**UNIT-1**

**PROJECT MANAGEMENT**

Project:- Project exist to bring about a product or service that hasnt existed before. Project are temporary in nature, while operations are going on. Projects have definitive start and definitive end dates. The project is completed when the goal and objectives of the project are accomplished and project is cancelled.

Characteristics of project

1. Projects are unique.
2. Projects are temporary in nature and have a definite beginning and ending date.
3. Projects are completed when the project goal are achieved.
4. A successful project is one that meets or exceeds the expectations of your stakeholders.

Project Management:- Management is a very necessary activity when computer based systems and events that occur as software develops from a preliminary concept to an operational implementation. Every one manages to some extent, but the scope of management activities varies with the person doing it. Software engineers manages their day to day activities ,planning, monitoring and controlling technical task. Project manager plan, monitor and control the work of a team of software engineers.

1. THE PROJECT MANAGEMENT SPECTRUM:

Computer software has a profound effect in every aspect of our life and has become pervasive in commerce, our culture and mumndane matters. As the software importance has grown over years today software industry has become a dominant factor in the industrialized economics. Eventually it has become important to manage big and complex software project. Software Project management involves activities such as planning, monitoring and controlling people and processes that occurs as the software evolves from conceptual stage to an implementation stage. Software development is a complex activity which involves many people working together for a long period. Hence it becomes necessary to manage such software projects efficiently and effectively. A software manager has to understand and organize the four Ps ---People, Product, Process and Project for an effective project management.

1. The People

Smart people are the most important ingredient in the software process as they are organized to perform effective software engineering. The software project is populated by stake holders who can be categorized people involved in every software project are categorized into five constituencies.

1. Stakeholders
2. Structure of the Development Team
3. The Software Team
4. Communication and coordination
5. Team leaders
6. Stakeholders

Stakeholders are the people who have something to either gain or lose as a result of the project. Some stakeholders are project managers, sponsor, customer, board of director manager, suppliers.

* 1. Senior Manager: Who define the business issues that often have significant influence on the project.
  2. Project (Technical) Manager who must plan ,motivate, organize and control the practioners who do software work.
  3. Practioners : Who deliver the technical skills that are necessary to engineer a product or application.
  4. Customer : Who specify the requirement for the software to be engineered and other stakeholder who have a peripheral interest in the outcome.
  5. End users: Who interact with the software once it is released for operational implementation.

There are two main consideration in managing people involves in project development.

* Structure of development team.
* Communication and coordination between team members.

1. Structure of Development Team

For the smooth development of software project,it is important that the team is dedicated and the values team goal more than individualistic goals.

Every organization has a different structure for a software development team. It is customary for an organization to deploy the following three types of generic team structure.

* + 1. Democratic decentralized(DD)
    2. Controlled decentralized
    3. Controlled centralized

1.Democratic decentralized (DD)

The DD team structure has no permanent leader. Different coordinators are appointed to manage different tasks for short duration. Each coordinator is assigned to a team. The leadership style is participative. These team members communicate directly among themselves in a horizontal manner.DD team structure

is useful when all team members are efficient and experienced and do not need to report to a senior manage and when a project is of short duration. This team structure ensures independence among team members and effective performance

of duties

2. Controlled decentralization (CD)

It follows a hierarchical structure that contains groups and subgroups. The team has a permanent leader who manages the main activities in the team and secondary leaders that manage subtasks. Problems are managed at group level. This structure is useful when the project is expected to spread over a long period, such as a year or more. Individuals can be hierarchically grouped according to their skill level and appoint the person with the highest skill level as the permanent leader.

3.Controlled Centralized (CC)

This structure has a team leader who handles top level problem solving and internal team coordination issue. Communication between the team leader and development Team occurs in vertical manner. This is the most common type of team structure and is deploys in all types of software projects. This structure is useful for small projects with a team size of ten or less members so that a single team leader can manage the entire team

1. There are almost as many human organizational structures for software development as there are organizations that develop software. The best team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels and the overall problem difficulty.
2. Communication and coordination between team members.

Effective communication and coordination between team members is critical for the success of any software project. Software projects are often complicated by following factors:

Large size of project, uncertainty, interoperability i.e. new software must be compatible

With earlier version of the same product and comply with constrain imposed by the system. To deal with them effectively there must be proper mechanism for formal and informal communication between team members. Formal communication is carried out through the exchange of formal documents, structured meeting and other no interactive channels where as informal communication is more personal which includes exchange of ideas on ad-hoc basis or ask for help as problem arise.

The following approach can be adopted to coordinate a project among team members:

1. Formal and interpersonal approach: It stresses the importance of quality assurance activities involving status review meeting. Team members have a frequently submit reports containing suggestions for improving the software product.

ii. Informal and interpersonal approach: In this structure group meetings are organized for informal discussion, problem solving and requirement specification. This is used to discuss and resolve the problem encountered during the development of the software.

iii. Interpersonal Networking approach: In this approach, informal discussion are conducted with team members and people not involved directly with the development team, such as members of other teams. This approach is used when some members who are not part of the development team are made responsible to monitor the progress of the software product being developed.

1. Team Leaders:- Project management is a people intensive activity and for this reason, competent practitioners often make poor team leaders. The model of leadership is motivation and ability to encourage (by push or pull)technical people to produce to their best ability. The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.
2. Product

The final thing that can be delivered of a software project is a product, which is also called a software solution. A software project is planned and defined to produce a software project. At the beginning of the project, a detailed analysis of the proposed project and the need of the client should be done. The software product design is based on their client needs.

Studying product details and requirement specification helps to make quantitative estimate about the effort required to successfully complete a project. In addition, studying product details also helps you to analyze and define the scope of software product includes :

* Context : The integrated of the final product to the larger product or business context, including constraints.
* Information objectives : The final information expected as output from final final product.
* Functions : The functions to be performed by the software to transform input to output.
* Performance : These includes special performance requirement of the software to be developed.

In addition to this, the project should be broken down into workable components. This process is known as problem decomposition. Decomposition is basically done in two areas:-

1. The functionality of the software
2. The process that will be used in developing the software.

After decomposition, cost and efforts estimates are quantified according to the components

1. Process

All software processes undergo the same generic phases, such as analysis, design. development, testing, implementation and support. Process model used in development process may vary depending on the software project. As a project manager may choose the process model most appropriate for the project by taking into consideration the customers needs and the environment in which the development team works.

A process model is selected after the development team defines the common activities it will perform. Processes are then decomposed to deliver small and manageable tasks. Process decomposition starts with the definition of processes that need to be executed to develop the project. The process might be decomposed into following tasks

* Communication
* Planning
* Modeling
* Construction
* Deployment

These are applied to software engineering work tasks (e.g different product functions)

These activities break the process into smaller pieces, which are easier to execute and manage. Different process models are available like sequential, prototyping ,spiral,project manager must decide about which model to use depending on

* Customer who have request the product
* People who would work on project
* Product characteristics
* Project environment

Project planning begins once model is selected.

4.Project

The final step in project management is to formalize the project and coordinate processes and people to develop the software product.

To successfully manage a software ,project manager must understand the potential problems that occur so as to ensure that the project is effective. The main problem that can lead to the failure of software projects are

* Poorly defined requirements
* Poor grasp of requirement by the developer
* Poor scope definition
* Poor change management
* Unrealistic deadlines
* Unskilled team members

To avoid these problems, the following measures can be taken :

* Set targets: Understand the problems and requirements completely, then set target for very one involved in development process. Select people with relevant skills and provide them with the resources they need to accomplish their goals.
* Maintain development pace: To maintain pace, one should try to develop strong relationships with the staff so that they see the project through to completion. Introducing new people in the middle of the development process reduces the development pace.
* Track the progress : Monitor the activities and asses requirements specifications and test cases. Progress can be tracked as a part of the quality assurance activity.
* Make effective decisions : make timely decisions to identify and avoid obvious risks. Allocate extra time to perform risky tasks. Use existing software components wherever possible to finish the task on time.
* Conduct post mortem analysis: Compare the performance against planned schedules and estimates. In addition, obtain feedback from customer as well as developers and record it for future reference.

(2)THE ROLE OF PROJECT MANAGER AS A PLANNING AGENT

A project manager is usually responsible for the success or failure of the project. Project Manager is the god of his project and he is the one who decide the success of project.

An efficient and smart project manager should possess the following skills:

1. Well Organized:- B y definition a project manager needs to be well organized. They need to be able to determine what needs to be done, in what order, so as to achieve a desired outcome.
2. Focus & vision:- It is vitally important for a project manager to have a clear picture and to be able to quickly envision what needs to be done to achieve the goal of the project.
3. Good communicator:-The project manager should be able to communicate effectively with all levels inside and outside of the organization. He should be able to negotiate fairly and effectively with the customer. He should be able to convey ideas and information clearly and concisely to all team members.
4. Calm under pressure:- A project manager must remain calm under problematic situations and focus on what is important than getting hysterical.
5. Quick learner:- A successful project manager must be moving on to larger and more complex projects.
6. Problem solving skills:- A successful project manager must look and analyze problems from all angels and have an excellent problem solving ability.
7. Controlling :- To control the project, the project manager implements a management information system designed to track actual progress and compare it with planned progress.
8. Leading :- Project manager should clearly define roles, responsibilities and performance expectations for all his team members.
9. Self management functions:- The project manager should be able to maintain focus and control when faced with uncertainty and should be able to show consistency among principal, values and behavior.
10. Customer awareness function:- Project manager should be able to anticipate customer needs effectively and proactively to satisfy them. He should be able to accurately translate the customer verbalized wants into what they actually needs.

