamic way. It also covers structural and functional testing. For quality testing also covers non functional requirements or implicit requirements.
- **Support :**
  To maintain software's validity we must update environmental effect on the software so there is possibility of corrective, adaptive, enhancement or preventive maintenance.

Some drawbacks of this model are as follows.

(1) This model assumes that all requirements stated in the beginning of project. It is sometimes difficult for the customer to give all requirements clearly. Many projects have ambiguity at the beginning so there may be confusion to apply this for such projects.

(2) Changes in this model create confusion as work goes on day by day. Major projects does not go linearly, there must be some kind of cycles. In this project cycles are not shown so it does so indirectly.

(3) Since user can use software after it is fully completed so the customer must maintain patience.

(4) Reviewing in between the project process eliminates errors in early stages which results in low cost. This is not possible in this.

(5) Team member of one stage must wait for completion of work of another to start their own work.

ITERATIVE MODEL

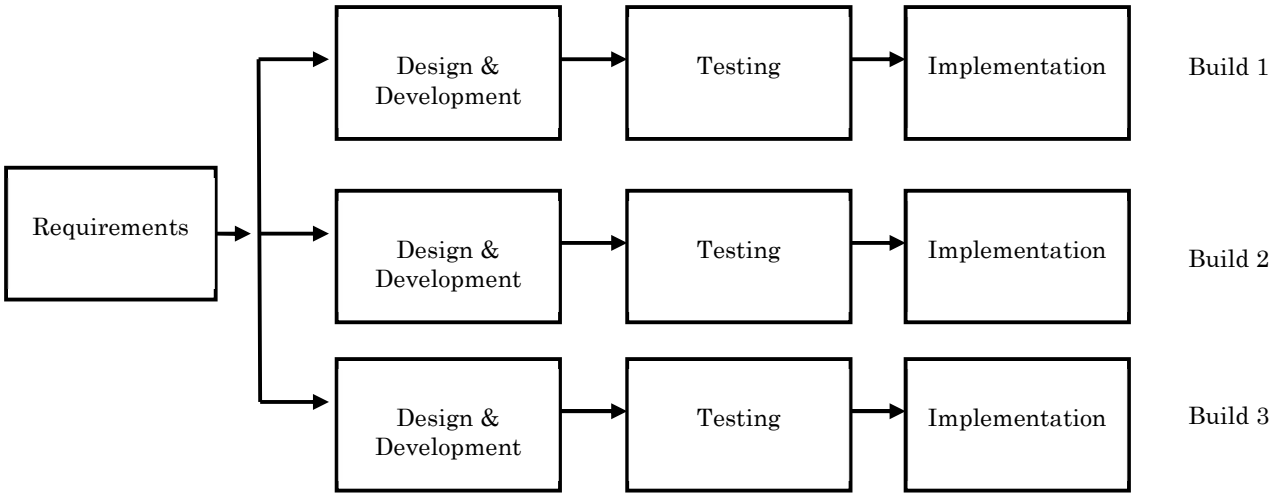
In Iterative model, iterative process **starts with a simple implementation of a small set** of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Iterative process starts with a simple implementation of a part of the software

requirements and repeatedly improves software until the full system is constructed. In each cycle, design changes and new functionalities are added. This model uses divide and conquer method and software developed in parts priority wise.

Following is the pictorial representation of Iterative and Incremental model.



Iteration word focus on the development module goes through the requirements, design, implementation and testing phases.

Incremental progress is the progress of system module by module. If we are constructing different module one by one gradually and adding modules in the system priority wise then it is considered as increments of software. The process continues till the complete system is ready as per the requirement.

Sometimes Iterative and Incremental development combined for system development. Multiple Modules can go parallel or one by one each modules developed during cycle (iteration) depends on model which you are applying.

After each iteration "user feedback" and in the beginning "requirement analysis" is necessary task which validates requirements. As the software constructed through successive iterations, tests have to be repeated to verify each version of the software.

In the development of core part of software if we can solve ambiguity regarding requirement, design, technology then later increment can be easily constructed. As we know this model can handle projects having big size by divide and conquer this model is used for projects having following criteria.

(1) User Requirements clearly defined and prioritized.

(2) After constructing core of the software, newer version can be planned. Planning of increments can be changed as requirement priority changes with time.

(3) Market constraint can be analyzed clearly and step by step.

(4) A new technology adaption becomes easy and manageable even though we are using it first time.

(5) We can start with insufficient resource for full project because we have to work with part of project only and for that we have sufficient resource.

(6) More convenient for Risk and goal handling by its nature of iteration and increments.

- Iterative Model Pros and Cons :

The following table gives the pros and cons of Iterative and Incremental Model.

Pros	Cons
(1) Core version can be used to early start of working in the business. We can start with less resource. (2) Concurrent development can be planned. Project can be controlled more easily. (3) Flexible in requirement change. Early Testing and debugging gives low cost. (4) Easier to manage priority - High Priority part is done first. Every iteration gives new increment which is operational product. (5) It supports changing requirements with customer feedback. (6) Critical Projects can be easily maintained.	(1) Total resource in terms of effort may increase with more management attention. (2) System architecture or design issues may arise. (3) How to define increments require for the complete system. (4) Not suitable for smaller projects. (5) Complex to define increment and manage. (6) Project's progress is highly dependent upon the risk analysis phase.

V-MODEL

The V-model is known as Verification and Validation model. Development processes happens in V-shape.

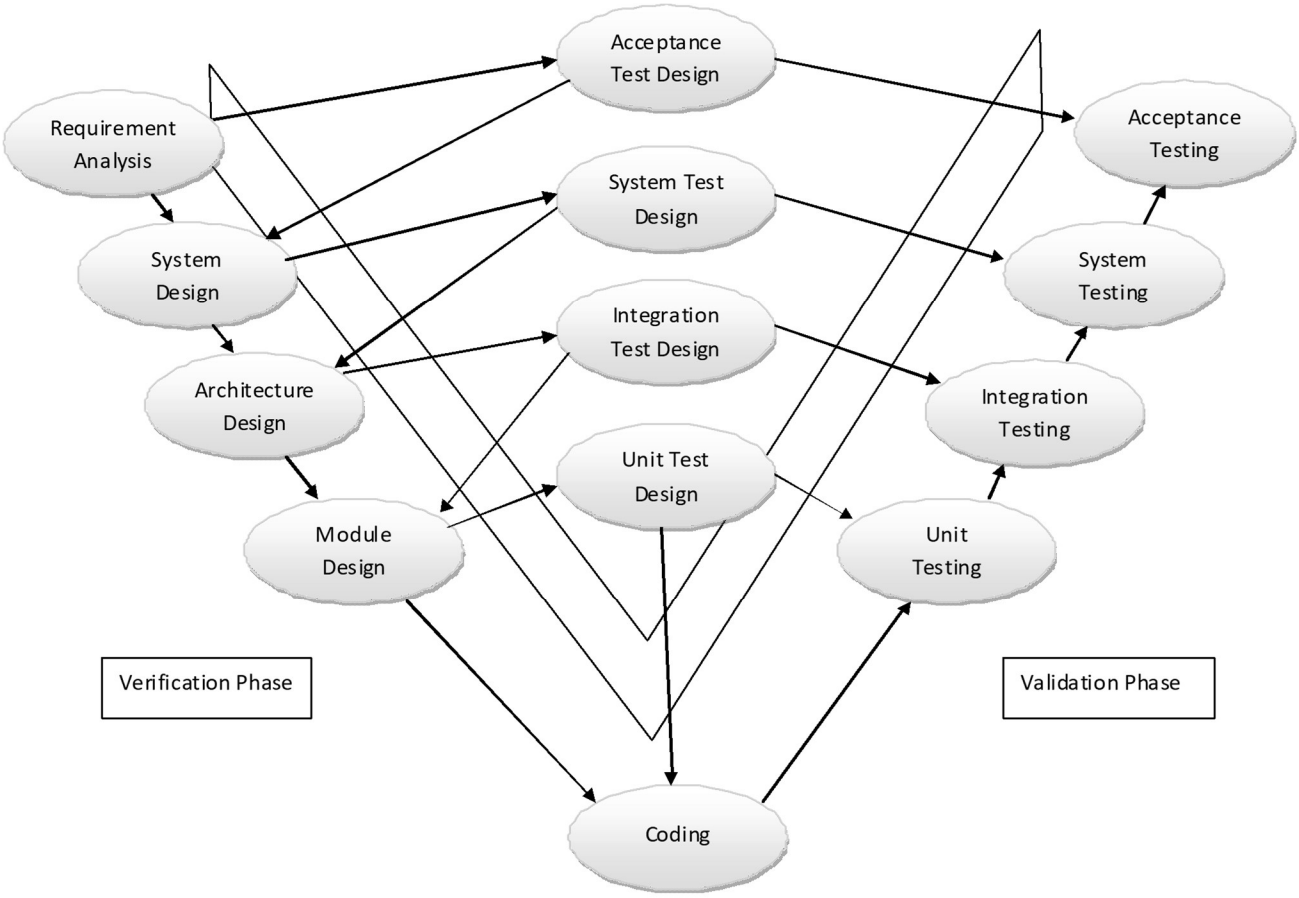
V - Model is an addition in the waterfall model. It correlates testing phase with corresponding development stage. So we can correlate each phase of linear sequential model with testing phases.

- V- Model design :

As we know in smoke testing every milestone or work task can be verified by user and then work can be preceded. Similarly in V-model we add validation in this we have to make plan for testing parallel with development. If we plan for testing while developing software which reduces overall testing time and we can give more attention on testing and effective test case generated.

In V-Model, the corresponding testing phase of the development phase is planned in parallel. So there are Verification phases on one side of the 'V' and Validation phases on the other side. Coding phase joins the two sides of the V-Model.

The below figure illustrates the different phases in V-Model of SDLC.



- Verification Phases :

- **Business Requirement Analysis :** The main task of analyst is problem definition. Problem definition includes fact gathering by analyst by communicating with users at each level of users like manager, officer, operator, worker, etc. Analyst must ask probable solution for this problem. Problem definition should be precise and requirements must be documented in SRS document. Workable solution approved by management from proposed solution plan. Until we define precise user needs we are unable to make design. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
- **System Design :** After defining all requirements and using specification document we can start design. System design includes detailed understanding of hardware, storage, communication, and logic of program (algorithm/flowchart) for the product under development. System test plan is generated which is based on the system design.
- **Architectural Design :** Architectural specifications designed in this phase. Technical approach depends on technical feasibility along with economical feasibility. Management (Steering committee) is responsible for project proposal selection among proposed technical solution. More precise design can be achieved called High Level Design (HLD) by breaking down into modules with different

functionality.

Module interface and user interface clearly and precisely defined. Using this details integration tests can be designed and documented during this stage.

- **Module Design :** Module Design is the process of designing the detailed internal design for all the system modules. It is also called Low Level Design (LLD). Modules must be compatible with the other modules having low coupling and high cohesion in the system architecture. Unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at module stage. Unit tests can be performed structured and functional both the ways.
- Coding Phase : Now design (logical design) would be converted in coding (physical design). The best suitable programming language is decided based on the system and architectural requirements. The coding is must satisfies guidelines and standards. Code should be passed in different types of static testing and optimized for best performance before the final build is tested.

➢ Validation Phases :

Following are the Validation phases in V-Model

- **Unit Testing :** The Unit of system is a Module. Unit tests designed in the module design phase. This is first step of validation. This would be structural and functional testing process. Unit testing is the testing of modules which helps to eliminate bugs at early stage, though communication errors cannot be uncovered by unit testing. Dependency of modules has been covered next.
- **Integration Testing :** Dependency of Modules tested here. Integration testing is performed to test the module interface and communication errors of the internal modules within the system.
- **System Testing :** System testing is directly associated with the System design phase. System tests validate the complete system functionality and the communication of the system under development with user interfaces. Most of the software and hardware compatibility issues can be uncovered during system test execution.
- **Acceptance Testing :** Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non functional issues such as load and performance defects in the actual user environment.

  Applications which are candidates for V- Model are almost same as waterfall model, as both the models are of sequential type. Requirements have to be very clear in information engineering, because it is usually expensive to go back and make changes. This model is used in the government sectors, as there are rule based formal system is applied.

➢ **Characteristics of V – Model.**

- As in waterfall model Requirements are well defined, clearly documented.

- Product definition would be fixed.
- Technology is priory decided using which we are going to develop the system.
- There should be no confusing requirements. All requirements precisely defined.
- The project should be short.

➢ V- Model Pros and Cons :

The advantage of V-Model is that it's very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today's dynamic world, it becomes very expensive to make the change.

The following table lists out the pros and cons of V-Model:

Pros	Cons
(1) Phases are completed one at a time systematically. (2) More suitable for smaller projects where requirements must be well understood. (3) Simple and easy to implement. (4) Easy to manage because we are simultaneously doing testing and validation. (5) Each phase has specific work product and a review process.	(1) Not suitable for complex projects. (2) Not suitable for long and ongoing projects. (3) Not suitable for the projects where requirements are changing. (4) We are finalizing application in each phase so going back is more costly and uncontrollable. (5) We have working software at last.

SPIRAL MODEL

In this model there may be many iterations of prototyping with the controlled and systematic aspects of waterfall model. The spiral model is a series of incremental releases. First iteration develops a paper model or prototype. During later iterations, gradually more complete versions of system are produced. A spiral model is separated into a number of framework activities, also called *task regions.*

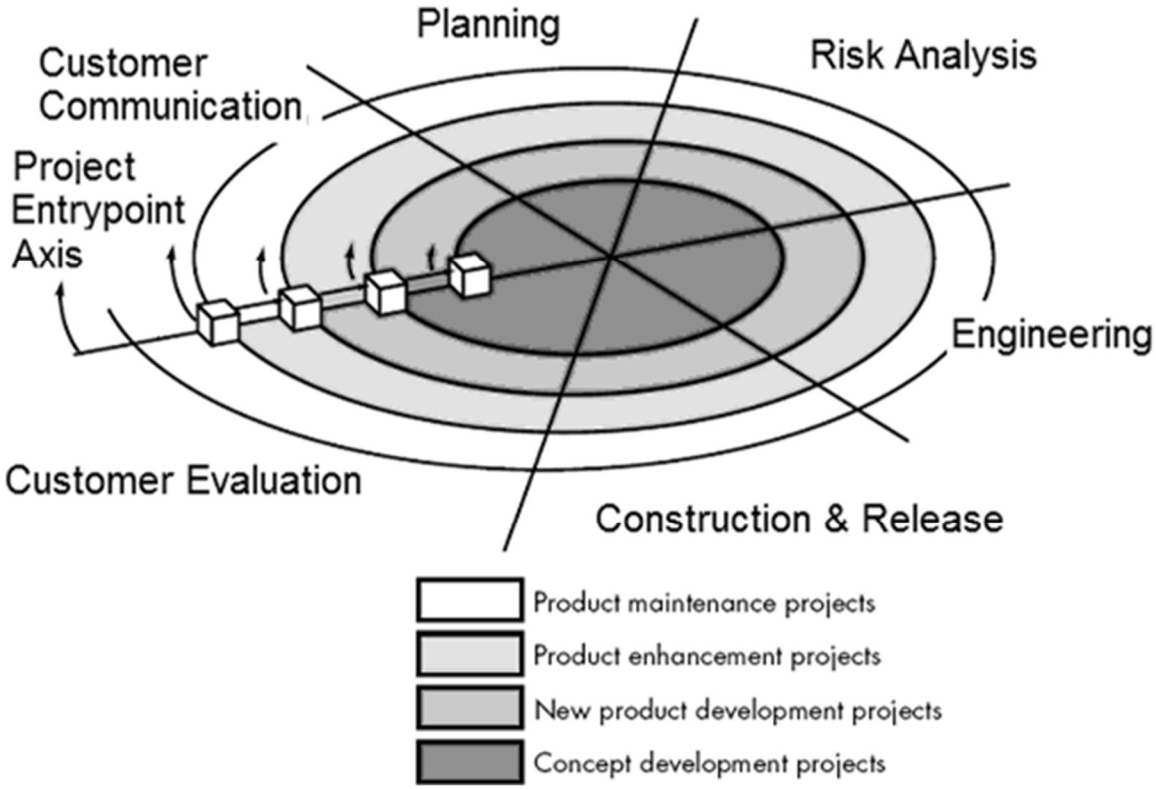
Customer communication — necessary to found useful communication between developer and customer.

Planning — necessary to identify resources, timelines, and other information.

Risk analysis—necessary to review both technical and management risks.

Engineering—necessary to make representations of the application.

Construction and release—necessary to create, test, install, and documentation and training.



Customer evaluation necessary to get customer feedback based on evaluation of the software representations created during the engineering stage and implemented during the installation stage.

(1) For small projects, the number of work tasks and their procedure is low. For larger, more critical projects, each task region contains more work tasks that are defined to achieve a higher level of formality.

(2) As this process begins, the software engineering team moves around the spiral in a clockwise direction, beginning at the center. Each iteration starts at project entry point axes. Each cube placed along the axis can be used to represent the starting point for different types of projects

(3) The first cycle results in the development of a product specification.

(4) Next cycle of the spiral used to develop a prototype.

(5) Next each cycle creates enhancement on the product.

(6) Cost and schedule are adjusted based on feedback derived from customer evaluation.

(7) Number of iteration depends on planning made by project manager.

(8) The spiral model can be modified to apply throughout the life of the computer software.

(9) A "concept development project" starts at the core of the spiral and will

(10) The spiral model is a realistic approach for construction of large-scale systems and software.

(11) The spiral model uses prototyping as a risk cut mechanism.

(12) It follows the classic life cycle but fit in it into an iterative framework that more practically reflects the real world.

(13) The spiral model deals with a direct consideration of technical risks at all stages of the project. It reduces risks before they become problematic.

(14) Sometimes it is difficult to convince customers in contract situations that this approach is controllable. It is necessary uncover risk on every step otherwise it may give problems. This is not widely used because of this reason.

