

Unit - II

Software Project management: project Planning, estimation, **Software requirements and specification:** gathering, analysis, specification, characteristics, organization.

SOFTWARE PROJECT MANAGEMENT AND IT'S ACTIVITIES

SOFTWARE PROJECT MANAGEMENT:

- Good project management can't guarantee success, but poor management on significant projects always leads to failure.
- **A project** is well-defined task, which is a collection of several operations done in order to achieve a goal (for example, software development and delivery).
- A Project can be characterized as:
 - ✓ Every project may has a unique and distinct goal.
 - ✓ Project is not routine activity or day-to-day operations.
 - ✓ Project comes with a start time and end time.
 - ✓ Project ends when its goal is achieved hence it is a temporary phase in the lifetime of an organization.
 - ✓ Project needs adequate resources in terms of time, manpower, finance, material and knowledge-bank.

SOFTWARE PROJECT

- A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.
- Software projects have several properties that make them very different to other kinds of engineering project.
- The product is intangible (unable to be touched; not having physical presence)
 - ✓ It's hard to claim a bridge is 90% complete if there is not 90% of the bridge there. It is easy to claim that a software project is 90% complete, even if there are no visible outcomes.
- We don't have much experience.
 - ✓ Software engineering is a new discipline, and so we simply don't have much understanding of how to engineer large scale software projects.
- Large software projects are often "bespoke" (made for a particular customer or user)
 - ✓ Most large software systems are one-off, with experience gained in one project being of little help in another.
- The technology changes very quickly
 - ✓ Most large software projects employ new technology, for many projects, this is the reason.

SOFTWARE PROJECT MANAGER

- A software project manager is a person who undertakes the responsibility of executing the software project. Software project manager is thoroughly aware of all the phases of SDLC that the software would go through. Project manager may never directly involve in producing the end product but he controls and manages the activities involved in production
 - Let us see few responsibilities that a project manager shoulders -
 - Managing People
 - ✓ Act as project leader
 - ✓ Liaison with stakeholders
 - ✓ Managing human resources
 - ✓ Setting up reporting hierarchy etc.
 - Managing Project
 - ✓ Defining and setting up project scope

- ✓ Managing project management activities
- ✓ Monitoring progress and performance
- ✓ Risk analysis at every phase
- ✓ Take necessary step to avoid or come out of problems
- ✓ Act as project spokesperson

NEED OF SOFTWARE PROJECT MANAGEMENT



- The image above shows triple constraints for software projects. It is an essential part of software organization to deliver quality product, keeping the cost within client's budget constrain and deliver the project as per scheduled. There are several factors, both internal and external, which may impact this triple constrain triangle. Any of three factors can severely impact the other two. Therefore, software project management is essential to incorporate user requirements along with budget and time constraints.

SOFTWARE PROJECT MANAGEMENT ACTIVITIES

- Software project management comprises of a number of activities, which contains planning of project, deciding scope of software product, estimation of cost in various terms, scheduling of tasks and events, and resource management.
- Project Planning
 - Software project planning is task, which is performed before the production of software actually starts. It is there for the software production but involves no concrete activity that has any direction connection with software production; rather it is a set of multiple processes, which facilitates software production. Project planning may include the following:
- Scope Management
 - it creates boundaries of the project by clearly defining what would be done in the project and what would not be done. This makes project to contain limited and quantifiable tasks, which can easily be documented and in turn avoids cost and time overrun.
 - During Project Scope management, it is necessary to -
 - ✓ Define the scope
 - ✓ Decide its verification and control
 - ✓ Divide the project into various smaller parts for ease of management.