RESPONSIBILITIES:-

1. The project manager is the person responsible for managing the project.
2. He is also responsible for the outcome of the project.
3. The project manager is involved with planning, controlling and monitoring and also managing and directing the assigned project resources to best meet project objectives.
4. The project manager controls and monitor triple constrain- project scope, time and cost in managing competing project requirement.
5. The manager is responsible to the project stakeholder for delivering a projected objective within scope schedule, cost and quantity.
6. The project manager is responsible for identifying, monitoring and responding to risk.
7. Quality management means taking care of quality of the process in question such that it meets various quality parameters set earlier.
8. Procurement management: - Various materials needed during the project need to be procured and manage with vendors and suppliers for successful completion of the project.
9. PROJECT ESTIMATION
10. INTRODUCTION

Software is the most expensive element of virtually all computer based systems. For complex custom system a large cost estimation error can make the difference between profit and loss. Cost overrun can be disastrous for the developer. Software cost and effort estimate will never be an exact science. Too many variables human technical, environmental, political can effect the ultimate cost of software. To achieve reliable cost and effort a number of options arise. Software project management begins with asset of activities that are collectively called project planning. Before the project manager and the software team must estimate the work to be done, the resources that will be required and the time duration.

Software project planning encompasses five major activities- estimation. Scheduling and change management planning.

THE PROJECT PLANNING PROCESS

Planning is to decide in advance. The objective of software project planning is to provide a framework which would enable the manager to make reasonable estimate of resources, cost and schedule. In spite of the uncertaint , the manager must ensure that a project activities are planned in advance.

Task Set for Project Planning

1. Establish project scope
2. Determine feasibility
3. Analyze risks
4. Define required resources
5. Determine human resources required
6. Define reusable software resources
7. Identify environment resources
8. Estimate cost and efforts
9. Decompose the problem
10. Develop two or more estimates using size,function points,process task,or use-cases
11. Reconcile the estimates
12. Develop a project schedule
13. Establish a meaningful task set
14. Define a task network
15. Use scheduling tools to develop a timeline chart
16. Define schedule tracking mechanisms

ESTIMATION

A software project estimate predicts the size of the new project or the amount of efforts required to complete a project. There are several project estimation techniques to assist a project manager in making project estimation. Estimations begins with a description of the scope of the project. The problem is then decomposed into a set of smaller problems, and each of these is estimated using historical data and experience before a final estimate is made.

SOFTWARE SCOPE AND FEASIBILITY

Software scope describe the functions and features that are to be delivered to end users,the data that are input and output.

Scope is defined using one or two techniques;

1. A description of software is developed after communication with all stakeholders.
2. Functions and features to be delivered to end user.
3. content presented to user as they use the software.
4. Performances considerations(processing and response time etc)
5. Constraints(limits placed on software by external hardware, available memory, or existing systems)
6. Input and output data
7. Interfaces and reliability that bound the system
8. A set of use-cases is developed by end user. A use-case is a scenario-based description of the users interaction with the software from the users point of view.
9. Once scope is understood, the software team and others must work to determine if the project feasible or not.

SOFTWARE ESTIMATION METHODS:

To achieve reliable cost and effort estimates a number of options arise:

1. Estimation can be done until the project starts.
2. Estimation can be done on the basis of past similar projects.
3. Usage of simple decomposition techniques to generate project cost and effort estimates.
4. Usage of one or more empirical models for software cost and effort estimation.

The first two option however attractive is not practical. The second option is suitable if the current project is quite similar to past efforts and other projects. Unfortunately past experience is not always a good indicator of future results. Decomposition techniques take a divide and conquer approach to software project estimation. By decomposing a project into major functions related project development activities, cost and estimation can be performed modularly. Result arrived at by Empirical estimation models can be used as cross reference to decomposition technique and offer potentially valuable estimation approach.

1. DECOMPOSITION TECHNIQUES

The decomposition approach was discussed from two different points of view

* Decomposition of the problem
* Decomposition of the process

But before an estimate can be made the project planner must understand the scope of the software to be built and generate an estimate of its size

1. Software sizing :- Software size refers to the length of software programs

Sizing represents the project planners major challenge. If a direct approach is taken size can be measured in LOC. If an indirect approach is chosen size is represented as FP.

Four different approaches to the sizing problem as proposed by Putnam & Myers:

* Fuzzy logic sizing:- To apply this approach, the planner must identify the type of application, establish its magnitude on a qualitative scale and then refine the magnitude within the original range.
* Function point sizing:- The planner develops estimates of the information domain characteristics. Any software has to perform basically two functions i.e. to accept data and to manipulate it. So functions are classified under five heads:
* Data Functions:
* Internal logical files:- These are the logical grouping of data inside the application boundary.
* External interfaces files:- These files reside in some other application and interact with the application.
* Transactional Functions:
* External inputs:- These files affect the internal logical files. Users interact with the system through these files.
* External outputs:-They provide reports,screens to the end users
* External inquires:- They are external online enquires and online responses of the system.

The functions of the system are affected by 14 information domain characteristics like Backup and recovery, data communication, online data entry, performance critical, code design for reuse etc.

* Standard components sizing:- Software is composed of a number of different standard components that are generic to a particular application area. For example the standard components for an information system are subsystems, ,screens ,reports ,interactive programs ,batch programs ,files ,LOC and object level instructions. To illustrate consider an information systems application. The planner estimates that 20 reports will be generated. Historical data indicates that 600 lines of v.b are required per report. This estimates and computation are made for other standard components.
* Changing sizing: This approach is used when a project encompasses the use of existing software that must be modified in some way as part of a project. The planner estimates the number and type (e.g. reuse, adding code, changing code, and deleting code) of modification that must be accomplished. Using an effort ratio for each type of change the size of the change may be estimated.

2. Problem Based Estimation

LOC and FP data are used in two ways during software project estimation.LOC and FP estimation are distinct estimation technique. Yet both have a number of characteristics in common. when LOC is used as the estimation variable, decomposition is absolutely essential and is often taken to considerable levels of detail. The greater the degree of partitioning, the more likely reasonably accurate estimates of LOC can be develop.

For FP estimates decomposition works differently. Rather than focusing on function, each of the live information domain characteristics as well as the 14 complexity adjustment values discussed in are estimated. The resultant estimates can then be used to derive a FP value that can be tied to past data and used to generate an estimate.

* The project planner begins with a precise statement of software scope and from this statement attempts to decompose software into problem function that can each be estimated individually.
* LOC and FP is then estimated for each function
* Historical data are then applied to the appropriate estimation variable and cost or effort for the function is derived.
* Function estimates are combined to produce an overall estimate for the entire project.

Once the expected value for the estimation variable has been determined, historical LOC or FP productivity data are applied. Are the estimates correct? The only reasonable answer to this question is we cant be sure. Any estimation technique, no matter how sophisticated must be cross-checked with another approach. Even then common sense and experience must prevail.

FB-BASED ESTIMATION

Decomposition for FB based estimation focuses on information domain values rather than software functions. It is known as indirect method for software sizing.

Benefits of FB- BASED estimation

1. It is independent of computer language.
2. It is useful for discussion with the end user to understand his/her requirements.
3. It is useful in data oriented application.
4. It is accurate than LOC method.

Referring to the function point calculation presented in table , the project planner estimates inputs, outputs ,enquires, files and exte4rnal interfaces for a software. For the purpose of this estimate the complexity weighing factor is assumed(3-15). The following Table presents the result of this estimate.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FUNCTION TYPES | SIMPLE | AVERAGE | COMPLEX | TOTAL |
| INPUTS | 1 X 3 | 1 x 4 | 1 x 6 | 13 |
| OUTPUTS | 1 X 4 | 1 x 5 | 1 x 7 | 16 |
| INQUIRES | 2 X 3 | 2 x 4 | 2 x 6 | 26 |
| INTERNAL FILES | 1 X 7 | 1 x 10 | 1 x 15 | 85 |
| EXTERNAL FILES | 0 X 5 | 0 x 7 | 0 x 10 | 0 |
| TOTAL | UNADJUSTED TOAL | | | 140 |

The complex internal factors are estimated and the complex adjustment factor is computed. The value to each factor ranges from 0 to 5 (0- no influence.1-incidental,2- average,3-moderate,4- important,5-essential)

Internal Factors estimated

Complex internal processing = 3

Code to be reusable =2

High performance =4

Multiple sites =3

Distributed processing =5

---------

Project adjustment factor =17

----------

* FP estimated=count-total X [.65+(.01XO(F))]

. =140X[.65+.01X17]

. =114

But how long will the project take and how much will it costs?