BIG BANG MODEL

This model is suitable when project size is very small and individual or two three person working on project. When requirements are not clear at beginning we can work according to this model. It does not follow any structured way. Here delivery date is less important. Planning can be done depending requirements but here requirements are not clear so planning cannot be done. Requirements gathered by customer communication. If customer is irresponsive then it is difficult to ensure requirements on early stage. Sometimes customer is not sure about what exactly he wants.

Big bang model uses all the possible resources in software development and coding, with very little or no planning. The requirements are implemented as they come. Any changes required we try to adapt these by less disturbing with complete software. For students this can be one of the approaches.

- Big Bang Model Pros and Cons :

Big Bang model has very high risk of technical or managerial type. In lack of planning time may be saved on early stage but later on rework may be there. Here we don't follow formal procedure so management can be done on programmer's conveniences.

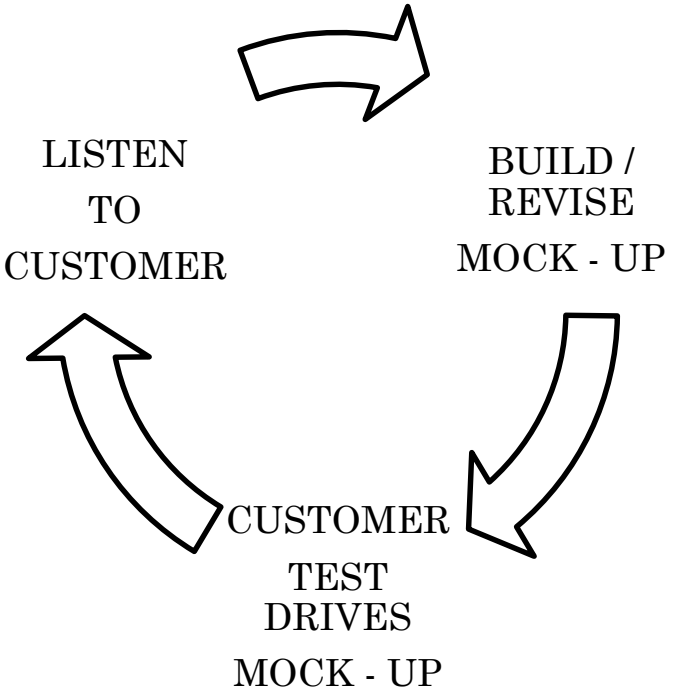
It is ideal for iterative or small size projects required less effort.

Pros	Cons
(1) Simplicity is one of the features. (2) Planning time saved. (3) No formal method to manage. (4) Early beginning may require less resource. (5) Developers work freely. (6) For students this can be one of the approaches.	(1) It has high technical or managerial risk. (2) Uncertainty is one of the serious problems. (3) Not applicable for complex projects. (4) Poor model for long projects. (5) Development cost may be raised unexpectedly.

PROTOTYPING MODEL

Prototype is small working product. Here customer gives general objectives for software but not able to identify detailed input, processing, or output necessities. The developer may need to implement system on trial basis to check logic, the adaptability of environment, or the user interfaces so that developer may choose a prototyping model.

Listen to customer is the phase of requirement definition. Developer and customer decide overall objectives and identify ambiguous requirements. We try to develop software for user's confirmation as possible as early. This quick design emphasis user's satisfaction and good user interface by giving good input approaches and output formats. User can start their work on priority basis if requirement identified so.



The prototype is used by the customer or user. Now user can have better idea about the next prototype with refined requirements.

This gives a chance to think and decide software requirements. Generally the first system built is hardly usable and may have some problem of speed, size. It may be awkward in use due to poor design of interface. We can start with scratch.

When we are testing new technology then this would be helpful. The prototype can serve as "the first system."

Generally customers and developers like the prototyping paradigm because users get a feel for the actual system and developer's starts implementation immediately. But there are some drawbacks of these.

(1) Overall software quality sometimes not considered seriously. To achieve after some iteration it is too complex. We have to rebuild or rework on the project.

(2) Some kind of win-win situation ignored. Developer can postpone exact implementation of requirement. There may be compromise in programming language, technology, algorithm quality and later he forgets to eliminate.

(3) The customer and developer must both agree that the prototype is built to serve as mechanism for defining requirements. Overall quality will maintained with guarantee in later iterations.

UNIT-IV: PROJECT MANAGEMENT WITH CAM TOOLS& CAPABILITY MATURITY MODEL

CONCEPTS OF PROJECT MANAGEMENT

All projects have following common attributes.

- An overall goal
- A project manager
- Individual tasks to be performed
- Timing for those tasks to be completed (such as three hours, three days, or three months)
- Timing relationships between those tasks (For example, you can't put a new manufacturing process in place until you train people in how to use the process.)
- Resources (people, equipment, facilities, and supplies, for example) to accomplish the work .
- A budget (the costs associated with those people, equipment, facilities, and supplies).

A project manager isn't always the highest authority in a project. Senior Management may impose limitation on project and manager has to work by satisfying these constraints. Rather, the project manager is the person on the front lines who makes sure that the parts of the project come together and assumes hands-on responsibility for successes as well as failures.

A project manager manages these essential aspects of a project.

- **The project plan or schedule :** This is what we create with any project. It includes the approximated steps and associated with timing and costs concerned in reaching the project goal.
- **Resources :** Allocating resources for particular activity may introduce resource conflicts and building agreement as well as assigning resources and monitoring and reevaluating their activities on the project. Resource may be man power, budget, equipment or any other material.
- **Communication with people (programmer, customer, manager) :** Getting customer feedback after every phase can avoid major blunder after implementation. So customer and developer interaction must be necessary to resolve any kind of ambiguity. To overcome any kind of problem detected earlier developer may ask to manger for resources and other support.

  As we know project management accomplished by planning, monitoring and controlling. If we want correct plan then following areas must be estimated properly because a true estimation can give idea of size, complexity and future risk. Project failure may occur due to unrealistic dead line or schedule overrun, budget overrun and there are many reasons that affects on project. It is not good sign to adapt strategy of fire-fighting mode (reactive method) but we must prepare in advance by predicting future risk (proactive method), requirement and prepare solutions in

terms of actions needed by project manager. If we work using quantitative data then accurate estimates can be derived for controlling the project.

- **Project complexity** : To measure complexity we uses function point (FP) Which is used to determine project size with effect of complexity using weighting factor. Complexity effects development progress. While designing or coding we face actual complexity which is difficult to determine earlier. To handle complexity we required good experience of past projects. Experienced programmer would not disturb by complexity while beginner found troubles to deal with it. To increase capability of a company to handle complex project, CMM – PM model cab be implemented. In this model major focus given to personnel of company starting from recruitment, incentives, training, enhancing their skills. Ultimately company can grow using progress of their employees and undertake more complex projects. There are number of quantitative software complexity measures have been proposed, such measures are applied at the design or code level and are therefore difficult to use during software planning before a design and code exist.
- **Project size :** Accuracy and usefulness of estimates also depends on size of project. Project size is also one of the risk factor. As size increases, the interdependency among various elements of the software grows rapidly. Resources should be sufficient for bigger project. We can apply divide and conquer strategy to deal with that. So problem decomposition is an important activity for estimating. Overall project growth depends on that how we manage interdependency between decomposed elements.
- **The degree of structural uncertainty :** Structure refers to the degree to which requirements have been set, the simplicity with which functions can be designed with low coupling and high cohesion. Good hierarchical nature of the information must be maintained. We can implement object oriented design to get good quality of software.

➢ **Project costing based on metrics :**

Software is the most expensive element of virtually all computer-based systems. For complex projects, cost estimation error can make significant change from profit to loss. Cost overrun is also related with schedule overrun. Cost affected by another factors like human, technical, environmental, political. To achieve perfect cost and effort estimates following point must be considered.

(1) Use past project estimates.

(2) Decompose process and project as simply as possible.

(3) Use empirical models to estimate.

(4) To perfectly estimate we must wait till all related information gathered.

If we wait till we know completely, there is less probability to make serious errors in estimation. Past experience has not always been a good indicator of future results but normally previous metrics becomes basis for future project.

Estimation must be cross-checked for the other. Empirical estimation models can be used to balance decomposition techniques and suggest matured estimation approach.

A model is based on experience & historical data and takes the form E = f(p) where d is one of a number of estimated values like effort, cost, project duration and p are selected independent parameters like estimated LOC or FP.

PROBLEM-BASED ESTIMATION

Using historical data, the planner can start estimating necessary, required, and sufficient size value for each function or count for each information domain value. An implicit indication of the degree of uncertainty is provided when a range of values is specified. Deviating estimates can often occur of two causes.

(1) Ambiguity to understand the scope.

(2) Historical data used for estimation is inappropriate for the application or misapplied.

We have two kind of measures in real world.

[1] Direct measures : Direct measures include cost and effort applied which can be directly measured. Direct measures of the product include lines of code (LOC) produced, execution time, memory size, and defects reported in a time span.

[2] Indirect measures : This kind of measures depends on other direct or indirect measures. Examples of this types is robustness, quality, complexity, efficiency, reliability, maintainability, and all other quality factors

How a planner can use metrics from different projects for estimation of current projects?

Two different project teams found and categorize all errors during the software process. Individual measures are then combined to develop team measures.

Team A found 200 errors during the software process prior to release. Team B found 184 errors. All other things being equal, which team is more effective in uncovering errors throughout the process? We must know other details of product and team size. Due to incomplete information we cannot give answer of this question.

- **Size-Oriented Metrics :**
  Size-oriented software metrics are derived by regulating quality and productivity measures by considering the size. If a software organization maintains simple records, a table of size-oriented measures as shown figure.

  The table lists name of past projected developed in a company. As per table project A, 24,200 lines of code were developed with 48 person-months of effort at a cost of $336,000.

  Effort and cost noted in the table stand for all software engineering activities like analysis, design, code, and test. Table also shows that 730 pages of documentation were developed, 268 errors were corrected before the software was released, and 58 defects were faced after release to the customer within the six years of operation.

Project	LOC	Effort	$(000)	pp. doc.	Errors	Defects	People
A	24200	48	336	730	268	58	6
B	54400	124	880	1448	642	172	10
C	40400	86	628	2100	512	128	12
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

Six people worked on the development of software for project alpha. All measures are normalized by line of code and we can find of simple size-oriented metrics for each project given as follows.

- Errors per KLOC (thousand lines of code).
- Defects4 per KLOC.
- $ per LOC.
- Page of documentation per KLOC
- Errors per person-month.
- LOC per person-month.
- $ per page of documentation.

Metrics derived by LOC sometimes difficult to use in following cases.
(1) programming languages are different

(2) Skills and experience of teams are different

(3) Same output can be obtained by different design having different code.

LOC can be used to use metrics of similar kind of projects to use in same kind of project.

- **Function-Oriented Metrics :**

Function-oriented software metrics use a measure of the functionality. Functionality cannot be measured directly. It must be derived indirectly using other direct measures. A measure Function points are derived using an relationship based on direct measures and evaluation of software complexity. To find function point we must complete following table.

Characteristics are determined and counts are filled in the appropriate table location. Measurement parameters are defined in the following manner.

Number of user inputs. Inputs should be distinguished from inquiries. Each user input provided to the software is counted.

Number of user outputs. Each user output that provides information to the user is counted. It refers to reports, screens, error messages, etc

Measurement Parameter	Count	Weighting Factor					
			Sample	Average	Complex		
Number of user inputs		×	3	4	6	=	
Number of user outputs		×	4	5	7	=	
Number of user inquiries		×	3	4	6	=	
Number of files		×	7	10	15	=	
Number of external interfaces		×	5	7	10	=	
COUNT TOTAL →							

Number of user inquiries. They are immediate software response preprocessed to display online (on-line output). Each distinct inquiry is counted.

Number of files. In a database there may be different tables, so each table as a part of database counted. There may be different types of files they are also be counted.

Number of external interfaces. Interfaces are data files on storage media that are used to transmit information to another system are counted.

All these parameter may be categorized in simple, average, or complex type. Weighting factor is assigned and multiplied with count and then totaled.

To compute function points (FP), the following relationship is used.

FP = count total – [0.65 + 0.01 – Σ(Fi)]

Where, count total is the sum of all FP entries obtained.

The Fi (i = 1 to 14) are complexity adjustment values depends on following questions.

(1) Does the system require reliable backup and recovery?

(2) Are data communications required?

(3) Are there distributed processing functions?

(4) Is performance critical?

(5) Will the system run in an existing, heavily utilized operational environment?

(6) Does the system require on-line data entry?

(7) Does the on-line data entry require the input transaction to be built over multiple screens or operations?

(8) Are the master files updated on-line?

(9) Are the inputs, outputs, files, or inquiries complex?

(10) Is the internal processing complex?

(11) Is the code designed to be reusable?

(12) Are conversion and installation included in the design?

(13) Is the system designed for multiple installations in different organizations?

(14) Is the application designed to facilitate change and ease of use by the user?

Questions are answered in 0 to 5. 0 for not required and 5 for extremely required. After function points have been calculated, we can normalize following attribute.

- Errors per FP.
- Defects per FP.
- $ per FP.
- Pages of documentation per FP.
- FP per person-month.

The planner must determine the cause of divergence and then reconcile the estimates.

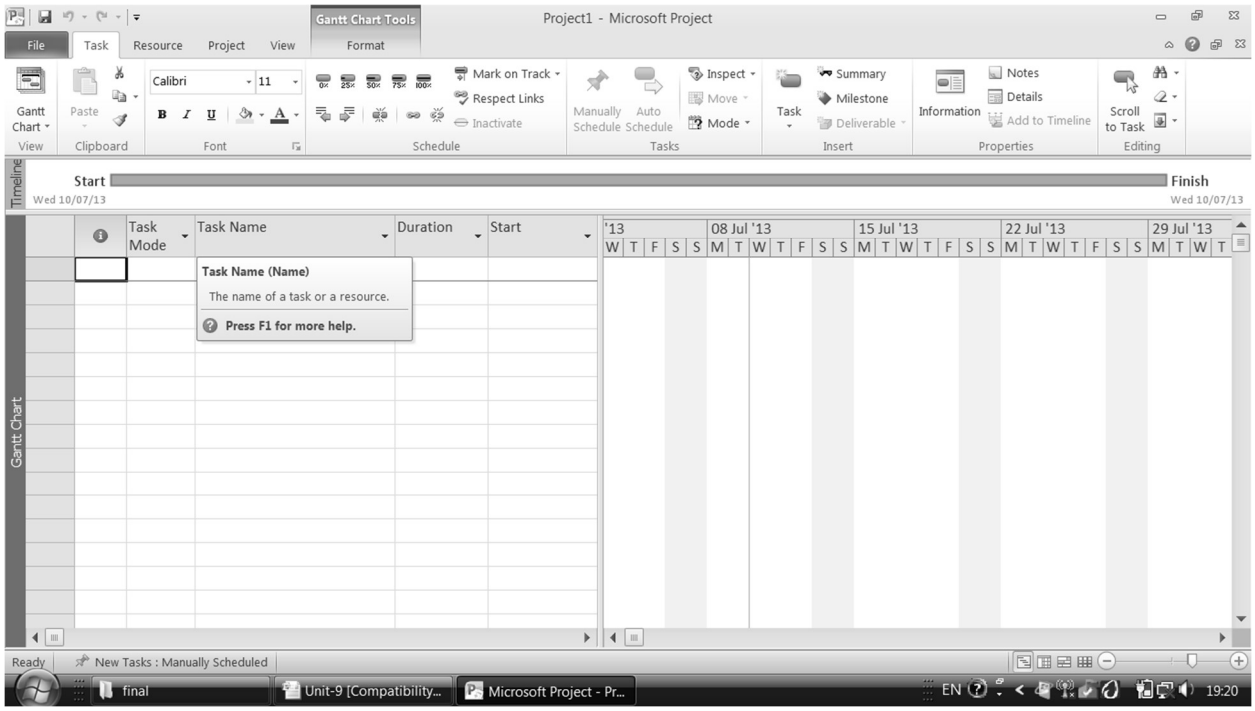
MS-Project for controlling and Project Management

Project management is the management of an organized set of activities directed toward a common goal, using specialized management structures and techniques. Project can be started by estimating manpower, budgets, and resources, Reporting Progress, Evaluating efficiency and effectiveness. In this context Ms Project help us to develop a better plan , makes calculated projections easy and more reliable, detect inconsistencies and problems in the plan, track progress and detect potential difficulties, communicate the plan to others.

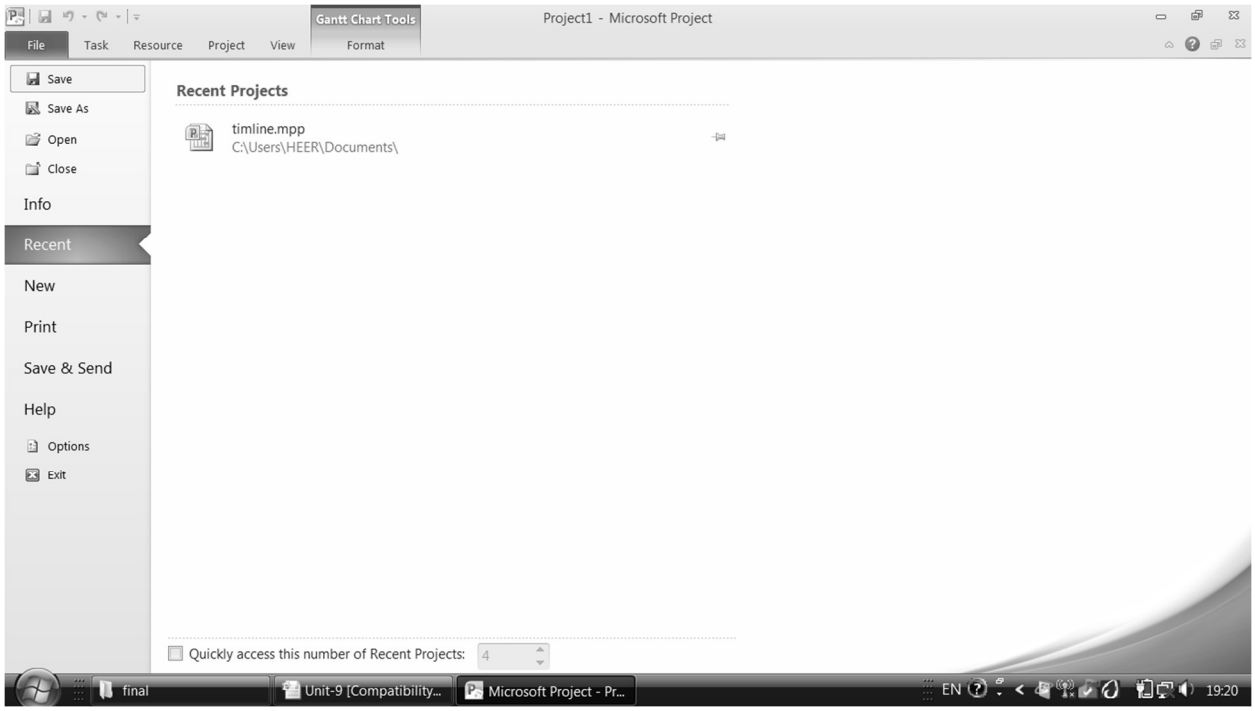
First of all we have to make plan for the project and the person who makes plan is called planner. He starts by defining a project, define the goal of the project, define project deliverables, plan project activities, define phases and create a task list, structure the tasks into their respective phases as well as a hierarchy of summary tasks and subtasks. To start planning we estimate task durations and determine task dependencies and constraints. We must estimate and plan for resources and their needs.