- ✓ Verify the scope
- ✓ Control the scope by incorporating changes to the scope
- Project Estimation
 - For an effective management accurate estimation of various measures is a must. With correct estimation managers can manage and control the project more efficiently and effectively.
 - Project estimation may involve the following:
 - Software size estimation
 - ✓ Software size may be estimated either in terms of KLOC (Kilo Line of Code) or by calculating number of function points in the software. Lines of code depend upon coding practices and Function points vary according to the user or software requirement.
 - Effort estimation
 - ✓ The managers estimate efforts in terms of personnel requirement and man-hour required to produce the software. For effort estimation software size should be known. This can either be derived by managers' experience, organization's historical data or software size can be converted into efforts by using some standard formulae.
 - Time estimation
 - ✓ Once size and efforts are estimated, the time required to produce the software can be estimated. Efforts required are segregated into sub categories as per the requirement specifications and interdependency of various components of software. Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure (WBS). The tasks are scheduled on day-to-day basis or in calendar months. The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.
 - Cost estimation
 - ✓ This might be considered as the most difficult of all because it depends on more elements than any of the previous ones. For estimating project cost, it is required to consider -
 - Size of software
 - Software quality
 - Hardware
 - Additional software or tools, licenses etc.
 - Skilled personnel with task-specific skills
 - Travel involved
 - Communication
 - Training and support

1. Software Project Planning

A Software Project is the complete methodology of programming advancement from requirement gathering to testing and support, completed by the execution procedures, in a specified period to achieve intended software product.

Need of Software Project Management

Software development is a sort of all new streams in world business, and there's next to no involvement in structure programming items. Most programming items are customized to accommodate customer's necessities. The most significant is that the underlying technology changes and advances so generally and rapidly that experience

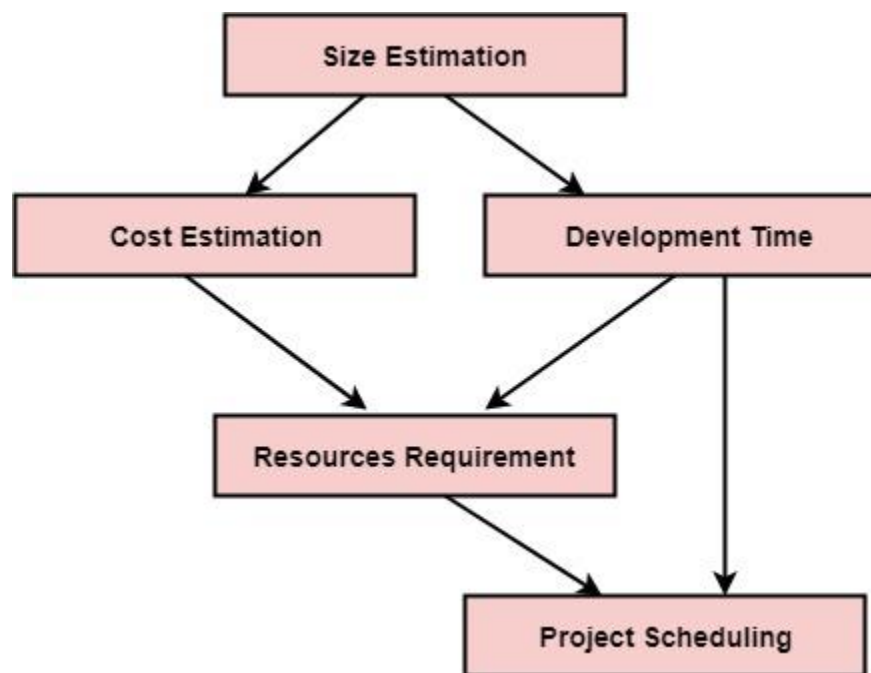
of one element may not be connected to the other one. All such business and ecological imperatives bring risk in software development; hence, it is fundamental to manage software projects efficiently.

Software Project Manager

Software manager is responsible for planning and scheduling project development. They manage the work to ensure that it is completed to the required standard. They monitor the progress to check that the event is on time and within budget. The project planning must incorporate the major issues like size & cost estimation scheduling, project monitoring, personnel selection evaluation & risk management. To plan a successful software project, we must understand:

- Scope of work to be completed
- Risk analysis
- The resources mandatory
- The project to be accomplished
- Record of being followed

Software Project planning starts before technical work start. The various steps of planning activities are:



The size is the crucial parameter for the estimation of other activities. Resources requirement are required based on cost and development time. Project schedule may prove to be very useful for controlling and monitoring the progress of the project. This is dependent on resources & development time.