* Programmers in the organization are estimated to complete an average 9 function points per month. Thus

114 FP divided by 9= 13 man months

* If the average programmers is paid $3000 per month(including benefits), then the [labor] cost of the project will be

13 man-months x $3000=$39000

LOC Based Estimation

LOC Based Estimation

LOC estimation method is said to be a direct method as it estimates in terms of number of lines of code to be developed. It is relatively easy and simple to develop and suitable for smaller applications.

Limitations of LOC method

1. Stating the size in LOC is less accurate than FP method
2. It is language dependant.
3. All the lines of codes are of not same complexity.

LOC method used for a CAD software can be explained as follows:

Initially a bounded scope of statement is prepared from the requirement specification list.

* A s/w package has to be develop for CAD application
* The CAD s/w will accept three dimensional geometric data from the user through an interface
* All the geometric data and supporting information will be in the CAD database.
* Design analysis module will be developed for producing required output which will be displayed on various graphic devices.
* The s/w will be designed to control and interact with peripheral devices like mouse, digitizer, laser printer and plotter.

List of software functions from the above scope would be:

* User interface and control facilities.
* Two dimensional geometric analysis.
* Three dimensional geometric analysis.
* Database management
* Computer graphic display facility
* Peripheral control function
* Data analysis models

A range of LOC estimate is established for each function.

Ex : for 3DGA

Optimistic-4600

Most likely-6900

Pessimistic-8600

After applying the equation

S=(S opt + 4S m + S pess)/6

The expected value of 3DGA will be 6800 LOC

The rest of estimates for listed functions is done in the similar fashion

So we arrive at the following estimate

|  |  |
| --- | --- |
| User interface and control facilities [UICF] | 2300 |
| Two dimensional geometric analysis [2DGA] | 5300 |
| Three dimensional geometric analysis [3DGA] | 6800 |
| Database management [DBM] | 3350 |
| Computer graphic display facility [CGDF] | 4950 |
| Peripheral control function [PCF] | 2100 |
| Data analysis models [DAM] | 8400 |
| Estimated lines of code (Total) | 33200 |

Finally computation of cost and effort required is

Historical data indicates that:

Average productivity of system

=620LOC/pm

Labor rate=$8000/month

Cost/LOC =$ 13

Total estimated cost=$431600

Estimated effort=54 pm.

4) PROJECT SCHEDULING

1. INTRODUCTION

In project management, a schedule consists of a list of a projects terminal elements with intended start and finish dates. Terminal elements are the items that are estimated in terms of resources, requirement budget and duration. Project schedule evolve over time

1. NEED FOR PROJECT SCHEDULING:

The following are the reasons for project to be scheduled:-

* To identify tasks and their relationship to one another-this is important for risk management, staffing and sequencing.
* To provide the picture of what has happened(actual) and what is now planned.
* To forecast resources requirements and provide a visual representation of task dependencies
* A project schedule gives everyone on the team insight into where the project is going and how their efforts impact outcomes.
* A project schedule helps keep track of accomplishments, needs and compliance with requirement.

1. TASK OF PROJECT SCHEDULING

Proper scheduling requires that-

* All tasks appear in the network.
* Efforts and timing are intelligently allocated to each task.
* Interdependencies between tasks are properly indicated.
* Resources are allocated for the work to be done.
* Closely spaced milestones are provided so that progress can be tracked.

1. BASIC CONCEPTS

Although there are many reasons why software is delivered late, most can be traced to one or more of the following root causes:

* An unrealistic deadline established by someone outside the software engineering group and forced on managers and practitioners within the group.
* Changing customer requirement that are not reflected in schedule changes.
* Risks that were not considered when the project commenced.
* Technical difficulties that could not have been foreseen in advance.
* Miscommunication among project staff that results in delays.
* Lack of action to correct the problem.

1. BASIC PRINCIPLES

The goal of software project schedule is to determine how a software project schedule is created. To create a project schedule, first grouping of similar activities together is done followed by determining the dependencies of different activities. Next is the allocation of the estimated time and resources

to each activity, definition of the roles and responsibilities and the output and validation criteria is done. 

Some of the guidelines for creating a software project schedule are discussed below.

1. Classification :- While managing a software project, we need to group similar tasks and the WBS and decomposition technique. Using these tools, we can divide a software project into different phases. The phases can be further subdivided into activities. The software project schedule is prepared according to the arrangement of the phases.

b) Interdependence:- A Software project is composed of multiple phases and each phase is composed of multiple activities. Although each activity is treated separately, it is linked to other activities. As project manager can determine the interdependence and sequence of activities. For example some activities can be completed without any input from other activities, whereas other activities cannot start unless a preceding activity is completed.

1. Time and Effort Allocation

Each activity in a software project needs a certain amount of time and effort for completion. To manage the project, project manager assign start and end dates to each activity, allocates appropriate effort to each activity. Most software projects operate with time and effort constraints. Therefore managing within the available resources is very important for a software project manager.

1. Validation criteria:- Determining the validation criteria allows ensuring that the optimal level of resources is available for a particular activity. Suppose 5 people are assigned to an activity that requires an effort of 3.5 person days. This means resources allocated are more than the actual requirement.
2. Defined Responsibilities and Outputs

A software project manager assigns roles and responsibility to all people in a software project. This defines the hierarchy in the development team. He also defines the outputs from each activity. This helps in identifying the result expected at the end of every activity. Linking the role to the output, each persons effort and the progress of each activity can be tracked.

PEOPLE VS. EFFORTS IN PROJECT MANAGEMENT

It has been observed that in a small software development project a single person can analyse requirement,perform design,generate code and conduct test. As the size of a project increases more people must be involved. Adding people late in a project often has a distribution effect on the project causing schedules to delay because the people who are added must learn the system

+ and the people who teach them are the same people who were doing the

work. While teaching , no work is done and the project fall further behind.

In addition to the time it takes to learn the system, more people increase the number of communication paths and the complexity of communication throughtout a project. Although communication is absolutely essential to successful software development, every now communication path requires additional effort and therefore additional time.

Over the years, it has been analysed that project schedules are elastic. This has two implications

1. The project completion period can be reduced by adding more resources -people and money.
2. Project completion date can be extended by reducing the number of resources.

The Putnam-Norden-Rayleigh (PNR) Curve provides an indication of the relationship between efforts applied and delivery time for a software project. A version of the curve representing project efforts as a function of delivery time, is shown in Figure below.

Ed

Ep

to td  tp

The implication here is that delaying project delivery can reduce costs significantly but this must be weighed against the business coast associated with the delay. AS the project time is extended from td to tp the effort can be reduced from Ed to EP But any further increase in the efforts does not compress the project completion time below tO which is the optimum time for project completion. The software equation is derived from the PNR curve demonstrates the highly nonlinear relationship between time to complete a project and human effort applied to the project. The number of delivered lines of code (source statements), L is related to effort and development time by the equation

L = P X E 1/3 t 4/3

Where L is the number of lines of code delivered, E is development effort in person-months, P is a productivity parameter that reflects a variety of factors that lead to high-quality software engineering work and t is the project duration in calendar months.

Thus the PNR curve helps in determining the minimum cost time required to complete a project irrespective of the resources applied in the project.

1. EFFORT DISTRIBUTION

Each of the software project estimation techniques leads to estimates of work units(e.g. person-months) required to complete software development. A recommended distribution of effort across the software process is often referred to as the 40-20-40 rule. Forty percent of all effort is allocated to front-end analysis and design.

Front end activities involve

* Customer communication
* Analysis
* Design
* Review and modification

A similar percentage (40% ) is applied to back-end testing which involves activities like

* Unit testing
* Integration testing
* Validation testing
* System testing

20 percent of effort is conferred to construction activity like

* Coding

This effort distribution should be used as a guideline only.

The characteristics of each project can have the distribution of efforts in the following manner .

* Work expended (consumed) on project planning accounts for more than 2-3 percent of effort.
* Requirement analysis may comprise 10-25 percent of project effort.
* A range of 20 to 25 percent of effort is normally applied to software design.
* A range of 15 -20 percent of effort for coding can be achieved.
* Testing and subsequent debugging can account for 30  40 percent of software development effort. The criticality of the software often dictates the amount of testing that is required.

1. DEFINING A TASK SET FOR THE SOFTWARE PROJECT

The process models offer different paradigms for software development. Regardless of whether a software team chooses a linear sequential paradigm, an iterative paradigm, an evolutionary paradigm, the process model is populated by a set of tasks that enable a software team to define, develop and ultimately support computer software.

An effective software process should define a collection of task sets, each designed to meet the needs of different type of projects.

A task set is a collection of software engineering work tasks , milestones and deliverables that must be accomplished to complete a particular project. The task set to be chosen must provide enough discipline to achieve high software quality. But at the same time, it must not burden the project team with unnecessary work.