We must input resource information and set working times, Cost of Resources, assign resources to tasks. To track the progress we check, starting and completion time of tasks, budget of project, and any problematic task which resists progress of project. By default we get following screen after opening Ms Project.

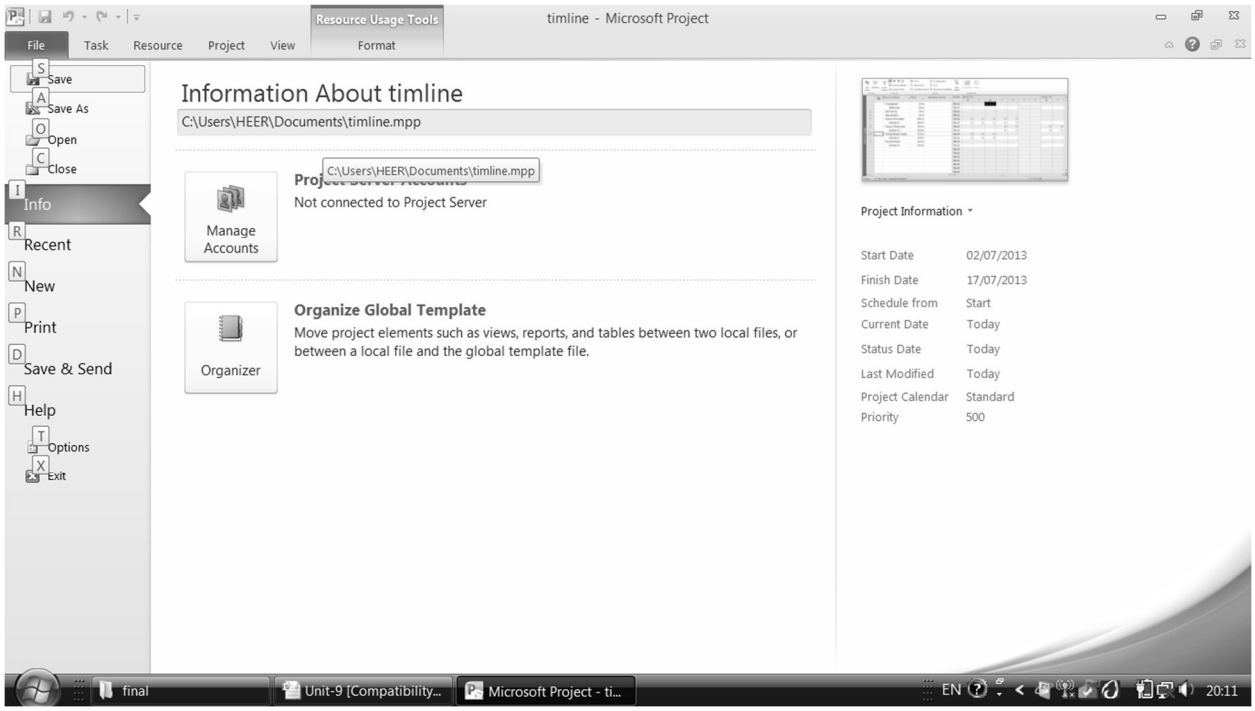
- **Default View of the MS-Project :**
  The default View of the Project is Gantt chart View where you are using Entry table with the column heading Task Name, Duration etc. If all columns are not displayed then click and drag the split bar.



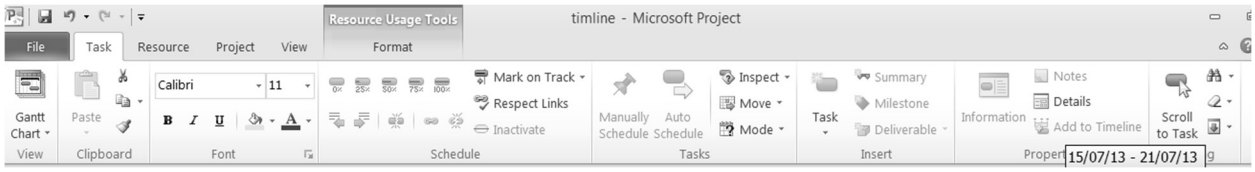
We can open recent project using Recent Option.



To create or open the file select File Menu – Click on New. It gives us option to select Blank project. To select from existing project we have options of Templates, Search in the box, On Office Online or on my computer.



In the left of Task Ribbon we have View Option which contains following types of view.



Some of the Views in MS Project are as follows.

- Gantt chart view
- Calendar view
- Network diagram view
- Task usage view
- Resource sheet view
- Resource usage view
- Resource graph view

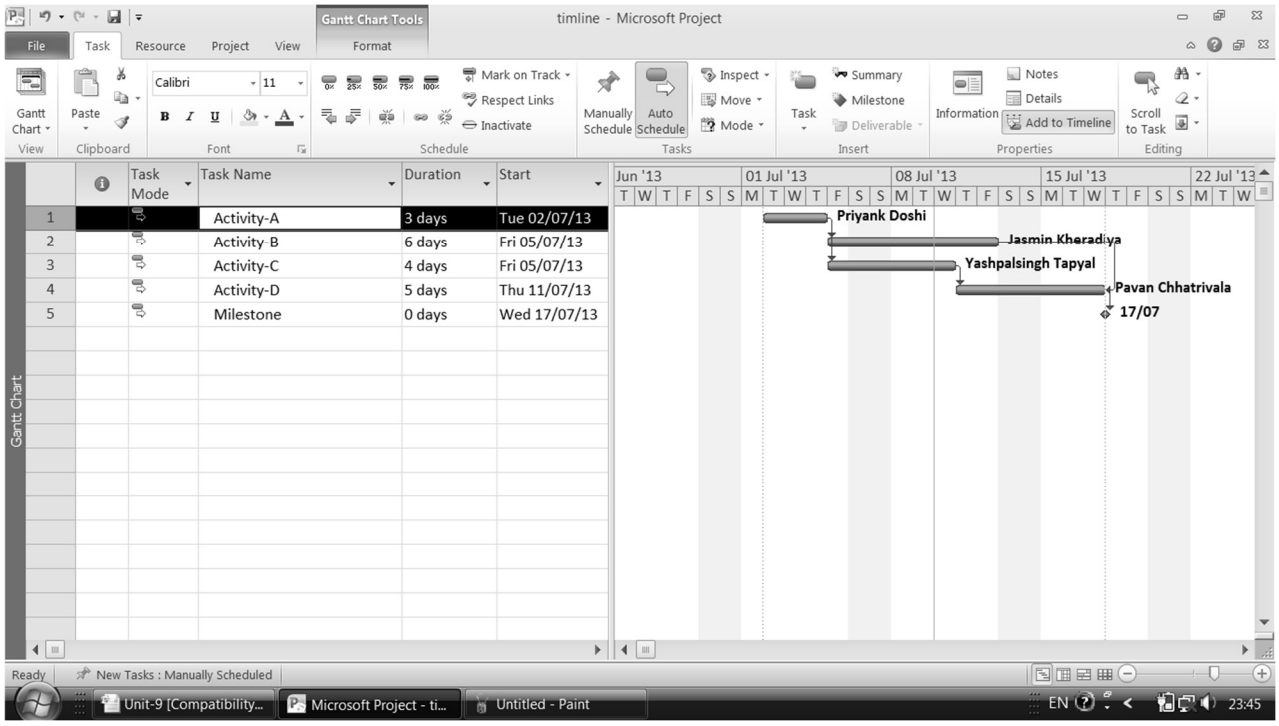
To start working with Ms. Project enter Task or Activity details like name, mode, duration, start date etc, in the Gantt Chart Table using Gantt Chart Tools in the left side. Simultaneously we can view Gantt Chart in the right side.

- **Create a Project Plan :**

- Click on File New- Blank Project
- Enter Project information
- Click on Project Menu- Project Information

➢ **Steps to enter Project Information :**

- Select "Project start date" from schedule from dropdown list
- Enter Start date of the project in Start date field
- Set the priority for the project
- Change Working Time
- Change the default days and time for current project file.



Fields in Gantt chart table :

- Types Of tasks
- Summary Task
- Sub Task
- Milestone

➢ **Entering Task and Duration :**

In the Task Name field, type the name of your first task (or use the entry bar).

Press ENTER, or click the green check mark on the entry bar.

An estimated duration of one-day (1day?) will be displayed automatically when you enter each task.

Duration is entered in months, weeks, days, hours or minutes.

E.g. 1 mo, 2 w, 3 d, 4 h, 45 m

D- Day, Mo-month, w - Week, h-hour, m-Minutes

Note : Let project enter the start date and end date for your project

- **Indenting (Outlining) tasks :**

  Outline Level. Tasks that are sub-tasks (or child tasks) to summary tasks (or parent tasks) are limited in how they are scheduled. A summary task gets its start and finish date from the earliest start and the latest finish of its sub-tasks. If a sub-task's start is held in place by a scheduling factor then so is the summary task. Or vise versa, if the summary task cannot reschedule then the sub-tasks will be affected.

  Select the sub tasks you want to indent. Click on demote button on the Standard toolbar.

  After performing this operation you can observe that the task above the indented tasks appears to be bold.

  - Linking tasks :
    Select the tasks you want to link. E.g. task1 and task2 are to be linked. Select the 2 tasks.

    Click on link button on the Standard toolbar
  - Unlinking tasks :
    Select the tasks you want to unlink.

    Click on unlink button on the Standard toolbar

    Always indent the tasks before linking. Do not link summary task to subtask or another summary task.
  - Task scheduling :
    Predecessor is a task whose completion (wholly or partly) determines the start of the successor. The driving task in the dependency is called a predecessor

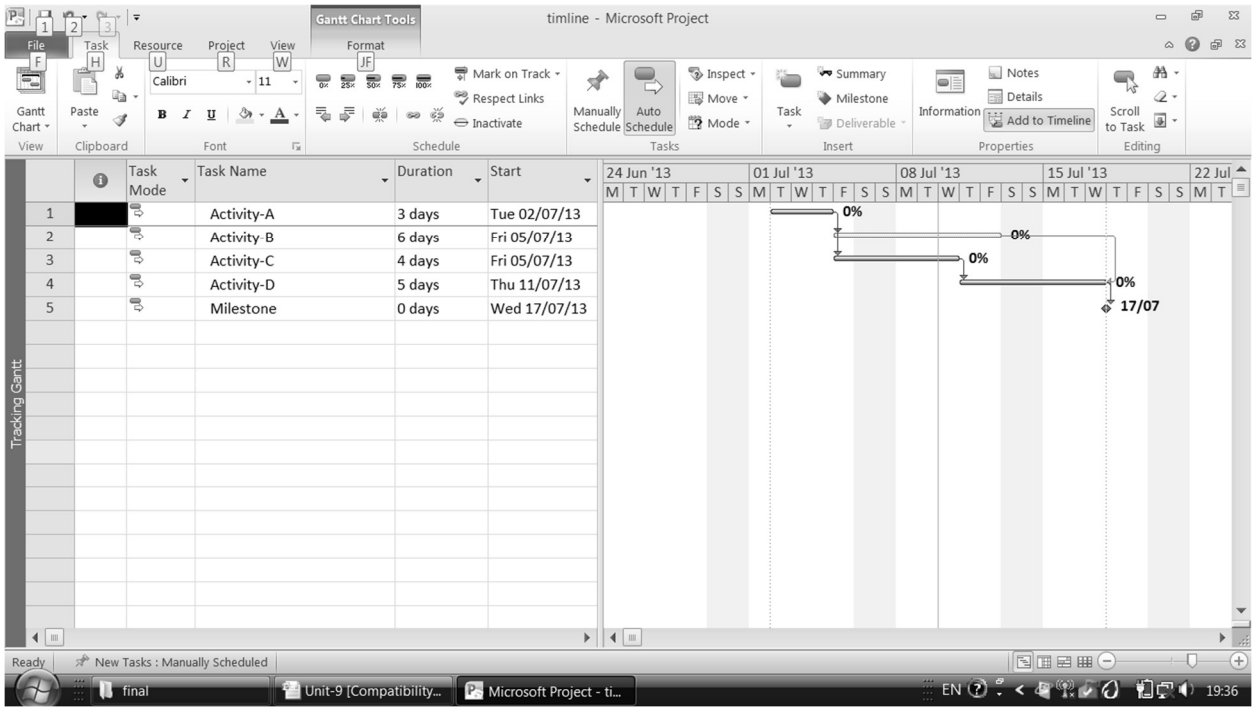
    Successor is a task which is dependent on the completion (wholly or partly) of the predecessor. The dependant task is called the successor.

    There are 4 types of relationship (links) or dependencies between Predecessor and successor. They are as follows:
  - Task Dependencies are as follows :
    Finish to Start [FS] : The task cannot be started until predecessor task is completed.

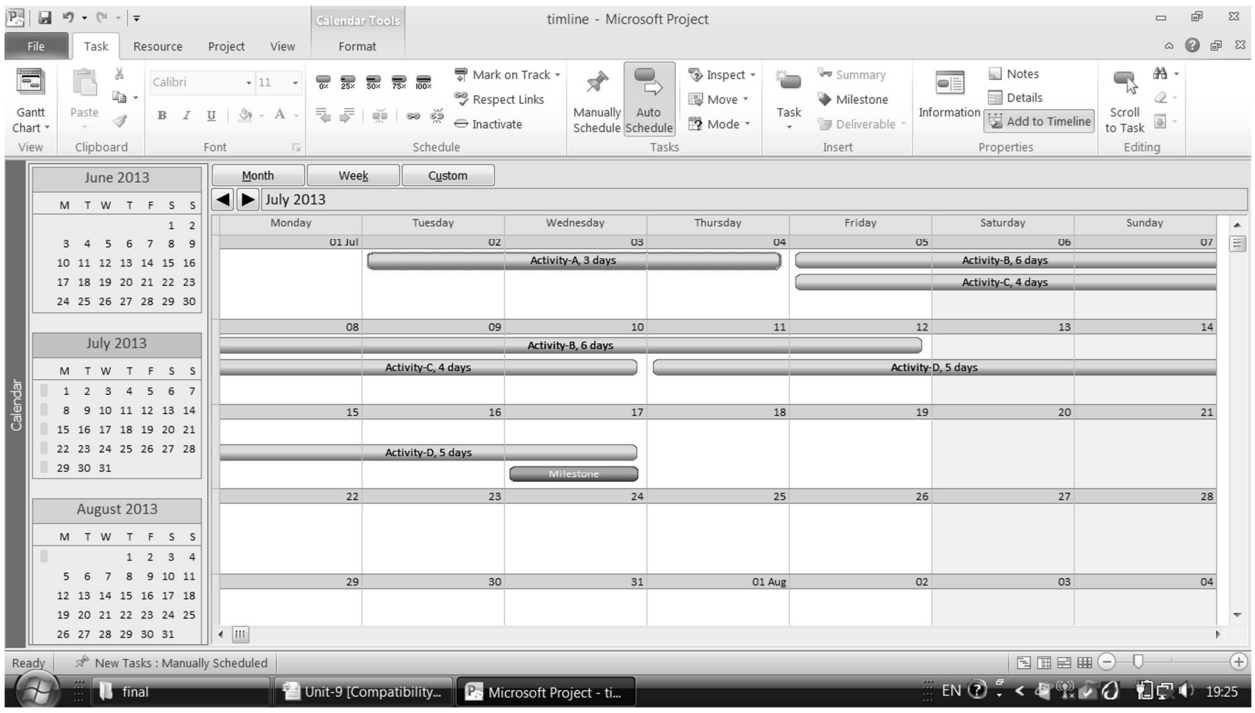
    Finish to Finish [FF] : The task cannot be finished until predecessor task is finished.

    Start to Start [SS] : The task cannot be started until the predecessor task is started.

    Start to Finish [SF] : The task cannot be finished until the predecessor task begins.