2. SOFTWARE PROJECT ESTIMATION :

- We discussed various parameters involving project estimation such as size, effort, time and cost.
- Project manager can estimate the listed factors using two broadly recognized techniques

Project size Estimation:

- Accurate Estimation of the problem size is fundamental to satisfactory estimation of effort, time duration, cost of a software project
- The size of a problem is obviously not the number of bytes that the source code occupies. It is neither the byte code size of the executable code.
- The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.

Metrics for project size estimation:

They are two metrics are widely used to Estimate size

1) Lines of code (LOC)

2) Function point (FP)

Lines of code:

- LOC is simplest among all metrics available to estimate project size
- Using this, the project size is estimate by counting the no. of source instructions in the developed program.
- Determining the LOC count at the end of a project is a very simple job however accurate estimation of the LOC count at the beginning of a project is very difficult
- LOC gives a numerical value of problem size that can vary widely with individual coding style, different programmes layout

- their code in different ways
- A good problem size measure should consider the overall complexity of the problem and the effort needed to solve it.
 - LOC measure correlates poorly with the quality and efficiency of the code.
 - LOC is very difficult to accurately estimate. LOC in the final product from the problem specification. The LOC count can be accurately computed only after the code has been fully developed.
 - LOC is any line of program text. That is comment, or blankspaces regardless of no. of statements on the line.

2. Function point Metric (or) Functional Count.

- Function point metric was proposed by Albrecht (1983).
- This metric overcomes many of the shortcomings of the LOC metric.
- Function point metric is that it can be used to easily estimate the size of a software product directly from the problem specification.
- * → Conceptual Idea underlying the function point metric is that the size of a software product is directly dependent on the number of different functions or features it supports.
- Functional count is appeared to be a solution to the size estimation problems.
- Size of the project is estimated on the basis of functions & services requested by the customer in the requirements specification.

→ It is structured technique that decomposes systems down into smaller components so that they can be better understood and analyzed.

→ Function point based estimations are based on the following 5 information domain values

1. No. of i/p's
2. No. of o/p's
3. No. of Inquiries.
4. No. of internal logical files
5. No. of external interface.

Ex: 1

Compute the FP value for the grade calculation of students. Assume that it is an average complexity size project. The information domain values are as follows:

No. of inputs: 13 No. of External files: 5

No. of outputs: 4 No. of Interfaces: 2

No. of Inquiries: 2

The total value of complexity adjustment attribute is 13 [CAA=13]

A) i) Calculation of UFP the avg. complexity size project

$$= (\text{No. of i/p's}) \times 4 + (\text{No. of o/p's}) \times 5 + (\text{No. of inquiries})$$

$$\times 4 + (\text{No. of files}) \times 10 + (\text{No. of Interfaces}) \times 7$$

$$= (13 \times 4) + (4 \times 5) + (2 \times 4) + (5 \times 10) + (2 \times 7)$$

$$= 145$$

2) Compute CAA which has the value = 13

$$\begin{aligned} &= 0.65 + 0.01 + (13 \times 3) \\ &= 0.65 + 0.01 + 39 \\ &= 1.04. \end{aligned}$$

3) $FP = UFP \times CAA$

$$= 175 \times 1.04 = 182$$

→ The value of these five information domains are collected and a subjective evaluation is performed to categorize them as 3 types.

1) Simple

2) Average

3) Complex

Ex: Information domains & their weights.

Information Domain	Simple	Average	Complex
No. of i/p's	3	4	6
No. of o/p's	4	5	7
No. of inquiries	3	4	6
No. of internal logical files	7	10	15
No. of external interface	5	7	10

Function point estimation is performed in the following 3 steps

Calculate the

(1) Unadjusted Function point (UFP)

(2) Complexity adjustment attribute (CAA)

(or) Technical complexity factor (TCF)

(3) Function point

1) Unadjusted function point (UFP):

It is calculated by simply counting the value of each information domain and multiplying it.