There are various factors affecting the task set like

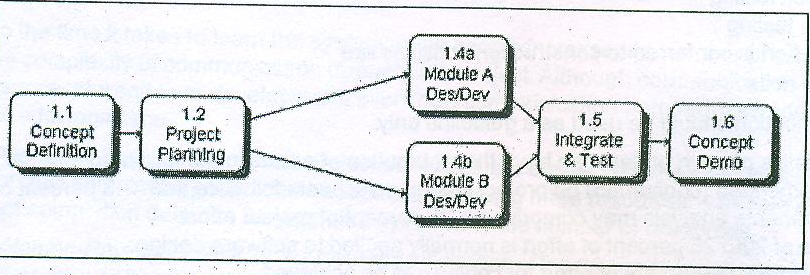
* Users of the product
* Size of the development team.
* Financial resources set aside for the project.
* Project completion date decided.
* Users communication etc.

Task sets are designed to accommodate different types of projects. Most software organization encounter the following projects :

1. Concept development projects that are initiated to explore some new business concept or application of some new technology.
2. New application development projects that are undertaken as a consequences of a specific customer request.
3. Application enhancement projects that occur when existing software under goes major modifications to function, performance or interfaces that are observable by the end-user.
4. Application maintenance projects that correct, adapt or extend existing soft ware in ways that may not be immediately are useful to the end user.
5. DEFINING A TASK NETWORK

A task network also called as a activity network is a graphic representation of task flow of a project. As any project encompasses number of tasks, there is a possibility that individual tasks and subtasks depend upon each other depending upon their sequences. Also if number of persons is more, these tasks can be performed in parallel. In such a case, current task must be completed first before one starts with the next task that needs the results from previous task. Thus there must be coordination between tasks. This is where the task networks are used. 

Following figure shows a systematic task network for a concept development project.



When more than one person is involved in a software engineering project,it is likely that development activities and task will be performed in parallel. When this occurs concurrent tasks must be coordinated so that they will be complete when later tasks require their work products. A task network also called activity network is a graphic representation of the task flow for a project.

1. NETWORK SCHEDULING TECHNIQUE

To plan the activity in a project the project manager can also use network scheduling technique. This technique use network schedule to trace the

completion of pre determine activities. There are two basic network scheduling techniques.

PERT AND CPM . Using PERT and CPM help to complete the project on time. By using this technique one can determine the latest time by when an activity should start so that it should be completed on time.

There are 2 form of activity network diagrams.

1. Project Evaluation and Review Technique :- It is a project management tool use to schedule organise and coordinate task within a project. It is develop by U. S. Navy in 1950 to manage the polarise submarine missile programme. PERT is organise by events and activities or tasks. PERT can by both cost and time management system.

* PERT is design for research and development project when activity completion time are uncertain.
* Each chart starts with an initiation node from which the first task or tasks originates.
* If multiple task began at the same time they are all started from the node or branch or fork out from the starting point.
* Each task is represented by line which states its name or other identifier, its duration, the number of people assigned to it.
* Other end of the task line is terminated by another node, which identifies the start of another task or the beginning of any slack time. i.e. waiting time between tasks.

Steps in drawing a PERT chart.

* Make a list of project tasks.
* Assign a task identification letter to each task.
* Determine the duration of each task.
* Draw the PERT network , number each node, liable each task, connect each node from state to finish and put each tasks duration on the network.
* Determine dummy task if any.
* Determine the earliest and latest completion time.
* Verify PERT for correctness.

Benefits of PERT

1. PERT network is continuously use ful to project managers prior to and during a project.
2. The PERT network is straightforward in its concept and is supported by software.
3. The PERT networks graphical representation of the project task help to show the task interrelationships.
4. The PERT networks ability to highlight the project critical path.
5. The use of PERT network is applicable in a wide variety of projects.
6. PERT controls time and cost during the project to have right balance between completing the project on time and completing it within the budget.
7. It allow scheduling and simulation of alternate schedule .

Limitation of PERT

1. PERT does not help in deciding which activities are necessary and how long each will take.
2. The PERT network is only as good as the time estimates that entered by project manager.
3. The PERT network does not deal very well with task overlap. It assumes the following task begin after their preceding task end.
4. The project task and their relationships have to be clearly define in order to make PERT network useful.

Critical Path Method (CPM)

CPM acts as a basis both for preparation of schedule and resource planning. They were develop in 1950 to control large defiance projects. In a CPM chart the critical path is indicated. Critical path is the path of longest duration as determined on a project network diagram. The critical path determines the total duration of the project. If the task on the critical path is delayed the final completion of the project will likely be delayed. The critical path is critical because task that follow a critical task can not be started until all of the previous task on the critical path are completed. The critical tasks will have starting and finishing times that are fixed relative to the start of the project. Task not on critical path will usually have some flexibility relative to when they start and finish. This flexibility is called float or slack.

Float is difference between the time available for performing a task and time required to complete a task. An effective critical path analysis can make difference between success and failure of complex project.

Benefits of CPM

* It identifies the task that must be completed on time for the whole project to becompleted on time
* It also identifies which task can be delayed for a while if resources need to be reallocated to catch up on missed tasks.

Limitations of CPM

* The relation of tasks to time is not as immediately obvious as with Gantt Charts.
* These are more difficult to understand.
* CPM helps to minimize cost while still achieving your objective.

Example of PERT/CPM

Building of Production Plant Unit

Let there be plan a project like the building of the plant

List of activities will be

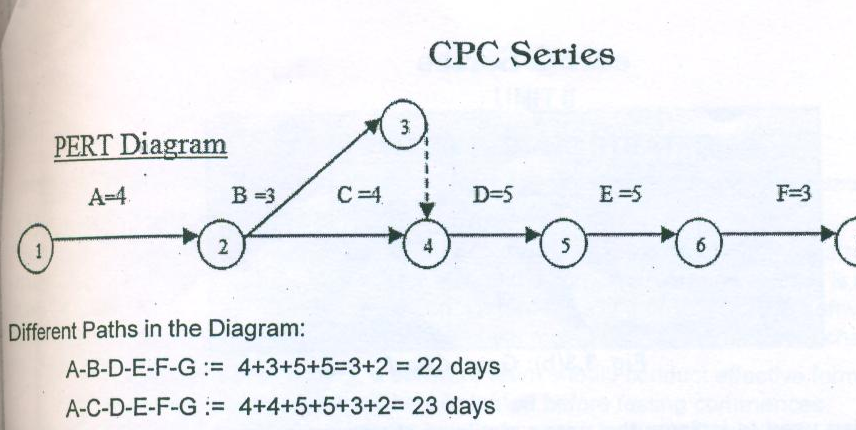
1. Activity A might be to dig the foundations.
2. Activity B might be to set the concrete base.
3. Activity C might be to build the plasterboard walls.
4. Activity D might be ventilation fittings
5. Activity E would be to paint the walls.
6. Activity F might be electrical fittings.
7. Activity G would be give the finishing touches and end the task.

There is the opportunity here that Activity C might be done at the same time as B

The activity project network table will be:

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Task label | Preceding Activity | Duration |
| Dig the foundation | A | - | 4 |
| Set the concrete base | B | A | 3 |
| Build the plasterboard walls | C | A | 4 |
| Ventilation fitting | D | B,C | 5 |
| Paint the wall | E | D | 5 |
| Electrical fittings | F | E | 3 |
| Finishing touches | G | F | 2 |

PERT Diagram



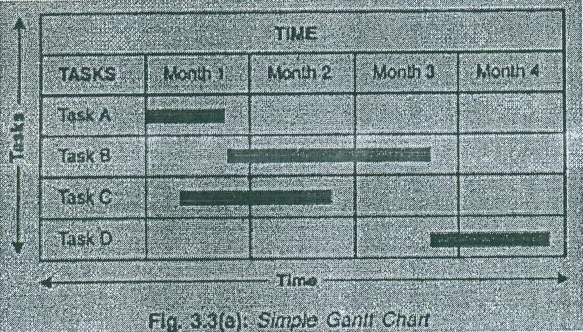
Therefore the critical path will be the path with maximum number of days=23 days.

Gantt Charts:- A Gantt Chart is a horizontal bar chart developed as a production control tool in 1917 by Henry.L.Gantt, an American engineer and social scientist.It is frequently used in project management.

A Gantt Chart is useful for tracking and reporting progress,as well as for graphically displaying a schedule. Gantt charts are often used to report progress because theyrepresent an easily understood picture of project status. However Gantt charts are not an ideal tool for project control.Purpose of Gantt chart is to present a project schedule that shows the relationship of activities over time. Gant chart are a project planning tool that can be used to represent the timing of task required to complete a project. Because Gantt chart are simple to understand and easy to construct,they are used by most project manager for all but the most complex projects.

Examples of Gantt Charts

Example 1: A simple Gantt Chart is shown in Fig 3.3(a)



Example 2: Typically Gantt charts indicate the exact duration of specific tasks,but they can also be used to indicate the relationship between tasks,planned and actual completion dates,cost of each tasks,the person or persons responsible for each task and the milestone in a projects development.Such a Gantt Chart is shown in fig 3.3(b).