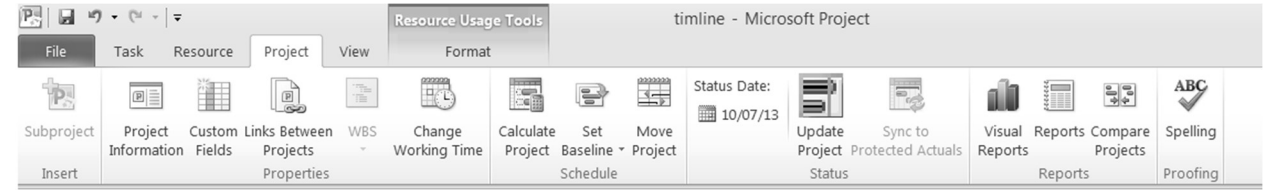


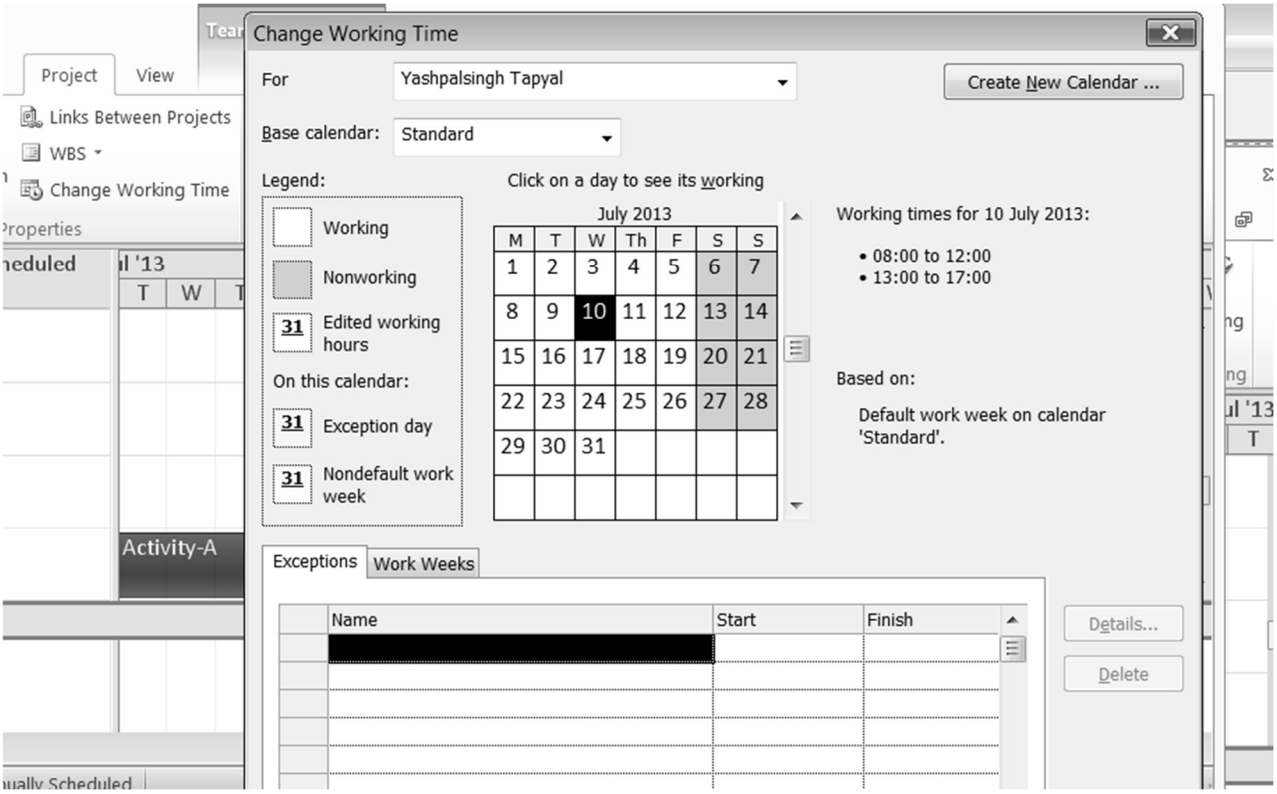
In above diagram 0% on the activity bar shows 0% progress of that activity. Immediately after entering task details we can see calendar view as given follows by clicking view option in Task Ribbon.



Using Project Ribbon we can use option Change Working Time to insert holidays or exception time in which work cannot go ahead. Set Exceptions in the Working calendar. You can also set Exceptional timings for the current project file. Name

the exception and click on details.The following choices appear when you click the Details button on the Exceptions tab in the Change Working Time dialog box. Nonworking Click this option to set the days indicated in the Start and Finish fields of the Exceptions tab as nonworking days.





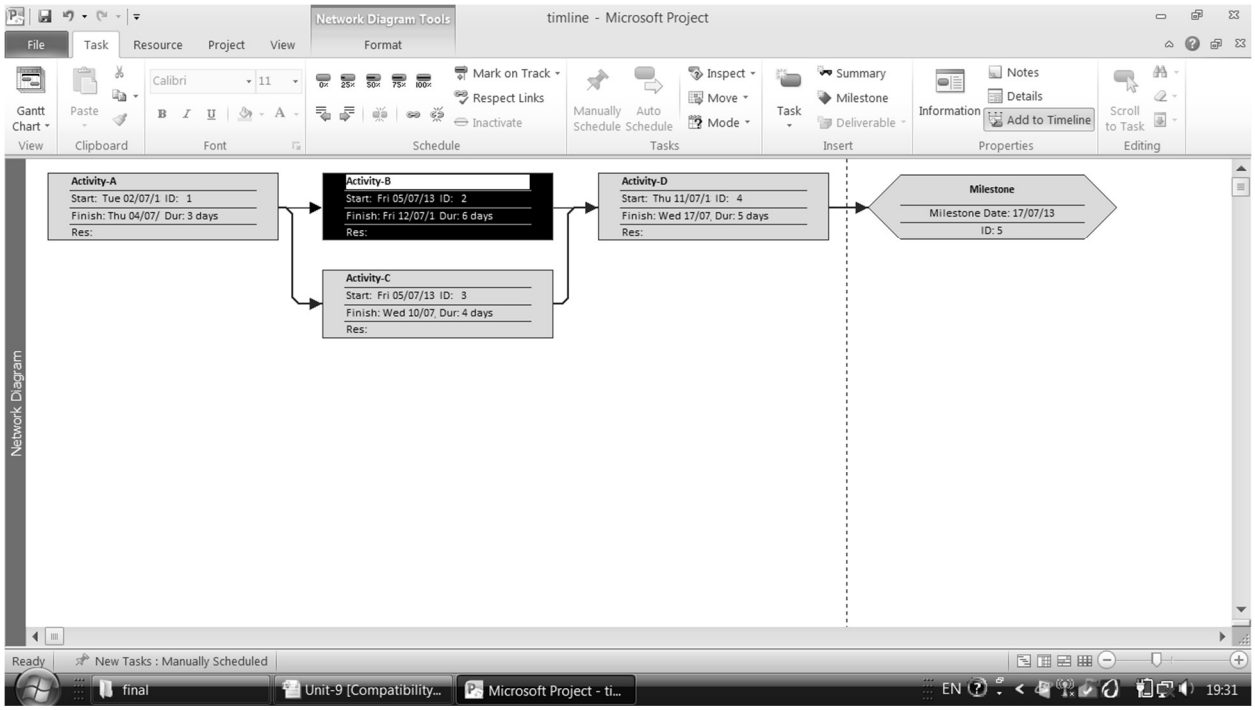
Working times Click this option to set the days indicated in the Start and Finish fields of the Exceptions tab as working days. Consider following as an example.

- Setting Calendar options
- Set the week start day - Monday
- Set the fiscal year start - April
- Set the default start time -9:00
- Se the default end time - 6:00
- Set the hours/day - 8:00
- Set the hours/week - 48:00
- Set the days/month - 26:00

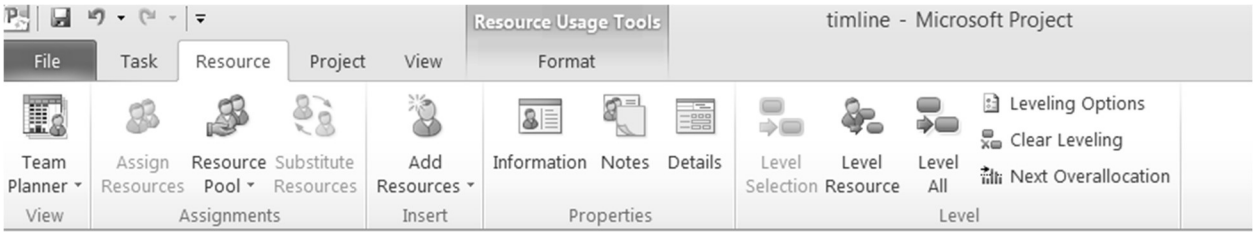
- **Critical Path Method (CPM) :**
  Critical Path Method is a technique of scheduling tasks in a project. The tasks on the longest sequence in a project whose delay expands the project schedule are called critical tasks. The series of tasks that must be completed on time for the project to be satisfactorily completed.
  - What are non-critical task?
    The tasks whose delay does not delay the project end date are called non-critical tasks.

    Following is Network Diagram displayed by view option.

    Sequence of Red Task shows critical path.



- **Resources :**
  Following is Resource Ribbon provides us necessary options to work with resources like detail, notes, information, add resources etc.



Create a resources with Time/Cost and Availability

- Assign resources
- Resolve resource allocations
- Baseline the plan
- Manage - track progress
- Share project information

➢ **Resource Sheet :**

Resource Name	Type	Material Label	Initials	Group	Max. Units	Std. Rate	Ovt. Rate	Cost/Use
	Work				100%	£0.00/hr	£0.00/hr	£0.00

Definition : A resource is an entity which performs effort or work on the task.

There are 3 types of resources :

- Work Resource
  E.g. Man, Equipments and facilities
- Material Resource

  E.g. Cement, Petrol, Diesel, paper etc
- Cost Resource

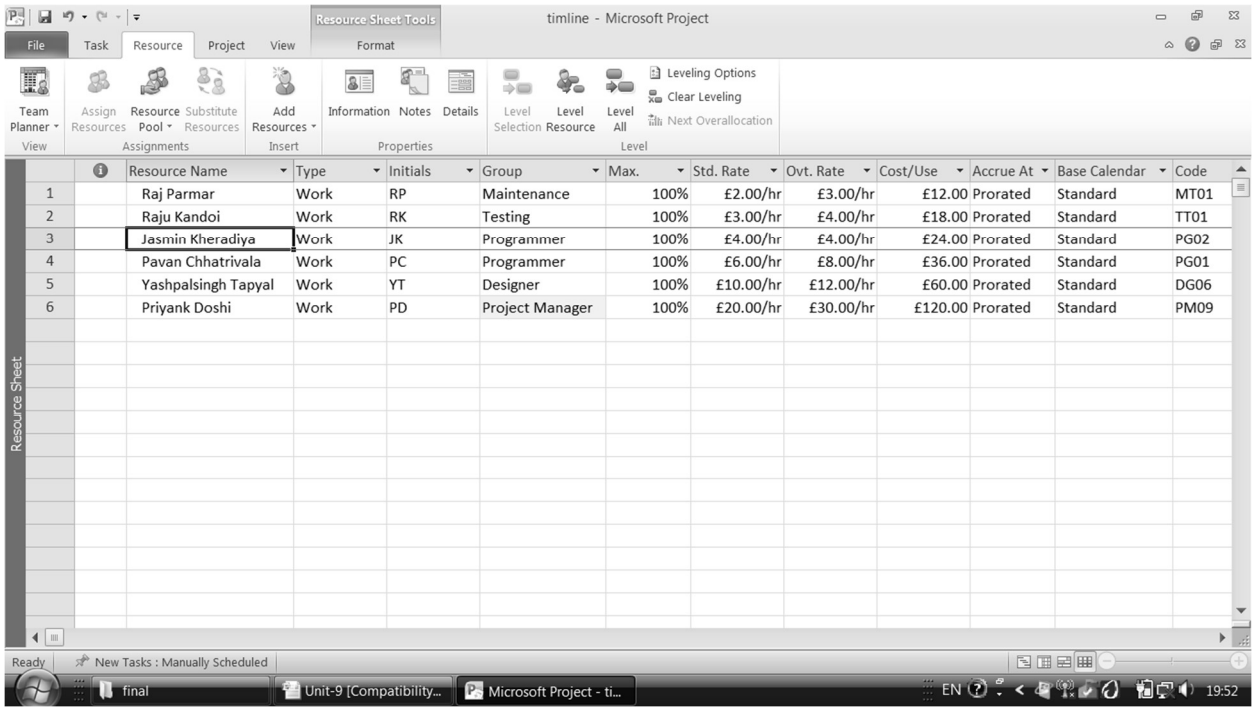
  E.g. Cost Heads like Octroi, Freight, Travelling Expenses, etc.

Fields in Resource Sheet Table :

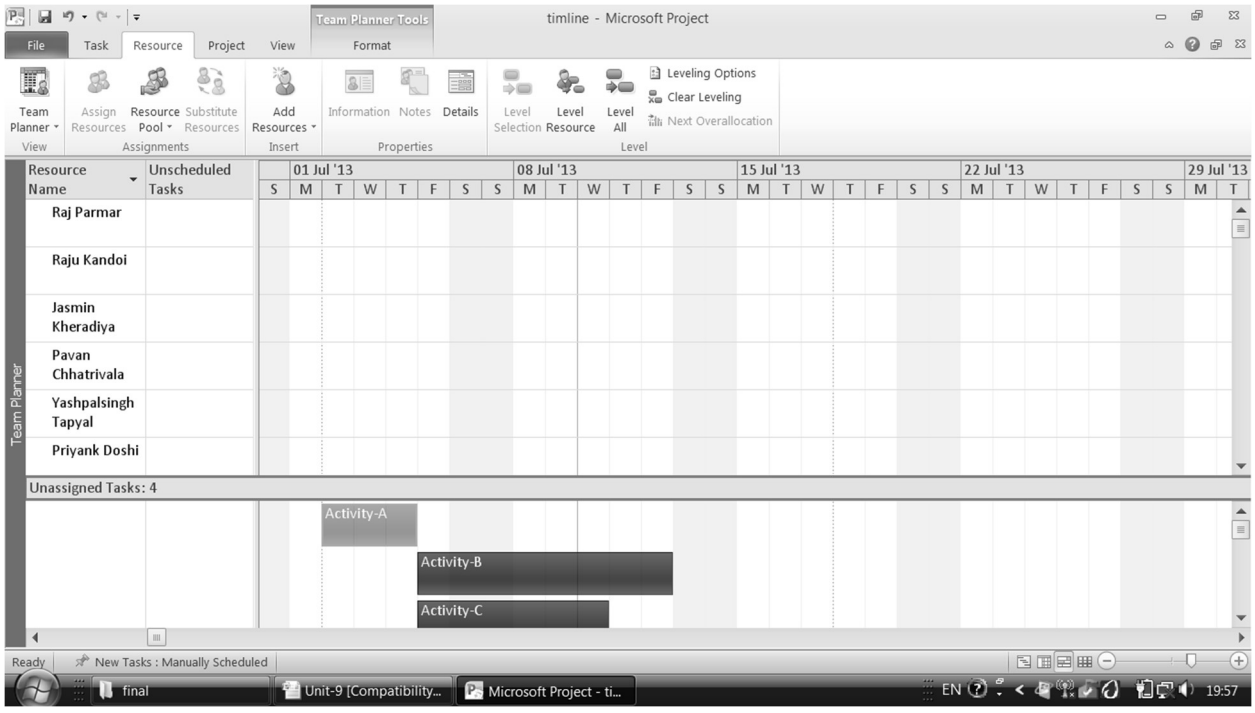
- Customizing MS Project
- Changing the default setting of Scheduling
- Changing the currency format
- Changing the date format
- Creating an Outline number for task
- Changing the default setting

Max Units : This field represents the percentage of time that a resource is available to work on the project plan. The default is 100%, based on the standard calendar that equates to 8 hours per day. Likewise if 50% was entered the resource would be available ½ time or 4 hours per day.

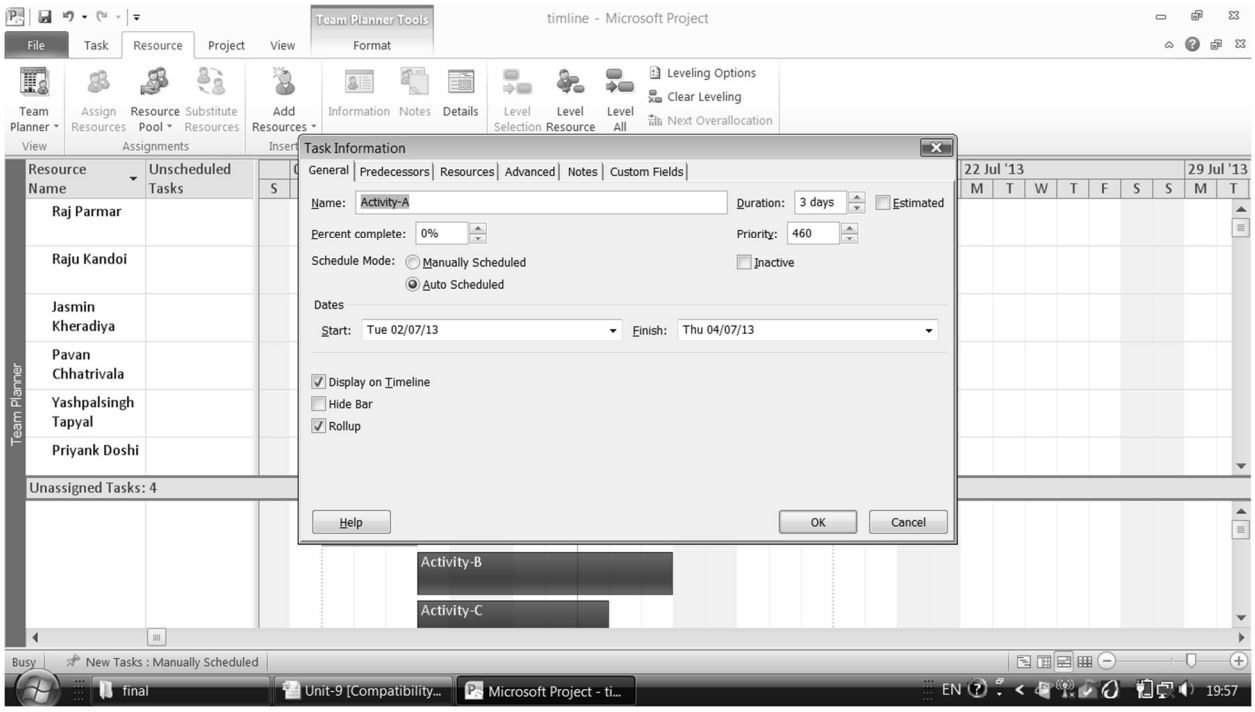
This value can also be displayed as a decimal. To change the default select the Tools menu, select Options and click the Schedule tab and change Show Assignment units as a: from the drop-down list.