2) Complexity adjustment attributes (CAA)

The CAA are complexity attributes that can vary from project to project

$$CAA = [0.65 + 0.01 \times \sum CAA_i]$$

The values of each of these factors is rated on a 5 point scale

0 : (not significant)

incidental : 1

moderate : 2

average : 3

significant : 4

highly essential : 5

3) $FP = UFP \times CAA$

Project Effort Estimation Techniques:

→ Effort Estimation predicts how much time is required to complete a project, how much it costs, how many engineers are required for completing the project

→ Cost Effort includes the cost of the software and hardware, salaries of engineers and ^{other} costs of including the training, travels, tools, supports etc

→ Effort Estimation is important for cost benefit determination, project scheduling and monitoring and control

- The most considerable effort is the requirement of skilled engineers. Therefore, it is generally estimated in terms of ~~person~~ person-Month.
- There are 3 broad categories of estimation techniques.

- Heuristic Estimation Technique (3)

- Empirical Estimation Technique (1)

- Analytical Estimation Technique (2)

1) COCOMO Heuristic Estimation Technique;

- COCOMO means CONSTRUCTIVE Cost Estimation Model

- COCOMO was developed by Barry Boehm in 1981

- COCOMO used for calculating Effort, development time and persons

- COCOMO is classified into 3 classes/model on the basis of s/w project. They are:

- 1) Organic Mode

- 2) Semidetached Mode

- 3) Embedded Mode

Organic Mode:

- ↳ project size: 2-50 KLOC

- Nature of the project: simple

- Deadline of the project: Not tight yet

Semidetached Mode:

- ↳ project size: 50-300 KLOC

- Nature of the project: Medium

- Deadline of the project: ~~Not tight~~ / Medium

Embedded Mode:

- ↳ project size: above 300 KLOC

- Nature of the project: complex

- Deadline of the project: within time

There are 3 types of COCOMO.

→ COCOMO categorized into 3 types:

- 1) Basic COCOMO
- 2) Intermediate COCOMO
- 3) Detailed COCOMO.

1) Basic COCOMO:

↳ It is used to estimate software development time, Effort & No. of persons (or) Engineers.

$$E = a_b (KLOC)^{b_b} \rightarrow P-M$$

$$D = c_b (E)^{d_b} \rightarrow M$$

$$P = E/D \rightarrow P$$

where E is effort applied for person-month

where D is Development Time in months

P is No. of persons in person

$a_b, c_b, b_b, d_b \rightarrow$ coefficients.

Software project	a_b	b_b	c_b	d_b
organic	2.4	1.05	2.5	0.38
semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Q: Suppose that a project was estimated to be 400 KLOC calculating the effort, development time, persons for organic mode.

Sol:- Given 400 KLOC

$$E = a_b (KLOC)^{b_b}$$

$$= 2.4 (400 KLOC)^{1.05}$$

$$= 1295.31 P-M$$

$$\Rightarrow D = C_b(E)^{db}$$

$$= 2.5(1295.31)^{0.38}$$

$$= 38.07 \text{ M}$$

$$\Rightarrow P = \frac{1295.31}{38.07} = 34.02 \text{ P}$$

Semidetached mode:

$$E \Rightarrow a_b(KLOC)^{bb} \quad D = C_b(E)^{db} \quad P = \frac{2462.79}{38.45}$$

$$\Rightarrow 3.0(400)^{1.12} = 2.5(2462.79)^{0.35}$$

$$\Rightarrow 2462.79 = 38.45 = 64 \text{ P}$$

Embedded mode:

$$E \Rightarrow a_b(KLOC)^{bb} \quad D = C_b(E)^{db} \quad P = \frac{4772.81}{37.59}$$

$$\Rightarrow 3.6(400)^{1.20} = 2.5(4772.81)^{0.32}$$

$$\Rightarrow 4772.81 = 37.59 = 126.97$$

$$= 127$$

Q: Suppose that a project was estimated to be 30,000 LOC. Calculate the effort, development time, persons for Semidetached mode.