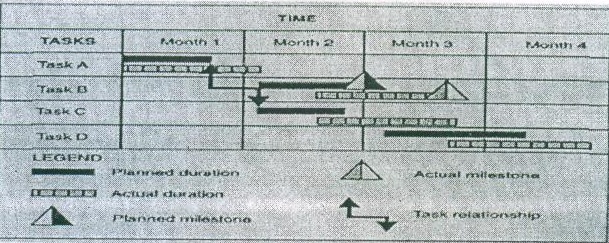
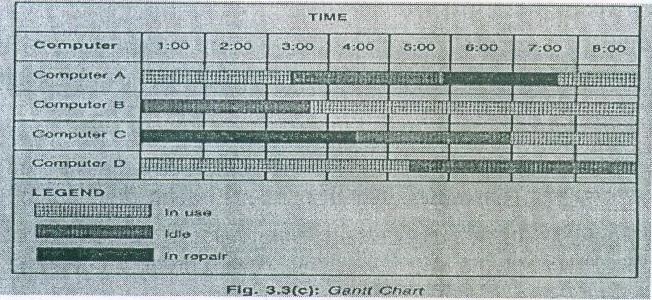


Fig 3.3(b): Gantt Chart

Example 3:

Gantt charts are also used to indicate the usage statistics of computer in a software organisation. Such a chart is shown in fig 3.3(c)



Reason to use Gantt Charts

Gantt Charts are conventional means of representing graphically schedules,appointments and events.Such charts help members of an organisation to plan activities better.

Information about WHAT is happening,WHEN is happening, and WHO will be present,is valuable to concerned people in the organisation.It is good because it plots activities as bars on a time scale. It indicates different Activities,Responsibilities,Estimated or planned activities start and target dates,Actual start and target dates.

It aids in:Scheduling activities,Coordinating activities,Evaluating progress,Reallocating resources in case of problems

Advantages of gantt Charts

* Provide an easy understandable overview of a project for those without any technical background
* Gantt charts are relatively easy to create and maintain.
* Helps in reflecting the status of each project task at any point of time.

Disadvantages of Gantt Charts

* It does not show task interrelationship.
* These are not ideal for project control.

**Unit II**

**SOFTWARE TESTING STRATEGIES**

STRATEGIC APPROACH TO SOFTWARE TESTING

Testing is a set of activities that can be planned in advance and conducted systematically. Software is tested to uncover the errors that occurred during the design and construction phase. Project manager adopts a systematic strategy for testing the software.testing basically is a set of activities that can be planned in advance and conducted systematically.For this reason software testing methods should be defined for the software process which must posses the following characteristics.

* To perform effective testing a software team should conduct effective formal technical reviews. By doing this many errors will be eliminated before testing commences.
* Testing begins at the component level and work outward towards the integration of entire computer based system.
* Different testing techniques are appropriate at different points in time.
* Testing is conducted by the developer of the software an independent test group.
* Testing and debugging are different activities but debugging must be accommodated in any testing strategy.

A strategy for software testing must accommodate low level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against customer require-ments. As the steps of the test strategy occur at a time when dead line pressure begins to rise,progress must be measurable and problems must surface as early as possible.

STRATEGIC ISSUES

The following issues must be considered if a successful software testing strategy is to be implemented:

1. State testing objectives explicity. The specific objectives of testing should be stated in measurable terms
2. Build robust software that is designed to test itself.
3. Use effective formal technical reviews as a filter prior to testing.
4. Conduct formal technical reviews to assess the test strategy and test cases themselves.
5. Develop a continuous improvement approach for the testing process.

1.VERIFICATION AND VALIDATION

Verification is the process of determining if software meets the conditions set forth at the beginning or during previous activities of the software development life cycle correctly.

These conditions are set forth in the system requirements which are formally documented.

Validation refers to a different set of activities that ensures that the software that has been built is traceable to customer requirement.verification typically involves reviews and meetings to evaluate documents,plans,code,requirement and specification and can be done with reviews,walkthroughs and inspection meetings. This is doing before coding phase to determine if the system is consistent,adheres to standards, uses reliable techniques and prudent practices and performs the selecte functions in correct manner. Thus verification is

Static testing.

Types of Verification

There are four types of verifications that can be applied to the various levels:

* Inspection
* Analysis
* Testing
* Demonstration.

Inspection :- Here the typical techniques include walkthroughs, formal reviews,peer reviews and formal inspections.

Analysis :- This includes mathematical verification of the test items, which can includes estimation of execution times and estimation of system resources.

Testing:- also known as white box or logic driven testing. The tester has the access to source code in order to help him in testing.

Demonstrations :- Also known as blank box or input/output driven testing. The input values are entered and the resulting output values are compared against the expected output values. Validation is the process of evaluating a system to determine whether it satisfies the specific requirement and performs functions for which it is intended and meets the organizations goal and user needs. It is traditional and is performed at the end of the system development. Validation typically involves actual testing and takes place after verifications are completed. Thus validation is dynamic testing.

Types of validation tests :-

1. Class test:- determines whether the data is numeric or alphabetic for example a drug number should contain only digits 0 to 9.
2. Sign test :- Determines whether the algebraic values of the number should be less than zero or greater than zero or equal to zero.
3. Reasonable test:-Reject the data that should never occurs
4. Sequence test:- Ensures that data entered must be in specific order.
5. Range test :-Determines that data falls between upper and lower limits.
6. Presence test:- ensures that data is present for future processing.
7. Data test:- to test that data is entered in specific format.
8. Combination test:- A certain group of data elements are present in same time and must be correct
9. Code test :- The data value must match one of the predetermined values.

Thus to conclude verification is checking whether we are doing correctly or not validation is checking whether we have done correctly or not.

2.ORGANISING FOR SOFTWARE TESTING

For every software project there is an inherent conflict of interest that occurs as testing begins. From psychological point of view,software analysis and design(along with coding) are constructive tasks. The software engineers analyzes models and then creates a computer program and its documentation. Like any builder the software engineer is proud of the product that has been built. When testing commences there is a subtle yet definite attempt to break the thing that the software engineer has built. From the point of view of the builder testing can be considered to be (psychologically) destructive. So the builder goes through lightly designing and executing test that will demonstrate that the programorks rather than uncovering errors. Unfortunately errors will be present. And if the software engineer doesnt find them,the customer will definitely find them upon using the software.

The software developer is always responsible for testing the individual units(components) of the program ensuring that each performs the function for which it was designed. In many cases the developer also conduct integration testing to complete the software structure. Only after the software architecture complete does an independent test group(ITP) become involved. The role of an independent test group is to remove the problems associated with tested software architecture(design/integration) testing. The developer and the ITG work closely throughout a software project to ensure that throughout tests will be conducted. While testing is conducted,the developer must be available to correct errors are uncovered.

The ITG is part of the software development project team in the sence that it becomes involved during analysis and design and stays involved throughout a large project. However in many cases the ITG reports to the software quality assurance organization thereby achieving a degree of independence that might not be possible if it were a part of the software engineering organization.

(2) A SOFTWARE TESTING STRATEGY FOR CONVENTIONAL SOFTWARE ARCHITECTURES

Initially system engineering defines the role of software and leads to software requirements analysis. Where the information ,function, behavior,performance, constraints and validation criteria for software are established. A strategy for conventional software testing may also be viewed in the context of the spiral shape.

Until testing begins at the vortex(shape of something rotating rapidly) of the spiral and contentrates on each unit(component) of the software as implemented in source code. Testing progresses by moving outward along the spiral to integration testing,where the focus is on design and the construction of the software architecture. Taking another turn outward on the spiral next is the validation testing where requirement established as part of the software requirements analysis are validation against the software that has been constructed. Finally is the system testing where the software and other system elements are tested as a whole.

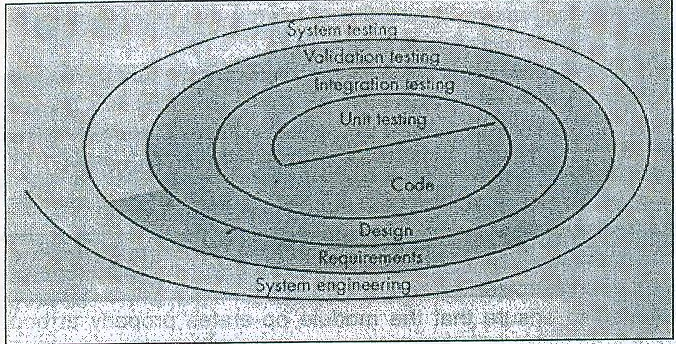
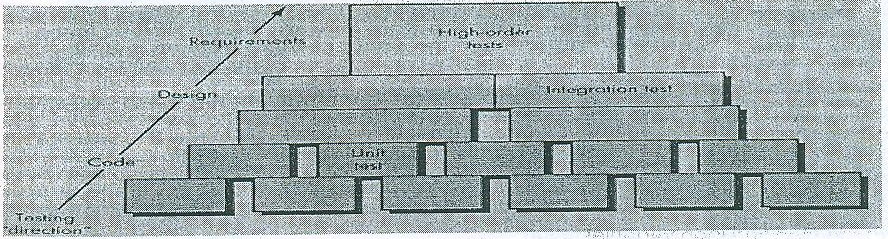