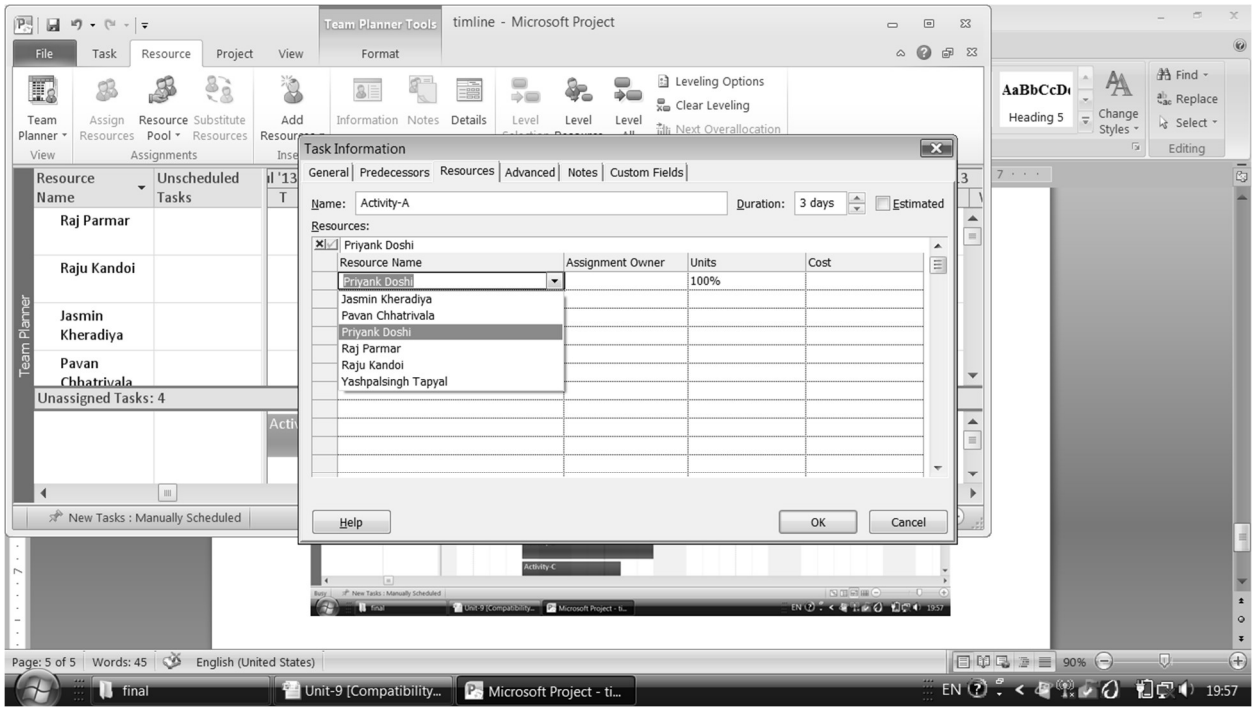
After Entering Resource we can open Team Planner chart to allocate duty to our staff members as manpower resource.



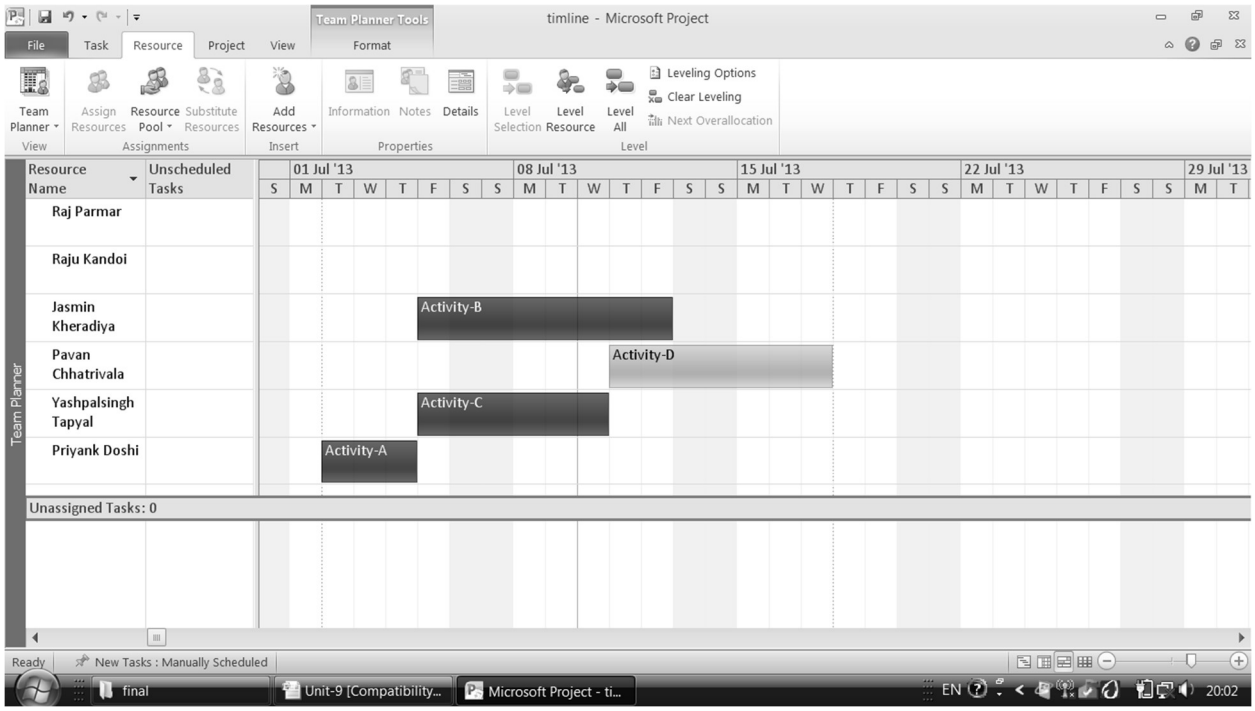
By double clicking unassigned activities we can open task information dialog box.



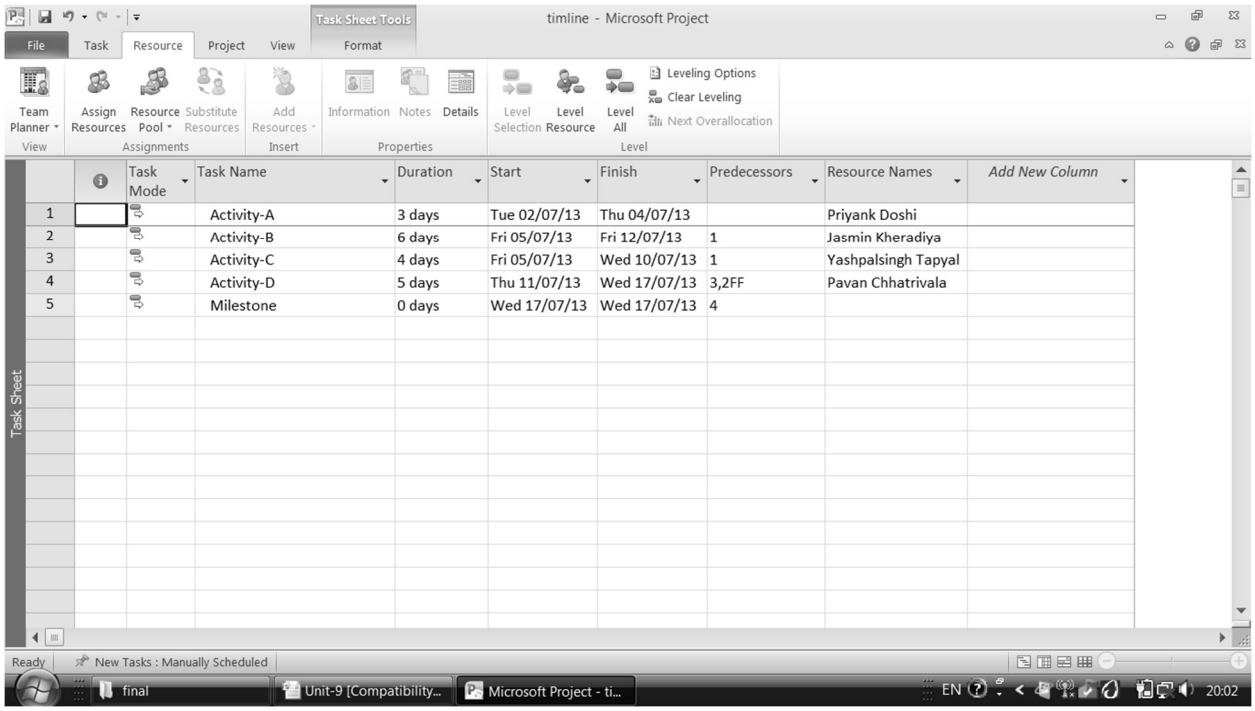
Resource pan shows list of resource like personnel name or equipment etc, Assign activities or task to a particular person as shown below.



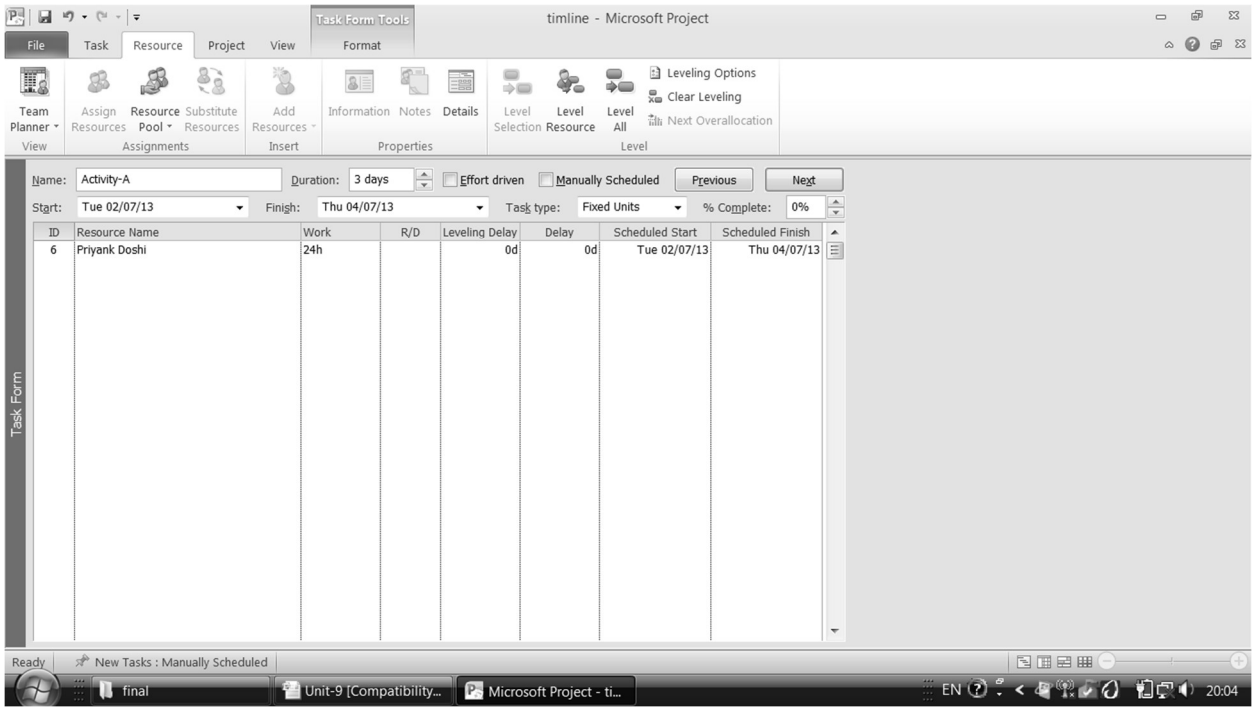
After assigning task to particular person that task would come in the line of name of Responsible person supposed to do that task showing calendar timings.

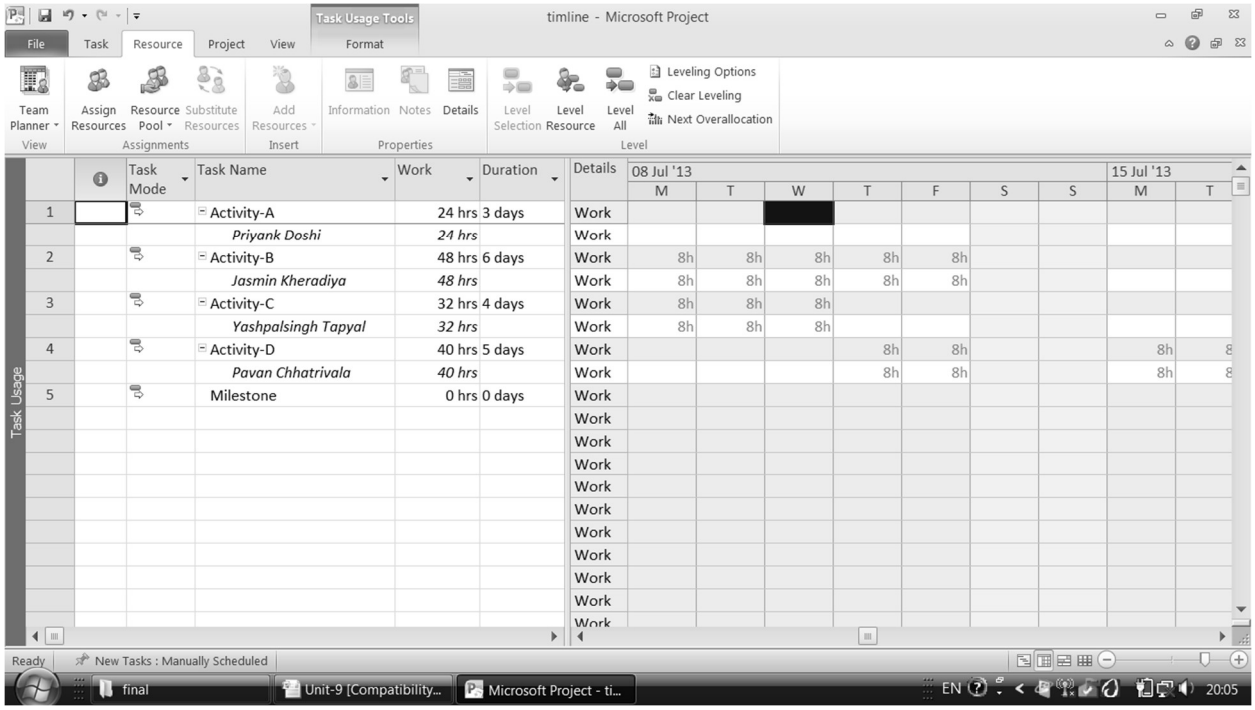


Now following task sheet will show Resource Names entered in the resource sheet tool.

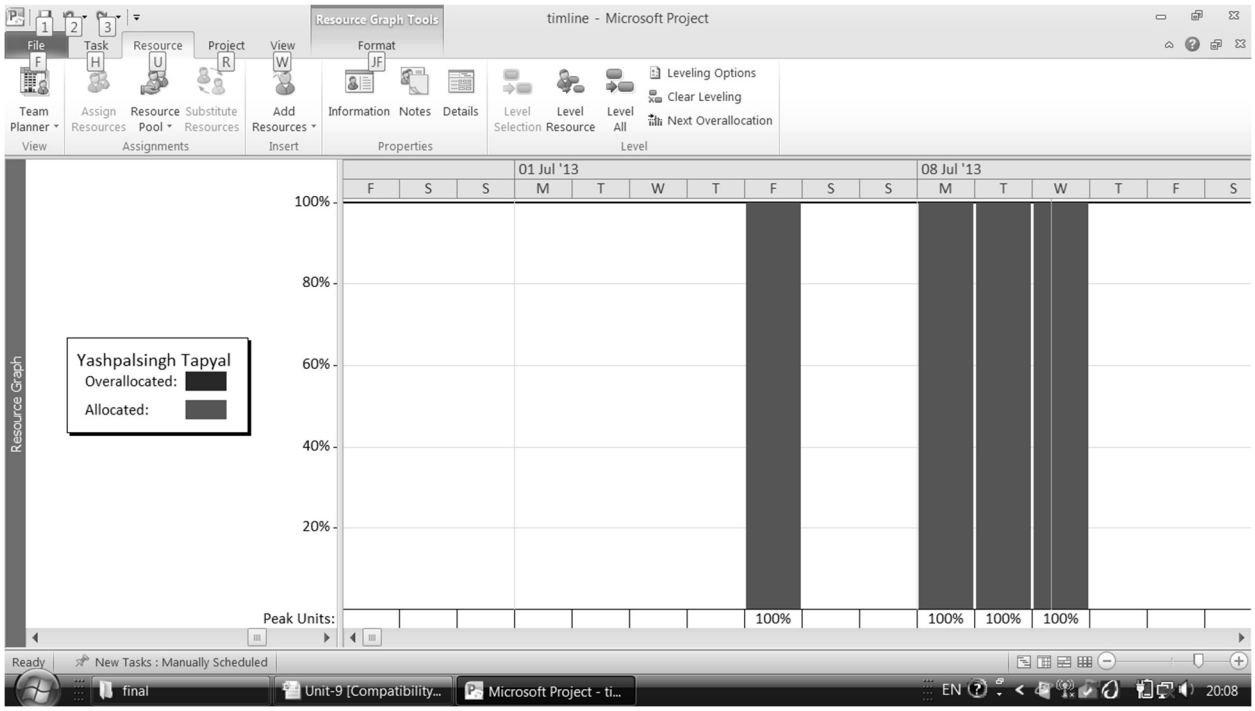


Following is the task form tool and task usage tool to enter details of task and track the usage of each task.

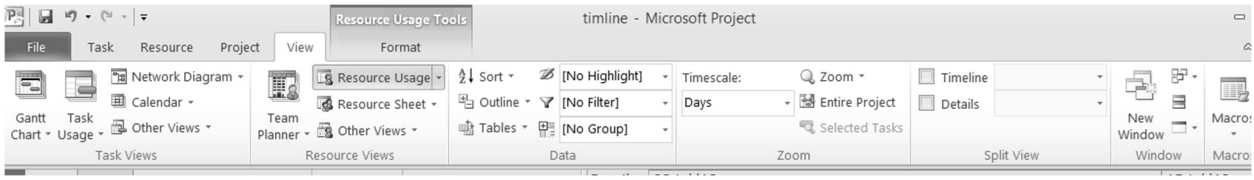




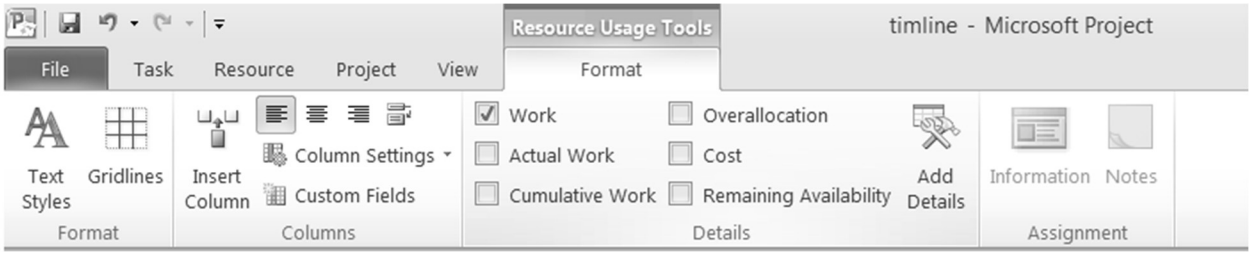
Following is Resource Graph tool to view graphical format of each resource allocated in terms of unit and calendar days to manage all resources (including manpower) successfully.



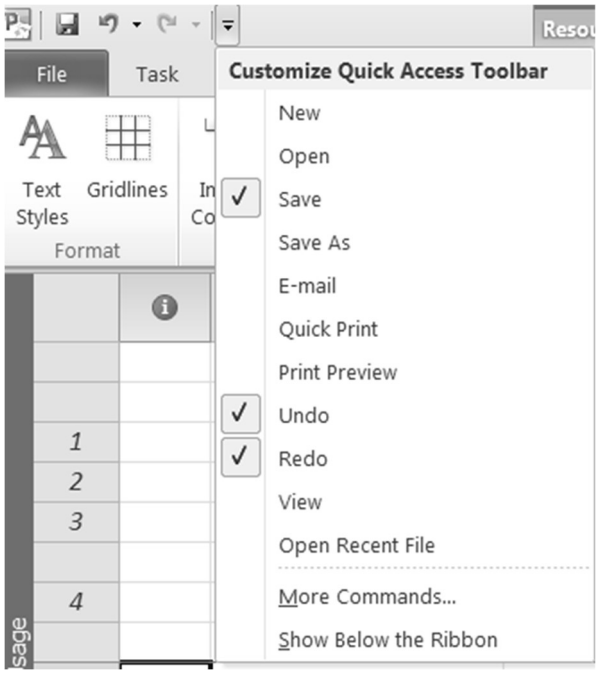
Using View Ribbon we can open team planner, resource sheet with other view including new window, arrange window, filter , sort etc options.



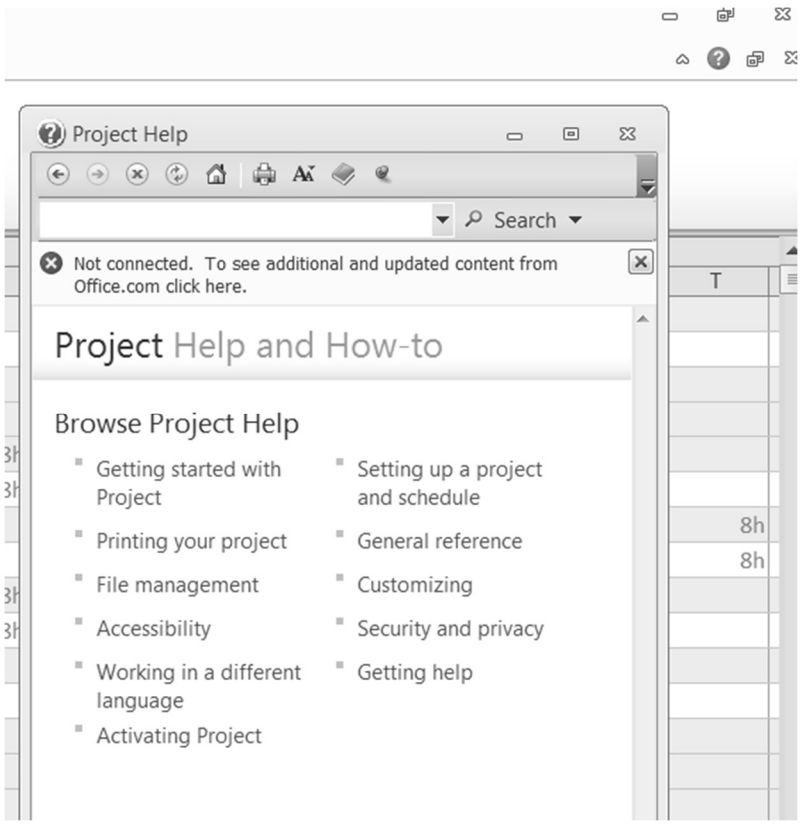
Format of drawings and text can be changed by Format ribbon which provide us option like Text Style, Gridlines, insert columns, and add details, alignment and other necessary information.



For file management we have option to Customize Quick Access Toolbar so that we can save open, print etc. project files quickly and easily.

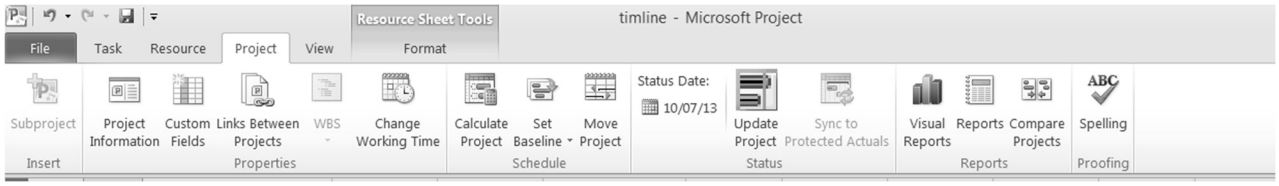


To get necessary help about any option of Ms Project We can use Help (F1) option for each. On right side of window we can press "?"symbol or press F1 key and we can get following help window showing documentation.

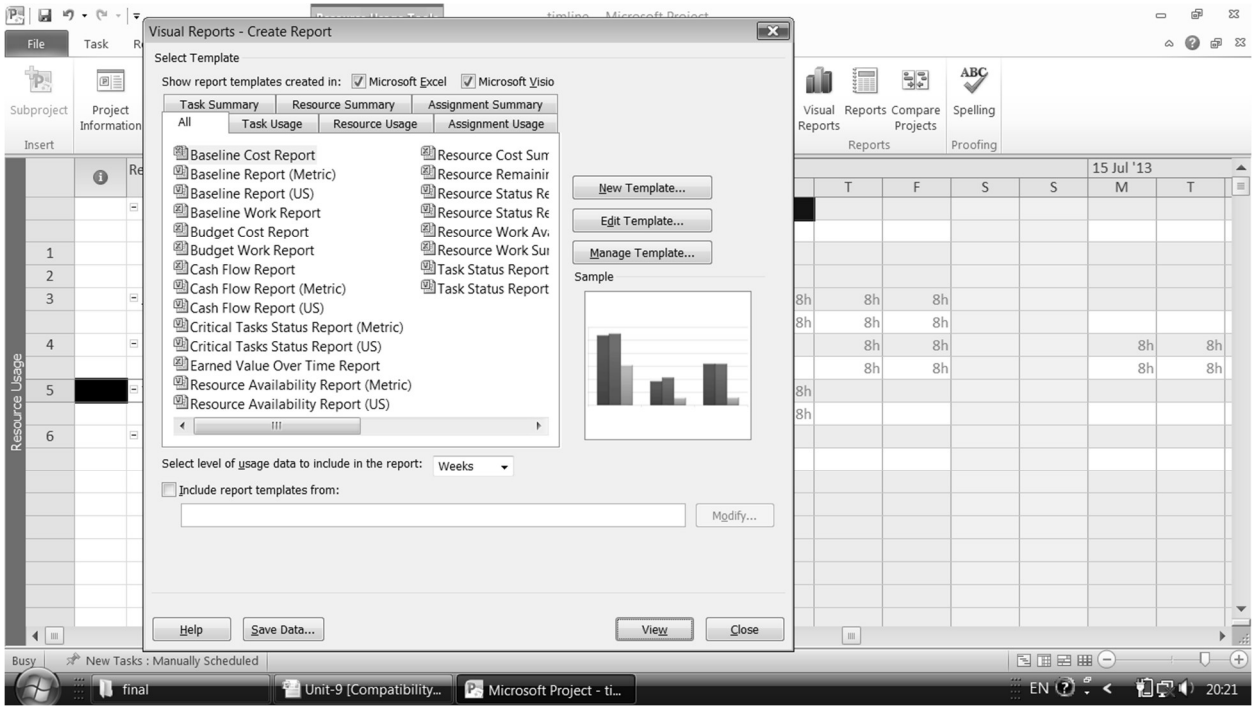


Project Ribbon gives us to work with operation by which we can manage project by changing working time , entering project information, custom fields, comparing

projects and much more important that updating project as project work grows time by time.



By updating project status a project manager can track accomplished task and remaining task to finish. For tracking we can use visual reports like base like cost report, budget cost report etc give as follow.



UNIT- V: SOFTWARE QUALITY ASSURANCE AND TESTING

INTRODUCTION TO SOFTWARE TESTING

Testing of Software carried out in the 5th phase of system development life cycle. This activity of testing tries to remove all possible errors occurred in either design or development phase. Testing increases quality and reliability of software. Fully tested software does not give guarantee of 100% error removal. So a tester tries to remove maximum possible errors. To do this he applies different levels and different techniques in each level. Testing discovers strength and weakness of our software. Untested

software can give many surprises after delivery which may convert into loss.

Software testing is a task to provide information about the quality of the product or service. Software testing also gives actual view of software to run the business to appreciate and understand the threats of software implementation. Test techniques include, but are not limited to, the process of executing a program or application and finding errors. It increases overall quality of software.

Software testing verifies that developed software

- Requirements satisfaction as per SRS document.
- Users implicit expectations.
- Completion according to users point of view.

Software testing can be started any time in the development process. Conventionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but the test effort is ongoing process depends on the methodology.

We required following to carry out testing process.

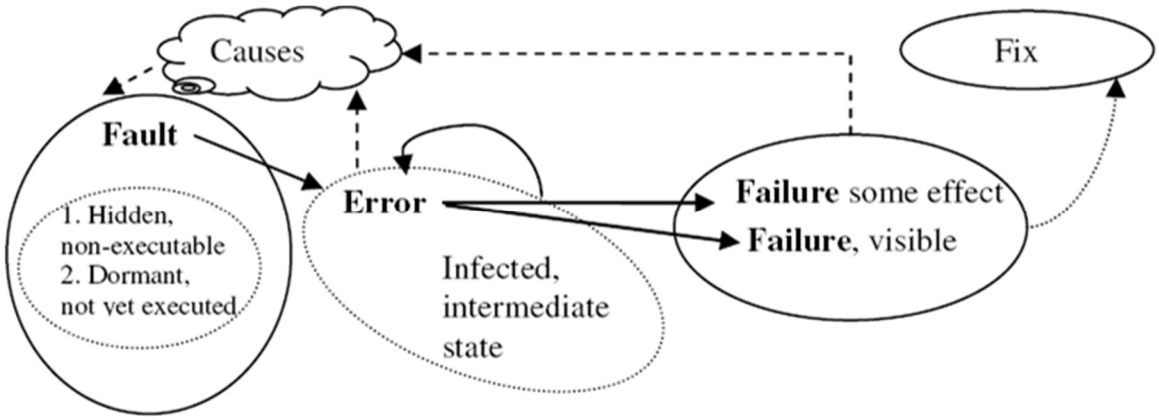
(1) Software requirement specification for expected output of software.

(2) We use coding which is generated in development phase which is to be tested.

(3) Test data. (test cases)

We compare output of software giving input test data. If output does not match with expected output during sample run, coding is corrected to achieve expected output.

SOFTWARE FAULT & FAILURE



TESTING PROCESS TERMINOLOGY

- **Fault :**
  A fault is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification.

  When we use term fault in existing code it means there may be some incorrect state of a instruction (set of instruction) or incorrect value of a single variable (set of variable). Later on we detect specific error from these.

Faults or defects are sometimes called "bugs." Use of term bug underestimates the impact of faults have on software quality. Use of the term "defect" is also associated with software artifacts such as requirements and design documents. Defects occurring in these artifacts are also caused by errors and are usually detected in the review process.

- **Failures :**
  When Software stops working it is considered as failed. An error in a software may from core part (logic of processing data) or from non-core part. An error in a software may or may not arise failure. Failure depends on seriousness of error.

  Failure may be of the type temporary or permanent depends on input. Permanent failure stops working for all input data while temporary failure occurred only for specific input. Alternatively Bohrbugs are permanent design faults and hence almost deterministic in nature and Heisenbugs, on the other hand, belong to the class of temporary internal faults and are intermittent.

  To recover and achieve consistent state from failure depends on seriousness or failure. Some are automatically recovered by pre-defined routines otherwise we have to have to intervene by rebooting or restarting software.

- **Bug :**
  Basically the term bug is used for an error or fault. This is a general term. When computer had been invented, thousands of vacuum tubes were used in its circuit. These increase its size. There were many incidents when insects were attracted by tubes and burnt in to the circuit so work cannot be continued. Until we clean the circuit by removing bugs computer cannot start its work. Gradually all kind of errors started to identify by bugs roughly. Similarly we started using related term debugging for removing errors. If you can classify by example as follows.

  - **Arithmetic bugs :**
    - Division by zero.
    - Arithmetic overflow or underflow.
    - Loss of arithmetic precision due to rounding.
  - **Logic bugs :**
    - Infinite loops and infinite recursion.
    - Counting one to many or too few when looping.
  - **Syntax bugs :**
    - Grammar unfamiliarity of any programming language. For example, in some languages A=10 will set the value of x to 10 while A==10 will check whether A is currently 10 or some other number. In other language same operator "=" is used to compare and to assign.
    - Prototype of function may differ from one language to another.
  - **Resource bugs :**

- Underflow or overflow problems.
- Null pointer dereferences.
- Using an uninitialized variable.
- Using an otherwise valid instruction on the wrong data type (see packed decimal/binary coded decimal).
- Access violations.
- Excessive recursion which though logically valid causes stack overflow.

❖ **Performance bugs :**

- Too high computational complexity of algorithm.
- Random disk or memory access.

➢ **Error :**

Error is incorrect statement or instruction in a program. Errors are following types commonly recognized by us.

(1) Compilation Errors : These types of errors are occurred when we compile the code. Generally misspelled by the programmer while writing code or a instruction results in this type of errors. In absence of exact knowledge of syntax of a function or programming construct, loop structure, control structure etc. ambiguity may arise and instruction written which is not proper to compile by compiler.

Ex : In the loop construct you starts with 'IF' but you forgot to write 'EndIf' at the last

(2) Run Time Errors:- At the execution time program faces of problem that is very difficult to carry out. Mostly program execution stopped. Using debugging procedure step by step run time errors can be eliminated.

Ex : divide by zero error. C= A/B, B must not be zero. Division by zero is not defined mathematically.

(3) Logical Errors:- When we get no error in compilation but the output what we are getting is not as per our expectation then there is some logical mistake in the writing code or in designing flowchart or algorithm. So we consider it as a logical mistake. Logical errors are very difficult to find and fix.

Ex : For example, you might have a variable named FirstName that is initially set to a blank string. Later in your program, you might concatenate FirstName with another variable named LastName to display a full name. If you forgot to assign a value to FirstName, only the last name would be displayed, not the full name as you intended.

➢ **Defect :**

When software is in operation at client side after delivery and implementation, if user gets any kind of error or unexpected behavior then the software is considered as defective and error is considered as defect. Basically defect is an error when software is actually operated by user with real data.

TESTING ARTIFACTS

- **Test case :**

  When we run software for the purpose of testing we need to provide inputs. We cannot provide all input because it is voluminous and tedious job. So we have to select input data by different ways. That forms a test case.

  Test cases identified by a triplet [I, S, O]. I stands for Input, S stands for State of software and O stands for expected out when I has been provided. A software may have different screen for different user containing different use cases. They have different inputs because their level of working (states) are different. When we have multiple test cases we can identified by suffix like,

  [I1, S1,O1],

  [I2,S1, O2],

  …

  [I1, S2,O1],

  [I2, S2, O2],

  …. and so on.

  The known input should test a precondition and the expected output should test a post condition.

  Test case may have following information.

  - test case ID
  - test case description
  - test step or order of execution number
  - related requirement(s)
  - depth
  - test category
  - author
  - pass/fail
  - remarks

- **Test Script :**

  The major advantage of Automated testing is that tests may be executed continuously without the need for a human intervention.
  In the case of automated testing we need to write small script to govern testing procedure. We can use instructions like PAUSE, START, STOP, PRINT etc. when instruction inserted within test case for the ease of automated testing it is known as test script.

  Test script can either be written using a special automated functional GUI test tool (such as HP QuickTest Professional, Borland SilkTest, and Rational Robot) or in a

well-known programming language (such as C++, C#, Tcl, Expect, Java, PHP, Perl, Python, or Ruby).

- **Test Harness :**

  Test harness is automated test framework consisting Test execution engine and Test script repository. Test harnesses enable us to carry out automation of tests. Using routines of software input data provided and output in the form of print out taken and result are compared. Goal of a test harness are following.