Fig :TESTING STRATEGY

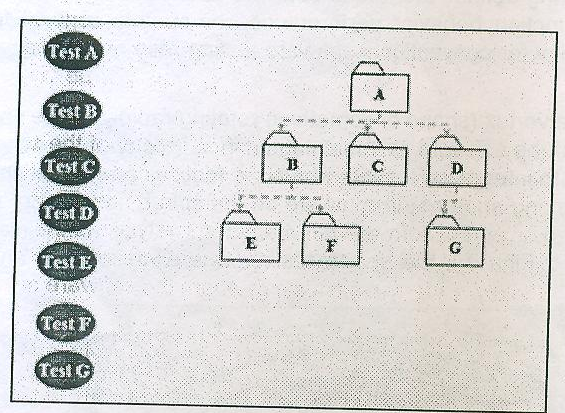
Considering the process from a procedural point of view,testing within the context of software engineering is actually a series of four steps that are implemented sequentially. Initially,test focus on each component individually,ensuring that it functions properly as a unit. Hense the name unit testing. Unit testing makes heavy use of testing techniques to ensure complete coverage and maximum error detection. Next components must be assembled or integrated to form the complete software package. After the software has been integrated(constructed) a set of high order tests are conducted. Validation criteria must be evaluated. Validation testing provides final assurance that software meets all functional,behavioral and performance requirements.

The last high order testing step falls the boundary of software engineering and into the broader context of computer system engineering. Software once validated must be combined with other system elements( e.g hardware,people,database) System testing verifies that all elements must properly and that overall system function/performance is achieved.



1. Unit Testing

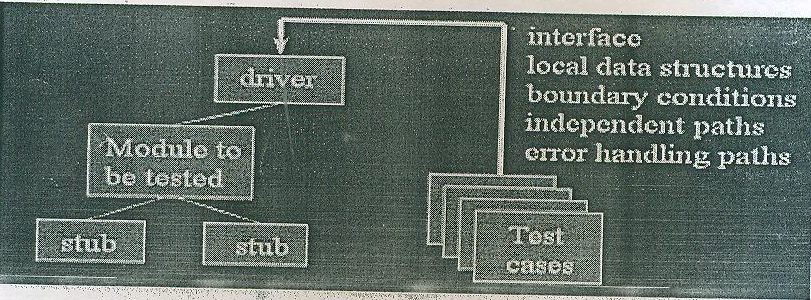
* In this testing each module is tested in isolation i.e without linking it with other modules.
* This enables to test and detect errors and coding in the module alone.
* The test cases used must cover each condition,range of values in the input and last condition in the iteration.



Unit test considerations

* The module interface is tested to ensure that information properly flows into and out of the program unit under test.
* Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithms execution.
* All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.
* Boundary conditions are tested to ensure that the module operates at boundaries established to limit or restrict processing.
* And finally,all error handling paths are tested.

Figure :- unit test considerations



Test cases should be designed to uncover errors due to erroneous computations,incorrect comparisons or improper control flow like:-

1.Comparison of different data types.

2. Incorrect logical operators

3. Incorrect comparison of variables

4. Improper or nonexistent loop termination.

5. Improperly modified loop variables.

Unit Test Procedure:

A component is not a stand alone program,driver and stub software must be developed for each unit test. In most application a driver is a main program that accepts test case data,passes such data to the components(to be tested), and print result. Stubs serve to replace modules that are called by components to be tested. A sub or dummy subprogram uses the subordinate module interface may do minimal data manipulation,provides verification of entry and returns control to the module undergoing testing. Drivers and stubs represent overhead. That is both are software that must be written but not delivered with the final software product.

Unit testing is simplified when a component with high cohesion is designed. When only one function is addressed by a components,the number of test cases is reduced and errors can be more easily predicted and uncovered.

1. Integration Testing

Integration testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing. The objectives is to take unit tested components and build a program structure that has been dictated by design.

* The entire system is viewed as a collection of subsystems.
* Goal: Test all interfaces between subsystems and the interaction of subsystems.
* The integration testing strategy determines the order in which the subsystems are selected for testing and integration.

Need for integration testing

* Unit tests only test the unit in isolation.
* Many failures result from faults in the interaction of subsystems.
* Without integration testing the system test will be very time consuming.
* Failure that are not discovered in integration testing will be discovered after the system is deployed and can be very expensive.

Big  bang Strategy

There is often a tendency to attempt non incremental integration that is, to construct the program using a big bangapproach. All components are combined in advance. The entire program is tested as a whole. A set of errors occurs. Correction is difficult because isolation of causes is complicated by the vast expanses of the entire program. Once these errors are corrected,new ones appear and the process continues in a seemingly endless loop.

**BIG-BANG APPROACH**

Test A

Test B

Test C

Test D

**Test A,B,C,D,E,F,G**

Test E

Test F

Test G

Incremental integration is the antithesis of the big bang approach. The program is constructed and tested on small increments, where errors are easier to islate and correct. Interfaces are most likely to be tested completely and a systematic test approach may be applied. A number of different incremental integration strategies are discussed below:-

Top-down Testing strategy

Modules are integrated by moving downward through the control hierarchy,beginning with the main control module(main program). Modules subordinate(and ultimately subordinate) to the main control module are incorporated into structure in either a depth first or breadth first manner.

Test the top layer or the controlling subsystem first.

Then combine all the subsystems that are called by the tested subsystem and test the resulting collection of subsystems

The integration process is performed in a series of five steps:

1. The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.
2. Depending on the integration approach selected i.e depth or breadth first subordinate stubs are replaced one at a time with actual components.
3. Tests are conducted as each component is integrated.
4. On completion of each set of tests another stub is replaced with the real component.
5. Regression testing may be conducted to avoid new errors.

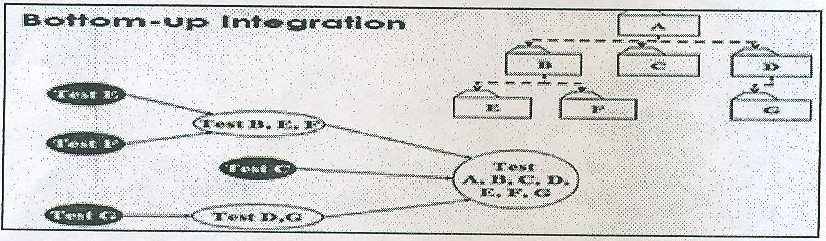
The most common of these problem with top down strategy are:

1. Delay in many tests until stubs are replaced with actual modules
2. Developing of stubs that perform limited functions that simulate the actual module.
3. Integration of the software from the bottom of the hierarchy upword.

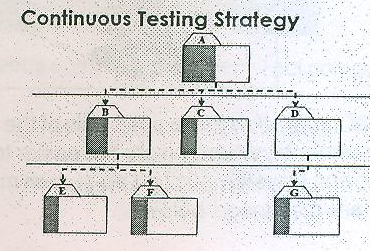
Bottom  up Testing Strategy

A bottom up integration strategy may be implemented with following steps:

* The subsystem in the lowest layer of the call hierarchy are tested individually
* Then the next subsystems are tested that call the previously tested subsystems
* This is repeated until all subsystems are included



Continuous Testing



Continuous build:

Build from day one

Test from day one

Integrate from day one

Regression Testing

This is the re execution of some subset of tests already conducted to ensure that the new changes do not unintended side effects.

* A representative sample of tests that will exercise all software functions
* Additional tests that focus on functions that are likely to be affected by the change.
* Tests that focus on software components that have changed.

(3) A SOFTWARE TESTING STRATEGY FOR OBJECT ORIENTED ARCHITECTURES

Object oriented architecture is a computer system design in which all identifiable components(files operations and processes) may be represented as data structure(objects) in the systems memory for manipulation by the system software. The overall strategy for object oriented software is identical in concept to the one applied for conventional architecture but differ in approach. We begin with testing in the small and work outward toward testing in the large. However our focus when testing in the small changes from an individual module(each unit)to a class that encompasses attributes and operations. As class are integrated into an object oriented architecture,a series of regression tests are run to uncover errors due to communication and collaboration between classes(components).

Finally the system as a whole is tested to ensure that errors in requirements are uncovered.

TEST STRATEGIES FOR OBJECT-ORIENTED SOFTWARE

The objective of testing is to find the greatest possible number of errors with a manageable amount of efforts applied over a given period of time. Although this fundamental objective remain unchanged for object oriented software,the nature of object oriented software changes both testing strategy and testing tactics.

OO Test Case Design

Conventional test case design are based on the process they are to test. OO test cases need to concentrate on the states of the class. To examine the states of the classes,the test cases need to follow appropriate sequence of operation in the class. Operation in classes can be tested using white box methods.

1. Unit Testing in the OO Context

When object-oriented software is considered the concept of unit changes. This means that each class and each instance of a class(object) packages attributes(data) and the operations(functions) that manipulate these data. An encapsulated class is usually the focus of unit testing. However operations within the class are smallest testable units. Because a class can contain a number of different operations and a particular operation may exist as a part of number of different classes the tactics applied to unit testing must change.