  - Automate the testing process.
  - Execute test suites of test cases.
  - Generate associated test reports.

  A test harness may provide some of the following benefits:

  - Increased productivity due to automation of the testing process.
  - Increased probability that regression testing will occur.
  - Increased quality of software components and application.

  There are various types of frameworks. They are categorized on the basis of the automation component they leverage. These are:

  (1) Data-driven testing

  (2) Modularity-driven testing

  (3) Keyword-driven testing

  (4) Hybrid testing

  (5) Model-based testing

  (6) Code driven testing

- Example of Test Atomization tools :

Tool name	Produced by
AutoIt	Jonathan Bennett & AutoIt Team
Cucumber	Open Source
eggPlant	TestPlant
EiffelStudio AutoTest	Eiffel Software
FitNesse	Open Source
HP QuickTest Professional	HP Software Division
IBM Rational Functional Tester	IBM Rational
LabVIEW	National Instruments
Maveryx	Maveryx
Oracle Application Testing Suite	Oracle Corporation
QF-Test	Quality First Software GmbH
Ranorex	Ranorex GmbH
Rational robot	IBM Rational
Robot Framework	Open Source
Selenium	Open source

Sikuli	Open Source
SilkTest	Borland
TestComplete	SmartBear Software
Testing Anywhere	Automation Anywhere
TestPartner	Micro Focus
Test Studio	Telerik
Time Partition Testing (TPT)	PikeTec GmbH
TOSCA Testsuite	TRICENTIS Technology & Consulting
Visual Studio Test Professional	Microsoft
Watir	Open Source

- **Test Suite (Set of Test Cases) :**
  Test suite is the collection of different test cases. Developed differently depending on different techniques.

  Test suites mainly contains

  - Test summary
  - Configuration

  A test suite sometime known as a validation suite, is a collection of test cases that are planned to be used to test a software program to show that it has some specified set of behaviors. A test suite contains goal of each collection of test cases and information on the system configuration to be used during testing. A group of test cases may also contain prerequisite states or steps, and descriptions of the following tests.

  An executable test suite (test script) is a test suite that can be executed by a program. This usually means that a test harness which is integrated with the suite. The test suite and the test harness together can work with the system under test (SUT).

- **Test Plan :**
  A test specification is called a test plan. This is also referred as test strategy. This is used by management and the developers. When we change code or adding some routines it become require more caution under test plan. Test engineers prepare this document.

  A test plan include following :

  - **Test Coverage :** Test Coverage is derived from design specifications and other requirements. Some requirements may be verified during Design Verification test, but not repeated during Acceptance test
  - **Test Methods :** It shows method of implementation of testing procedure which include standards, performance, documentation etc.
  - **Test Responsibilities :** Includes planning, acquiring test equipment and other resources necessary to implement the test methods for which they are responsible. Also includes what data will be collected, and how that data will be stored. Outcome of a successful test plan should be a recorded. Verification of all design specifications and requirements must be satisfied.

Except these a test plan may includes following.

- *Design Verification or Compliance test* - to be performed during the development or approval stages of the product, typically on a small sample of units.
- *Manufacturing or Production test* – while software is developed, parallel this test can be performed for verification and quality control .
- *Acceptance or Commissioning test* – testing at the time of delivery or installation of the product.
- *Service and Repair test* – indicates service life of the product.
- *Regression test* – modification or addition of a routine may not disturb other modules so that regress test performed.

IEEE 829 test plan structure is as follows. IEEE, pronounced "Eye-triple-E", stands for the Institute of Electrical and Electronics Engineers. The association is chartered under this name and it is the full legal name.

IEEE 829-2008, also known as the 829 Standard for Software Test Documentation, is an IEEE standard that specifies the form of a set of documents for use in defined stages of software testing, each stage potentially producing its own separate type of document.

- Test plan identifier
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Suspension criteria and resumption requirements
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

STATIC TESTING

In Static testing software isn't actually used. This checks algorithm in document and

checks structural representation without executing it. It is primarily syntax checking of the code and manually reviewing the code or document to find errors. This can be used to care for coding standards. We can also find uninitialized variable, parameter mismatching in prototypes, variable declared but not used.

 Informal Review :

Software reviews can find issues earlier and more cheaply than they would be identified by testing or by defect detection process. The cost to find and fix a defect by review is very less than when the same defect is found by test execution.
These reviews can be used to train personnel who prepare documentation easily and they can contribute in defect prevention process.

Without waiting the work has been completed, early and frequent reviews work samples can identify errors, which can be corrected before committing more erroneous work. This reduces overall development time and used to develop high-quality software, and dramatically decrease the error-rate.

 Walkthrough :

A designer or programmer organize peer review and invite members of the development team and other interested parties to study a software product including coding, and the participant can ask questions and make comments about possible errors, violation of development standards, and other problems.

 Formal Technical Review :

A method involving a structured encounter in which a group of technical personnel analyzes or improves the quality of the original work product as well as the quality of the method.

Software Quality Improvement can be achieved by improved quality of the original work and finding defects early (less costly). It reduces defects and leads to improved productivity. Reducing rework build throughout the project Requirements design, coding , testing may reduce time and cost. More efficiently project can be accomplished.
The review team prepares a formal review plan around the objectives of the review, the type of evaluation to be carried out and the time schedule required. An overall plan covers the following areas:

(1) Administrative Plan: Review area objectives, operating costs, actual operating Performance and benefits.
(2) Personnel requirements Plan: Review performance objectives and training performance to data.
(3) Hardware Plan: Review performance specifications
(4) Documentation Review Plan: Review the system development effort.

The review not only assesses how well the current system is designed and implemented, but also is a valuable source of information that can be applied to the next systems project.

The review team prepares a formal review plan around the objectives of the review, the type of evaluation to be carried out and the time schedule required. The review plan cover the following areas:

(1) Administrative Plan : The following two activities are reviewed under this plan
User Objective : This is an extremely critical area since it may be possible that over a period of time either the system does not meet the initial objectives of the user or the user objectives get changed as a result of changes in the overall objectives of the organization. The results of the evaluation are documented for future reference.
Operating Costs and Benefits : Under the administration plan, current budget designed to manipulate the costs and savings of the system is closely reviewed.

(2) Personnel Requirement Plan : Under this plan, all activities involving system personnel and staff members associated with the system are evaluated. After the plan is developed, the review group evaluates:
Personnel performance objectives compared with current performance levels
Training performance through testing, conducting interviews and other data gathering techniques.

(3) Hardware Plan : The hardware of the new system is also reviewed including terminals, CRT and communication network. The main target is a comparison of current performance specifications with design specifications. It also points out necessary modification to be made.

(4) Documentation Review Plan : The reason for review plan is to evaluate the accuracy and completeness of the documentation compiled to date and to its conformity with documentation standards established earlier.

 Inspection :

An inspection is a visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications.
It is not widely used because following reasons.

- Lack of time
- Not seen as a priority
- Not seen as value added (measured by LOC)
- Lack of understanding of formalized techniques
- Improper tools used to collect data
- Lack of training of participants
- Pits programmer against reviewers

DYNAMIC TESTING :

Program executed and analysis is performed on its real output. It output recorded for different test cases included in the test suite. Dynamic analysis tool can be used to analyze and produce reports.

Test cases can be generated according to white box or black box it does not matter. The main concept is to produce report in the form of histogram or any line chart etc. showing how much proportion of code executed in terms of branch , code coverage or any other way. The output of dynamic testing can be used as evidence of testing.

Testing Levels :-

- Unit Testing :

In unit testing the analyst tests the programs making up a system. For this reason, unit testing is sometimes called program testing. Unit testing gives stress on the modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained within that module alone. The errors resulting from the interaction between modules are initially avoided.

A program represents the logical elements of a system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly with other programs. it is the responsibility of a programmer to have an error free program. At the time of testing the system, there exist two types of errors that should be checked. These errors are syntax and logic. A syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted key words are common syntax errors. These errors are shown through error messages generated by the computer. A logic error, on the other hand, deals with incorrect data fields out of range items, and invalid combinations. Since the logical errors are not detected by compiler, the programmer must examine the output carefully to detect them.

When a program is tested, the actual output is compared with the expected output. When there is a discrepancy, the sequence of the instructions, must be traced to determine the problem. The process is facilitated by breaking the program down into self-contained portions, each of which can be checked at certain key points.

For example, a hotel information system consists of modules to handle reservations; guest check in and checkout; restaurant, room service and miscellaneous charges; convention activities; and accounts receivable billing. For each, it provides the ability to enter, modify or retrieve data and respond to different types of inquiries or print reports. The test cases needed for unit testing should exercise each condition and option.

Unit testing can be performed from the bottom up, starting with smallest and lowest-level modules and proceeding one at a time. For each module in bottom-up testing a short program is used to execute the module and provides the needed data, so that the module is asked to perform the way it will when embedded within the larger system.

 Integration Testing (String Testing) :

Programs are invariably related to one another and interact in a total system. Each program is tested to see whether it conforms to related programs in the system. Each part of the system is tested against the entire module with both test and live data before the whole system is ready to be tested.

Different modules of a system are integrated and interface errors or communication errors are detected during integration testing. Main objective is to test all module interfaces. For this we have to study dependency of modules to test modules and their interface in proper order. We can integrate module in following order and test for interface errors.

[1] Big Bang Technique
[2] Top down Technique
[3] Bottom up Technique
[4] Mixed Technique

[1] Big Bang :
If all modules are integrated in single step and tested then the case is known as Big Bang. This depends on the size of system. Errors are difficult to locate and fix. This becomes expensive when system having big size.

[2] Top Down :
It starts with Main Program which is calling program and others are called routine. Initially there may be very few modules called in main but gradually other modules are completed, they can be called in the main.

All working modules are leaf nodes in structure chart so until all lower modules are finished main is not able to accomplish the job. So it is sometimes difficult to test upper component in absence of lower modules.

Full system is tested repeatedly after calling routines in main so effect of adding a single modules on other modules is also recognized step by step so this becomes more manageable.

[3] Bottom Up :
Each sub system tested in discrete way and integrated gradually. Modules are integrated after unit testing and component has been formed. Individual components tested separately and sub system generated. All subsystem tested and gathers to form system. In system having big size and several levels this approach gives best result. All team can accomplish their individual testing parallel. Disadvantage is that we can run complete system at the end of each sub system.

[4] Mixed :
In top down technique testing can start only after the top level modules have coded and unit tested while bottom up testing can start only after leaf modules are ready. Mixed techniques eliminates disadvantage of both and testing can be started in both ways finding more interface errors according both techniques. This may give best result to test some systems.

- System Testing :

The important and essential part of the system development phase, after designing and developing the software is system testing. We cannot say that every program or system design is perfect and because of lack of communication between the user and the designer, some error is there in the software development. The number and nature of errors in a newly designed system depend on some usual factors like communication between the user and the designer; the programmer's ability to generate a code that reflects exactly the systems specifications and the time frame for the design.
Theoretically, a newly designed system should have all the parts or sub-systems are in working order, but in reality, each sub-system works independently. This is the time to gather all the subsystem into one pool and test the whole system to determine whether it meets the user requirements. This is the last change to detect and correct errors before the system is installed for user acceptance testing. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits.

Testing is an important function to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully activated. Another reason for system testing is its utility as a user-oriented vehicle before implementation.

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing. In this input data are collected from actual business environment and errors in this process immediately corrected at user's site. This may take short period of time.

Beta testing comes after alpha testing. Beta testing carried out at customer's site. This may take long period of time. This Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users. Errors are corrected and collectively updated in the software.

System testing consists of the following steps
(1) System testing

(2) System documentation
System testing is designed to uncover weaknesses that were not found in earlier tests. This includes forced system failure and validation of total system as it will be implemented by its user in the operational environment. Under this testing, generally we take low volumes of transactions based on live data. This volume is increased until the maximum level for each transaction type is reached. The total system is also tested for recovery and fallback after various major failures to ensure that no data are lost during the emergency. All this is done with the old system still in operation. When we see that the proposed system is successful in the test, the old system is discontinued.

 System Documentation :
All design and test documentation should be well prepared and kept in the library for future reference. The library is the central location for maintenance of the new system.

 Acceptance Testing :
This is performed by customer. An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system's procedures operate to system specifications and that the integrity of important data is maintained. Performance of an acceptance test is actually the user's show. User motivation is very important for the successful performance of the system. After that a comprehensive test report is prepared. This report shows the system's tolerance, performance range, error rate and accuracy.

Smoke testing is used before acceptance testing because smoke test continuously verifies user requirement by asking for confirmation about the progress of development. So in acceptance testing no big blunder can arise.

TECHNIQUES OF SOFTWARE TESTING

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

 Black Box Testing :

Black box testing takes an external perspective of the test object to derive test cases. Test cases are mainly functional so it is also known as functional testing. There is no knowledge of the test object's internal structure. Typical black box test design techniques include Equivalence partitioning, Boundary value analysis, Decision table testing, Pairwise testing, State transition tables, Use case testing , Cross-functional testing .
Specification-based testing intends to test the functionality of software according to the applicable requirements. The tester inputs data and observes the output to the test

object. This level of testing usually requires thorough test cases to be provided to the tester. Tester then compares real output with the expected output for the success of testing process. Specification-based testing is necessary, but it is insufficient to guard against certain risks.

Advantages and disadvantages: The black box tester has no tie up with the code, and a tester's insight is very simple: a code must have bugs. There are situations when A tester writes many test cases to check something that could have been tested by only one test case. Some parts of the back-end are not tested at all.

This is applicable to all levels of software testing: unit, integration, functional testing, system and acceptance. This method can uncover unimplemented parts of the specification, one cannot be sure that all existent paths are tested.

As per SRS document, all requirements are tested by preparing test data under black box testing. For an example if we are testing function of finding roots of any quadratic equation, function works depending on values of value of Δ greater than 0, less than 0 or equal to 0 (functionality of routine). If option for $\Delta < 0$ does not implemented then it can be uncovered in Black box.

- Equivalence Partitioning :

The set of input values from which we are generating test cases is divided into sets of similar kind of values. Assuming that similar kind of inputs will give similar types of outputs. This gives facility to choose some of the values (considered as representatives of sets from which they belong) from these partitioned sets. We give inputs of these values and compares output if it is as per expectation or different.

The sets containing similar kind of test cases is known as Equivalence classes. Classes can be formed as follows.
(1) Valid values or Invalid values. -ve value of age is invalid and more than 100 is rare so if we decide valid range [0,100] then less than 0 and greater than 100 becomes invalid set of equivalence classes.
(2) Even values or Odd values. Even values : {2,4,6,…} Odd values : { 1,3,5,7…}
(3) Sets having remainder 0, 1, 2, 3, 4 when divided by 5. Or similar like this.

- Boundary Data Analysis :

When we test boundary values as a test case, we also need that if we input values immediately below lower boundary and immediately greater than upper boundary software should be robust. It should not give error. If it will fail by providing these values it is not considered as flexible.

If our domain of input is (1, 1000). Test must be performed with test cases with values like {-1,1,1000,1001} along with other test cases.

AUTOMATED TESTING USING (ANY FREEWARE OR OPEN SOURCE) SOFTWARE TESTING TOOLS.

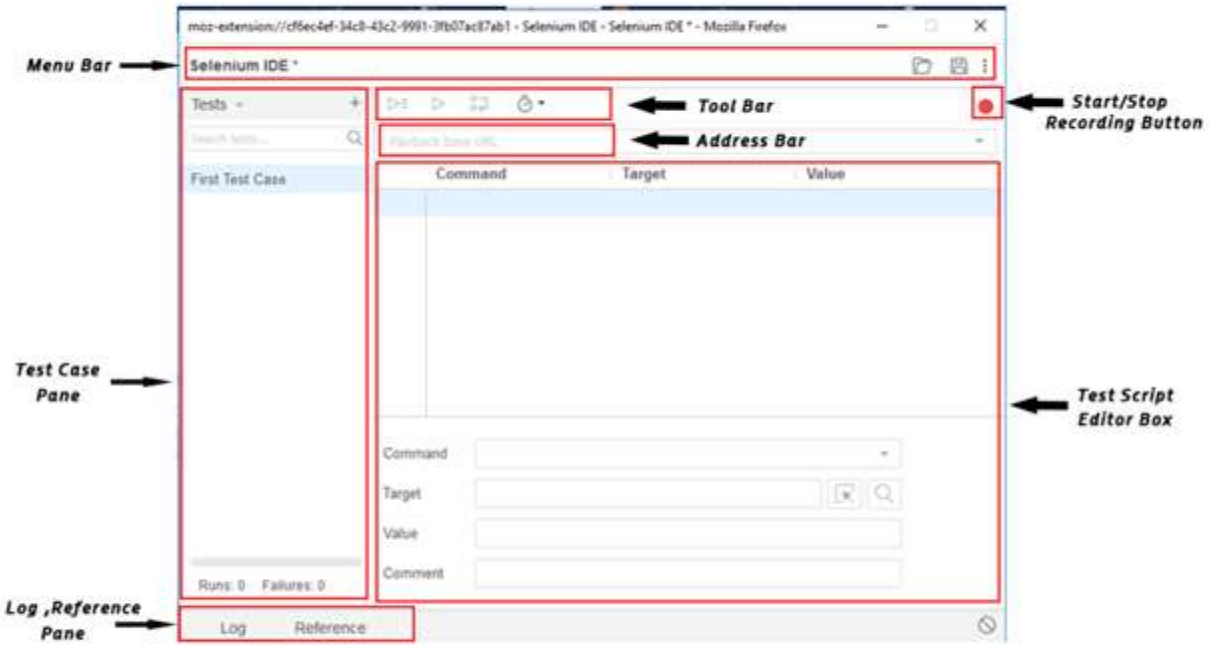
Selenium Integrated Development Environment (IDE)

Selenium IDE (Integrated Development Environment) is an open source web automation testing tool under the Selenium Suite. Unlike Selenium WebDriver and RC, it does not require any programming logic to write its test scripts rather you can simply record your interactions with the browser to create test cases. Subsequently, you can use the playback option to re-run the test cases.

Selenium IDE-Features

Selenium IDE is divided into different components, each having their own features and functionalities. We have categorized seven different components of Selenium IDE, which includes:

Menu Bar
Tool Bar
Address Bar
Test Case Pane
Test Script Editor Box
Start/Stop Recording Button
Log, Reference Pane



Now, we will look at the features and functionalities of each component in detail.

1. Menu Bar
Menu bar is positioned at the top most portion of the Selenium IDE interface. The most commonly used modules of menu bar include:

Project Name
It allows you to rename your entire project.

Open Project
It allows you to load any existing project from your personal drives.

Save Project

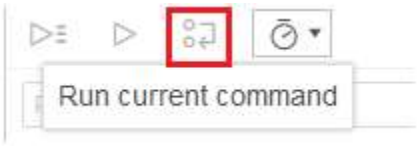
It allows you to save the entire project you are currently working on.

2. Tool Bar

The Tool bar contains modules for controlling the execution of your test cases. In addition, it gives you a step feature for debugging you test cases. The most commonly used modules of Tool Bar menu include:

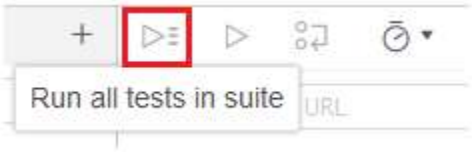
Speed Control Option
It allows you to control the execution speed of your test cases.

Step Feature
It allows you to "step" through a test case by running it one command at a time. Use for debugging test cases.



Run Tests
It allows you to run the currently selected test. When only a single test is loaded "Run Test" button and "Run all" button have the same effect.

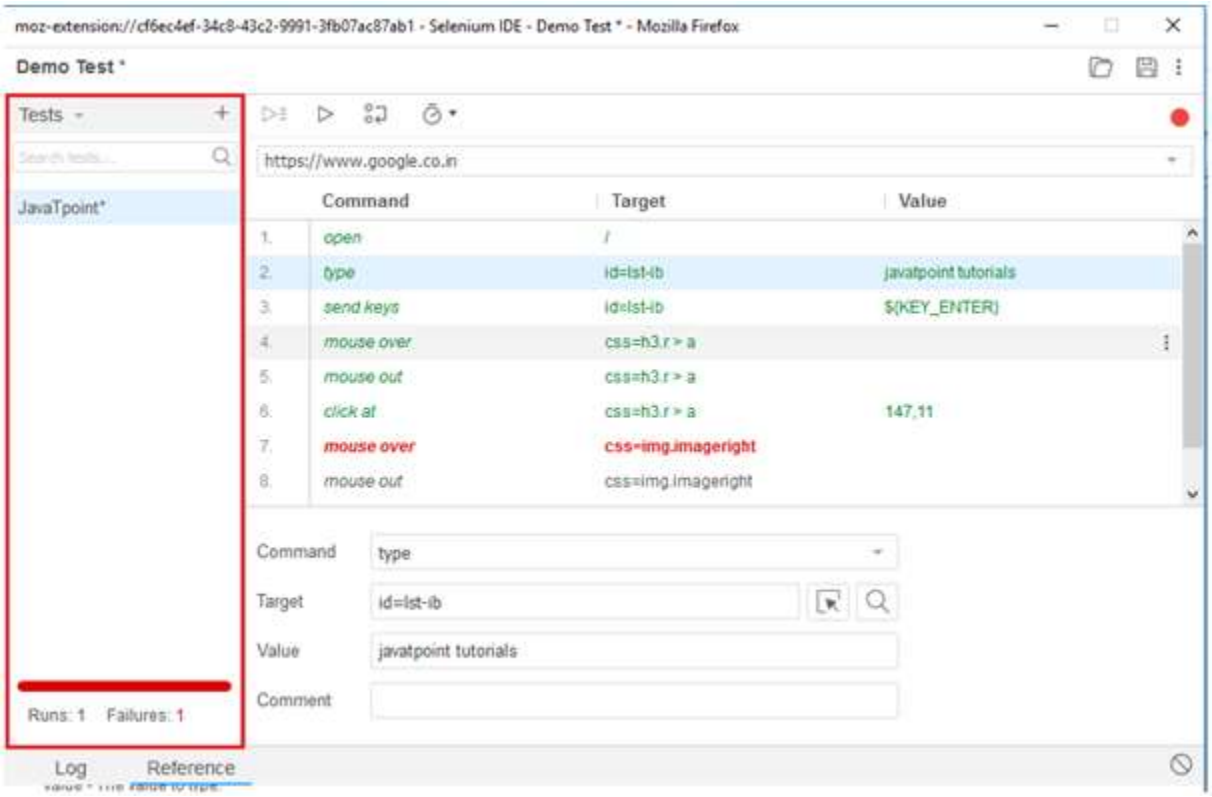
Run All
It allows you to run the entire test suite when a test suite with multiple test cases is loaded.



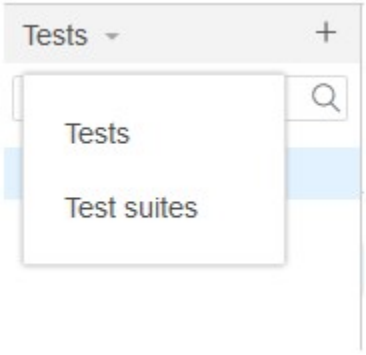
3. Address Bar
This module provides you a dropdown menu that remembers all previous values for base URL. In simple words, the base URL address bar remembers the previously visited websites so that the navigation becomes easy later on.

4. Test Case Pane

This module contains all the test cases that are recorded by IDE. In simple words, it provides the list of all recorded test cases at the same time under the test case pane so that user could easily shuffle between the test cases.

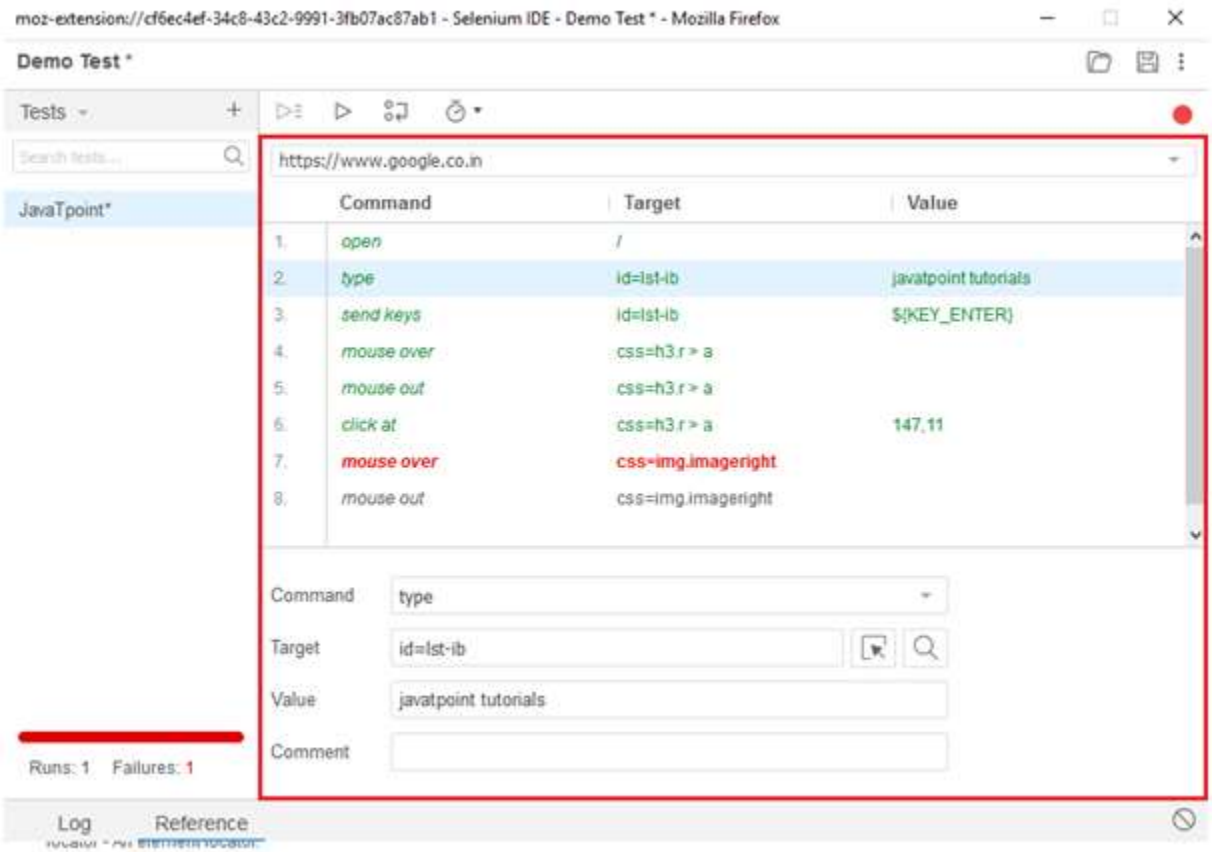


At the bottom portion of the Test Case Pane, you can see the test execution result summary which includes the pass/fail status of various test cases.
Test Case Pane also includes features like Navigation panel which allow users to navigate between test cases and test suites.



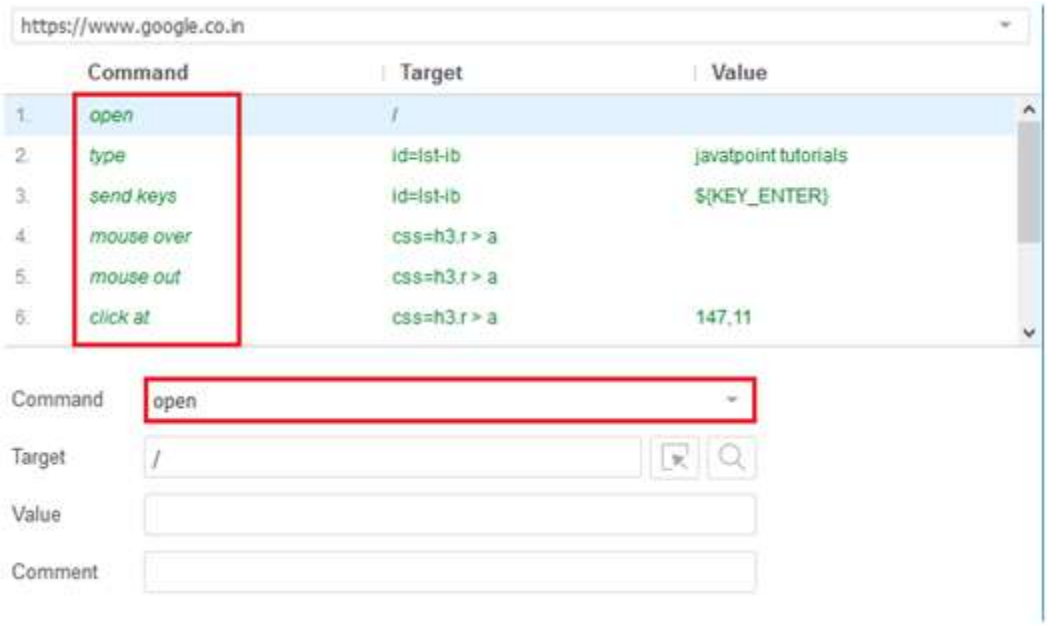
5. Test Script Editor Box

Test Script Editor Box displaysall of the test scripts and user interactions that were recorded by the IDE. Each user interaction is displayed in the same order in which they are performed. The Editor box is divided into three columns:Command, Target and Value.



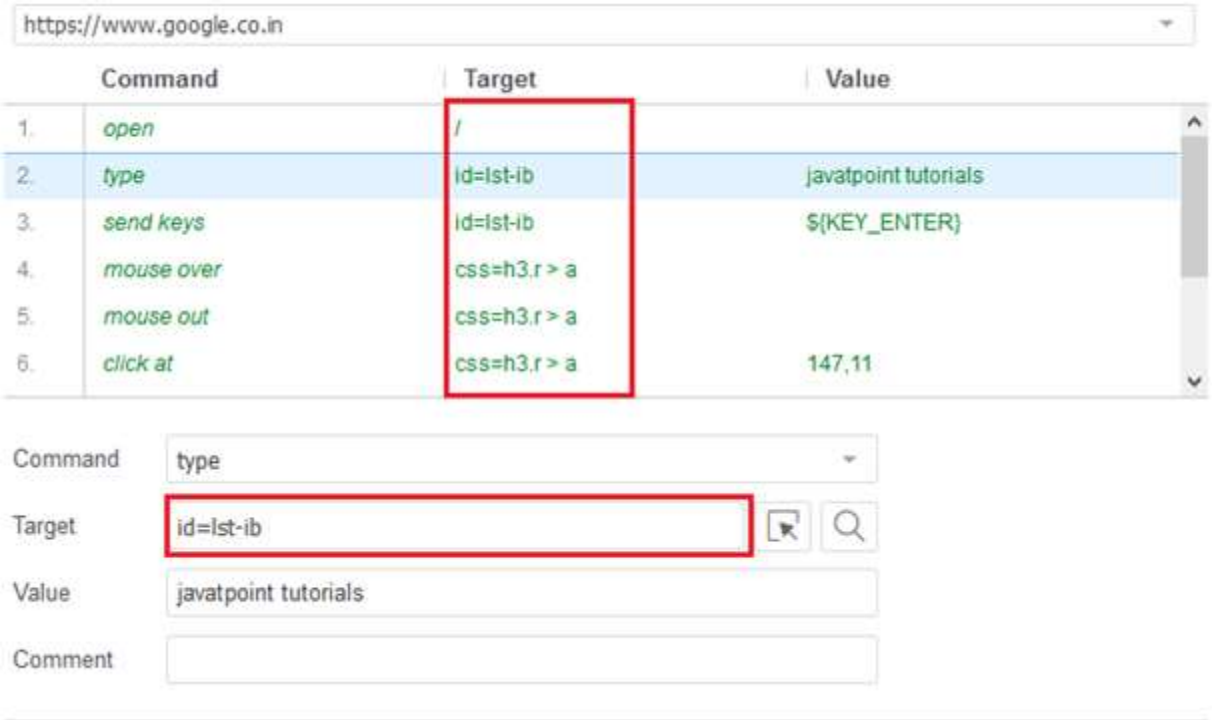
Command:

Command can be considered as the actual operation/action that is performed on the browser elements. For instance, if you are opening a new URL, the command will be 'open'; if you are clicking on a link or a button on the web page, then the command will be 'clicked'.



Target:

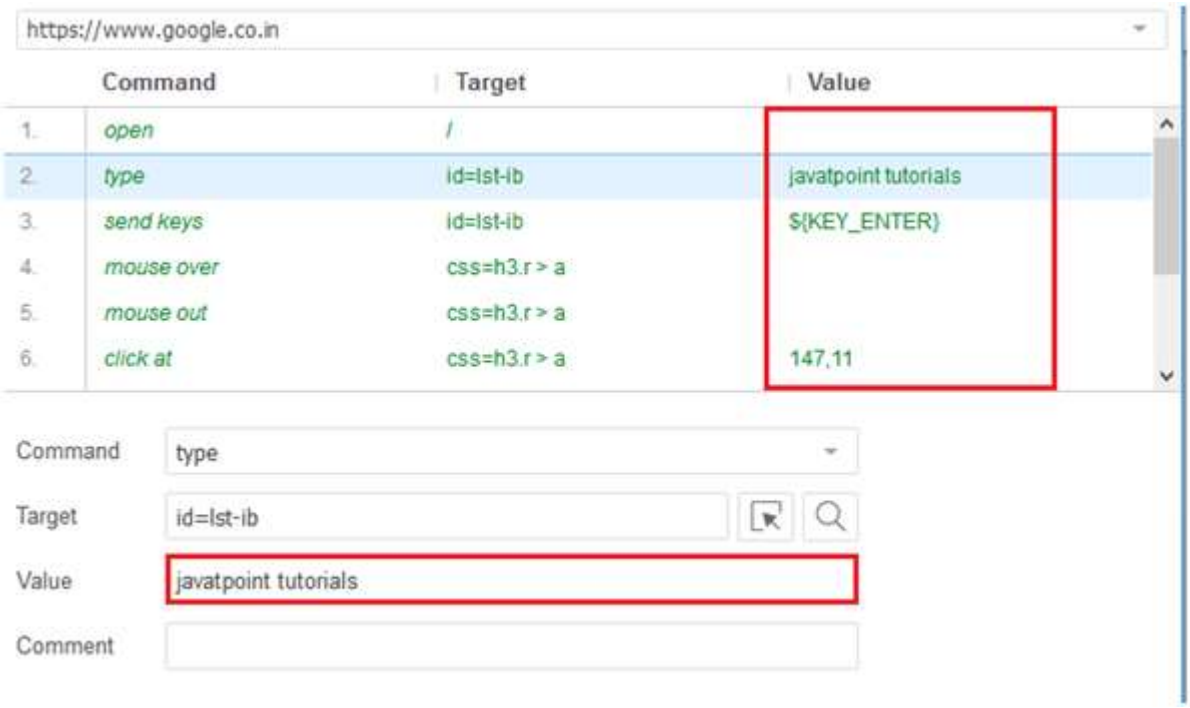
Target specifies the web element on which the operation has to be performed along with a locator attribute. For instance, if you are clicking on a button called javaTpoint, then the target link will be 'javaTpoint'.



Value:

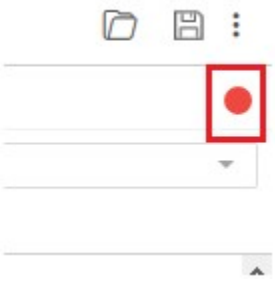
Value is treated as an optional field and can be used when we need to send some actual

parameters. For instance, if you are entering the email address or password in a textbox, then the value will contain the actual credentials.



6. Start/Stop Recording Button

Record button records all of the user actions with the browser.



7. Log, Reference Pane

The Log Pane displays the runtime messages during execution. It provides real-time updates of the actions performed by the IDE. It can be categorized into four types: info, error, debug and warn.