We can no lo9nger test a single operation in isolation(the conventional view of unit testing) but rather as part of a class. To illustrate consider a class hierarchy in which an operation X is defined for the superclass and is inherited by a number of subclasses.

Each subclass uses operation X but it is applied within the context of the private attributes and operations that have been defined for the subclass.Because the context in which operation X is used varies in subtle ways,it is necessary to test operation X in the context of each of the subclasses. This means that testing operation X in a standalone fashion(the conventional unit testing approach) is usually ineffective in the object-oriented context.

Class Level Methods:

Different types of methods in the class testing are:

1.Random testing: It is based on developing a random test sequence that tries minimum number of operations typical to the behavior(attributes) of the class.

2.Partition testing: This method categorizes the input and outputs in order to tests them separately. To categorise partition can be done based on attributes they use and states of a class.

3.Scenario-based testing: This form of testing concentrates on capturing the user actions and then simulating them.

1. Integration Testing in the OO Context

Because object-oriented software does not have an obvious hierchical control structure,traditional top and bottom-up integration strategies have little meaning. In addition integrating operations one at a time into class(the conventional incremental integration approach) is often impossible because of the direct and indirect interactions of the components that make up the class.

There are two different strategies for integration testing of OO systems

The first thread based testing integrates the set of classes required to respond to one input or even for the system. Each thread is integrated and tested individually. Regression testing is applied to ensure that no side effects occur.

The second integration approach use  based testing begins the construction of the system by testing those classes(called independent classes) that use very few classes. After the independent classes are tested the next layer of classes called dependent classes,which use the independent classes are tested. This sequence of testing layers of dependent classes continues until the entire system is constructed. The use of drivers and stubs also changes when integration testing of OO systems is conducted. Drivers can be used to test operations at the lowest level and for testing of whole group of classes. Stubs can be used in situations in which collaboration between classes is required. Clustering is one step in the integration testing of OO software. Here a cluster of collaborating classes is exercised by designing test cases that attempt to uncover errors in collaborations.

(4) VALIDATION TESTING

Validation testing begins at the culmination of integration testing,when individual components have been exercised the software is completely assembled as a package and interfacing errors have been uncovered and corrected. At the validation or system level the

distinction between conventional and object oriented software disappears. Testing focuses on user visible actions and user recognizable output from the system. Validation succeeds when software functions in a manner that can be reasonably expected by the customer.

1. Validation Test Criteria

Reasonable expectations are defined in the software requirement. The specification contains a section called validation criteria. Information contained in that section forms the basis for a validation testing approach. Software validation is achieved through a series of tests that demonstrate conformity with requirements. A test plan outlines the classes of tests to be conducted and a test procedure defines specific test cases. Both the plan and procedure are designed to ensure that all functional requirement are satisfied, all behavioral characteristics are achieved,all performance requirements are attained documentation is correct and usability and other requirements are met.

After each validation test case has been conducted, one of two possible conditions exist:

1. The function or performance characteristic conforms to specification and is accepted
2. A deviation from specification is uncovered and a deficiency list is created.
3. Configuration Review

An important element of the validation process is a configuration review. The intent of the review is to ensure that all elements of the software configuration have been properly developed are cataloged and have the necessary detail to bolster the support phase of the software life cycle. The configuration review is sometimes called an audit. A software configuration audit should consider the following things:

* To check whether changes specified in the ECO(economic change order) has been made.
* Additional modifications if any have incorporated or not.
* Has formal technical review been conducted to assess technical correctness.
* Has the software process been followed and have software engineering standards been adhered to.
* Have the SCM(software configuration management) procedures for nothing the change,recording it and reporting it has been followed.

Thus software aduit and review increases the visibility and traceability. The developers and the customers are able to use audits to agree on what has been designed and built.

1. Alpha and Beta Testing

It is virtually impossible for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted.strange combinations of data may be regularly used.output that seemed clear to the tester may be unintelligible to a user in the field when custom software is built for one customer a series of accepetance tests are conducted to enable the customer to validate all requirements. Conducted by end user rather than software engineers an acceptance test can range from an informal test drive to a planned and systematically executed series of tests. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

The alpha test is conducted at the developers site by end users. The software is used in a controlled environment with the developer recording errors and usage problems.

The beta test is conducted at end user sites. Unlike alpha testing,the developer is generally not present. Therefore,the beta test is a live application of the software in an environment that cannot be controlled by the developer. The end user records all problems(real or imagined) that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta tests, software engineers make modifications and then prepare for release of the software product to the entire customer base.

(5) SYSTEM TESTING

System Testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose,all work to verify that system elements have been properly integrated and perform allocated functions.

Types of system tests

The following are the types of system tests:

1. Recovery Testing : Many computer-based systems must recover from faults and resume processing within a prespecified time. In some cases a system must be fault tolerant that is processing faults must not cause overall system function to cease. In other cases a system failure must be corrected within a specified period of time or severe economic damage will occur. Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. If recovery is automatic(perform by system itself) reinitialization data recovery and restart are evaluated for correctness. If recovery requires human intervention,the mean time to repair(MTTR) is evaluated to determine whether it is within acceptable limits.
2. Security Testing : Any computer based system that manages sensitive information ia a target for improper or illegal penetration. Penetration spans a broad range of activities: hackers who attempt to penetrate systems for sport,disgruntled employes who attempt to penetrate for revenge , dishonest individuals who attempt to penetrate for illicit personal again. Security testing verifies that protection mechanism built into a system will protect it from improper penetration. During security testing the tester plays the role of the individual who desire to penetrate the system. The tester may attempt to acquire password through external clerical means,may attack the system with custom software designed to break down any defence that have been constructed.thereby denying service to others, may purposely cause system errors hoping to penetrate during recovery may browse through insecure data hopping to find the key to system entry. The role of the system designer is to make penetration cost more than the value of the information that will be obtained.
3. Stress Testing:

Stress tests are designed to confort program with abnormal situation. Stress testing executes a system in a manner that demands resources in abnormal quality,frequency, or volume. For example (1) special tests may be designed that generate ten interrupts per second,when one or two is the average rate(2)input data rates may be increased by an order of magnitude to determine how input functions will respond (3) test cases that requires maximum memory or other resources are executed (4) test cases that may cause memory management problems are designed. Essentially the tester attempts to overwhelm the program.

1. Performance Testing:

For real time and embedded systems software that provides required function but does not conform to performance requirement is unacceptable. Performance testing is designed to test the run time performance of software within the context of an integrated system.

Performance testing occurs throughout all steps in the testing process. Even at the unit level the performance of an individual module may be assessed as tests are conducted. However it is not until all system element are fully integrated that the true performance of a system can be ascertained. Performance tests are often coupled with stress testing and usually require both hardware and software instrumentation. By instrumenting a system the tester can uncover situation that lead to degradation and possible system failure.

6 THE ART OF DEBUGGING

Software testing is an action that can be systematically planned and specified. Tests cases design can be conducted a strategy can be defined and results can be evaluated against prescribe expectations. Debugging is that activity which is performed after executing a successful test case. Debugging consist of determining the exact nature and location of the suspected error and fixing the error. Although debugging can and should be an orderly process,it is still very much an art. A software engineer,evaluating the result of a test is often confronted with a symptomatic indication of a software problem. That is the external manifestation of the error and the internal cause of the error may have no obvious relationship to one another. The poorly understood mental process that connects a symptom to a cause is debugging.

1.The Debugging Process

Debugging is not testing but always occurs as a consequences of testing below,the debugging process begins with the execution of test case. Result are assessed and a lack of correspondence between expected and actual performance is found. Debugging attempts to match the symptom with cause thereby leading to error correction. The steps in debugging are as follows:

Step 1: An attempt is made to reproduce the problem that is the program is run again.

Step 2: The input of the program is simplified so that debugging is much easier. For this some part of the program is used to reproduce the error.

Step 3: Debugging tools(call stack) may be used to track the origin of the problem.

Postmortem analysis can be done by the obtaining the dump of the process space from the system. Core dump are often generated as the system crashes due to certain unhandled errors.

Debugging will always have one of two outcomes:

(1) The cause will found and corrected

(2) The cause will not be found in the later case,the person performing debugging may suspect a cause design one or more test cases to help validate that suspicion and work towards error correction in an iterative fashion.

The following are some of the characteristics of bugs

1.The symptom and the cause may be geographically remote. That is the symptom may appear in one part of a program while the cause may actually be located at a site that is far removed.

2.The symptom may disappear(temporarily) when another error is corrected.

3. The symptom may be caused by human error that is not easily traced.

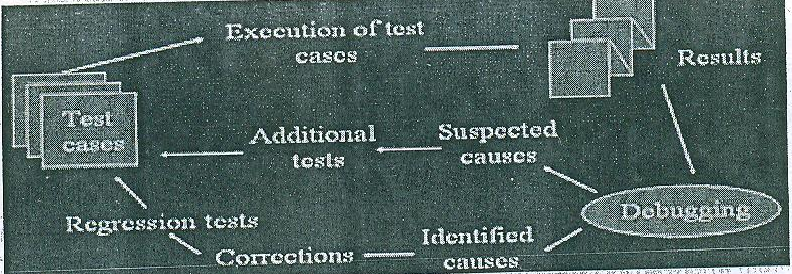
4. The symptom may be result of timing problems rather than processing problems.

5. It may be difficult to accurately reproduce input conditions(e.g a real time application in which input ordering is indeterminate)

6. The symptom may be intermittent. This is particularly common in embedded system that couple hardware and software inextricably.

7. The symptom may be due to cause that are distributed across a number of tasks running on different processors.

Figure: Debugging process



2.Psychological Consideration

The ability to debug differ from programmer to programmer with the same education and experience commenting on the human aspects of debugging Shneiderman states:

Debugging is one of the more frustrating parts of programming. It has elements of problem solving or brain teasers,couple with the annoying recognition that you have made a mistake. Heightened anxiety and the unwillingness to accept the possibility of errors increases the task difficulty. Fortunately there is a great sign of relief and a lessening of tension when the bug is ultimately corrected.

Debugging is probably the most difficult activity in software development from psychological point of view for the following reasons:

* Debugging is done by the person who develop the software and it is hard for that person to acknowledge that an error was made.
* Of all the software development activities debugging is the most mentally taxing because of the way in which most programs are designed and because of the nature of most programming languages i.e the location of any error is potentially any statement in the program.
* Debugging is usually performed under a tremendous amount of pressure to fix the suspected error as quickly as possible.
* Compared to the other software development activities comparatively little research,literature and formal instructions exist on the process of debugging.

1. Debugging strategies

Regardless of the approach that is taken debugging has one overriding objective: to find and correct the cause of a software error. The objective is realized by a combination of systematic evaluation strategies have been proposed

In general three debugging strategies have been proposed

1. Brute force
2. Backtracking
3. Cause elimination

* Debugging by induction
* Debugging by deduction

(4)Program slicing

Each of these strategies can be conducted manually but modern debugging tools can make the process much more effective.

Debugging tactics.

The bruteforce category of debugging is probably the most common and least efficient method for isolating the cause of software error. Generally this method is used when all the other available methods fail. This method is the most common and least effective method where debugging is done using either a memory dump or automated debugging tools. The relationship between the storage locations and variable present in the source program is established if memory dump method is used. After this the program is loaded with the output statement which results in production of a large amount of information.

However using a memory dump for finding errors suffers from the following drawbacks.

1. Establishing a link between storage locations and the variables used in the source program is a cumbersome process.
2. Massive amount of irrelevant data are analysed.
3. The dynamics of the program are needed to detect most errors but a dump shows only the static state of the program at one point of time.
4. No formal procedure exist to find the cause of an error by analyzing a storage dump.

Backtracking is a fairly common debugging approach that can be used successfully. For small programs, in the method of backtracking is often used effectively in locating errors. To use this method, start at the place in the program where an incorrect result was produced and go backwards in the program one step at a time, mentally executing the program in reverse order, to derive the state (or values of all variables) of the program at the previous step. Continuing in this fashion, the error is localized between the point where the state of the program was what was expected and the first point where the state was not what was expected.

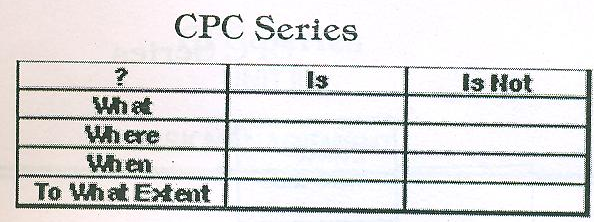
The third approach to debugging  cause elimination  is manifested by induction or deduction.

Debugging by induction

Many errors can be found by using a disciplined thought process without ever going near the computer. One such thought process is induction, where one proceeds from the particulars to the whole. By starting with symptoms of the error, possibly in the result of one or more test cases, and looking for relationships among the symptoms, the error is often uncovered.

The induction process is illustrated in Figure 1 and described as follows :

* Locate the pertinent data. The first step is the enumeration of all that is known about what the program did correctly and what it did incorrectly (i.e. the symptoms that led one to believe that an error exists)
* Organize the date A particularly useful organizational technique that can be used to structure the available date is shown in the following table. The What boxes list the general symptoms, the Where boxes describe where the symptoms were observed, the When boxes list anything that is known about the times that the symptoms occur, and the To What Extent boxes describes the scope and magnitude of the symptoms. Notice the Is and Is Not columns.



* Devise a hypothesis. The next steps are to study the relationships among the clues and devise.one or more hypotheses about the cause of the error. if one cannot devise a theory, more data are necessary,possibly obtained by devising and executing additional test cases.
* Prove the hypothesis. A major mistake at this point,given the pressure under which debugging is usually performed is skipping this step by jumping to conclusion and attempting to fix the problem. A failure to do this often result in the fixing of only a symptoms of the problem or only a portion of the problem. The hypothesis is proved by comparing it to the original clues. If it does not the hypothesis is invalid,the hypothesis is incomplete or multiple errors are present.

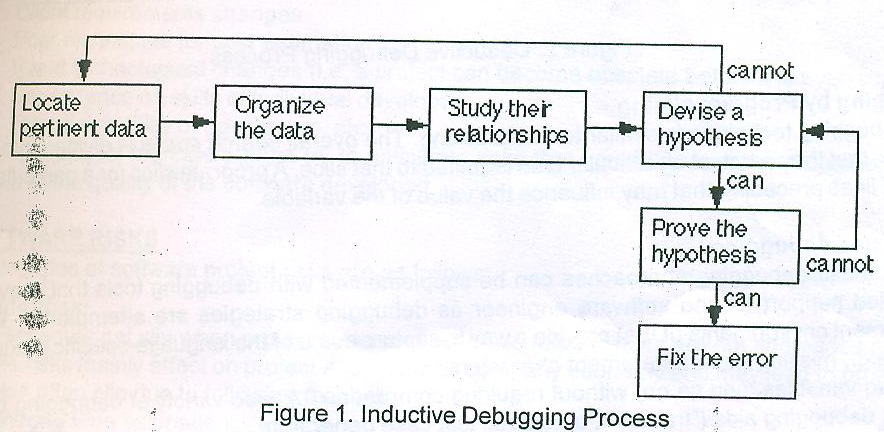


Figure 1. Inductive Debugging Process

Debugging By Deduction

An alternate thought process,that of deduction is a process of proceeding from some general theories or premises to arrive at a conclusion. This process is illustrated in fig 2 and also described as follows:

* Enumerate the possible causes or hypotheses. The first step is to develop a list of all conceivable causes of the error. They need not be complete explanation they are merely theories through which one can structure and analyze the available data.
* Use the data to estimate possible causes. By a careful analysis of the data, particularly by looking for contradictions,one attempts to eliminate all but one of the possible cause. If all are eliminated,additional data are needed to device new theories. If more than one possible cause remains,the most probable cause(the prime hypothesis) is selected first.
* Refine the remaining hypothesis. The possible cause at this point might be correct,but it is unlikely to be specific enough to pinpoint the error. Hence the next step is to use the available clues to refine the theory to something more specific.
* Prove the remaining hypothesis. This vital step is identical to the fourth step in the induction method.

CPC SERIES

Collect more data

Use process of eliminates

Refine remaining hypothesis

Enumerates possible causes

Prove remaining hypothesis

Fix the error

Figure 2. Deductive Debugging Process

Debugging by program slicing

This debugging technique is similar to backtracking. The overall search space is divided into program slices so that the search at a particular time is limited to that slice. A program slice for a particular variable is set of lines preceding that may influence the value of the variable.

Automated debugging

Each of these debugging approaches can be supplemented with debugging tools that provide a automated support for the software engineer as debugging strategies are attempted. Integrated development environment(IDEs) provide a way to capture some of the language specific predetermined errors(e.g missing end of statement characters,undefined variables and so on) without requiring compilation.A wide variety of debugging compilers dynamic debugging aids(traces) automatic test case generators and cross reference mapping tools are available. However tools are not a substitute for careful evaluation based on a complete design model and clear source code.

4. Correcting the error

Once a bug has been found it must be corrected. But a correction of a bug can introduce other errors and therefore do more harm than good. Three simple questions that every software engineer should ask before making correction that removes the cause of a bug.

1. Is the cause of the bug reproduced in another part of the program? In many situations a program error is caused by an erroneous pattern of logic that may be reproduced elsewhere. Explcit consideration of the logical pattern may result in the discovery of other errors.
2. What next bug might be introduced by the flix that is made? Before the correction is made, the source code(or better the design) should be evaluated to assess coupling of logic and data structures. If the correction is to be made in a highly coupled section of the program,special care must be taken when any change is made.
3. What could we have done to prevent this bug in the first place? This question is the first step toward establishing a statistical software quality assurance approach. If we correct the process as well as the product,the bug will be removed from the current program and may be eliminated from all future programs