Sol: $E \Rightarrow a_b(30KLOC)^{bb} \quad D = C_b(E)^{db} \quad P = \frac{135.36}{13.93}$

$$\Rightarrow 3.0(30)^{1.12} = 2.5(135.36)^{0.35}$$

$$\Rightarrow 135.36 \text{ P-M} = 13.93 \text{ M} = 9.71 \text{ P}$$

INTERMEDIATE COCOMO:-

- It is an extension of Basic COCOMO.
- It includes set of 15 additional "cost drivers attributes" to compute the cost of software.
- 15 cost drivers attributes divided into 4 types.

They are:

- 1) Product Attribute
- 2) Hardware Attribute
- 3) Personal Attribute

1. Product Attributes:

- ↳ Software Reliability (RELY)
- ↳ Size of DataBase (DATA)
- ↳ Product complexity (CPLX)

2. Hardware Attributes:

- ↳ Runtime performance constraints (TIME)
- ↳ Memory storage constraints (STORE)
- ↳ Virtual storage constraints (VIRT)
Machine volatility.
- ↳ Required Turn About Time (TURN)

3. Personal Attributes:

- ↳ Analyst capability (ACAP)
- ↳ Applications Experience (AEXP)
- ↳ programmer Capability (PCAP)
- ↳ Virtual Machine Experience (VEXP)
- ↳ programming Language Experience (LEXP)

4. Project Attributes:

- ↳ Modern programming practices (MODP)
- ↳ Use of Software Tools (TOOL)
- ↳ Development Schedule (SCHED)

→ 15 Attributes get ^{point} 64 scale ranging from very low to Extra-high.

→ Each cost drivers ~~are~~ ratings can be viewed as

* very low

* Low

* Normal

* high

* very high

* Extra high

→ Intermediate Model Equations:

$$E = ab(Kloc)^{bb} \times EAF \rightarrow \text{Effort Adjustment Factor} \cdot (P-M)$$

$$D = C_b(E)^{db} (M)$$

$$P = E/b (P)$$

→ The Effort of ~~Card Drive~~ Cost Driver Attributes is given by following table: the product of all Effort multiplies result in Effort Adjustment Factor (EAF)

	Very Low	Low	Normal	High	Very High	Extra High
1. RELY	0.75	0.88	1.00	1.15	1.40	—
2. DATA	—	0.94	1.00	1.08	1.16	—
3. CPLX	0.70	0.85	1.00	1.15	1.30	1.65
4. TIME	—	—	1.00	1.11	1.30	1.66
5. STORE	—	—	1.00	1.06	1.21	1.56
6. VIRT	—	0.87	1.00	1.15	1.30	—
7. TURN	—	0.87	1.00	1.07	1.15	—
8. ACAP	1.46	1.19	1.00	0.86	0.71	—
9. AEXP	1.29	1.13	1.00	0.91	0.82	—
10. PCAP	1.42	1.17	1.00	0.86	0.70	—
11. VEXP	1.21	1.10	1.00	0.90	—	—
12. LEXP	1.14	1.07	1.00	0.95	—	—
13. MODP	1.24	1.10	1.00	0.91	0.82	—
14. TOOL	1.24	1.10	1.00	0.91	0.83	—
15. SCHED	1.243	1.08	1.00	1.04	1.10	—

Q: A project having 30,000 LOC which is an embedded software with critical area software. Reliability is high. Estimate the development effort and no. of persons.

Sol: $E = a_b (KLOC)^{b_b} \times EAF$

$$= 3.6 (30)^{1.20} \times 1.15$$

$$= 245.21 \text{ P-M}$$

$$D = c_b (E)^{d_b}$$

$$= 2.5 (245.21)^{0.32}$$

$$= 14.54 \text{ M}$$

$$P = \frac{E}{D} = \frac{245.21}{14.54} = 16.86 \Rightarrow 17 \text{ P}$$

Q: A project having 40 KLOC which is an organic software with critical area software. Analyst Capability is normal. Estimate the development effort & no. of persons.

Sol: $E = a_b (KLOC)^{b_b} \times EAF$

$$= 2.4 (40)^{1.05} \times 1.00$$

$$= 115.44 \text{ P-M}$$

$$D = c_b (E)^{d_b}$$

$$= 2.5 (115.44)^{0.38}$$

$$= 13.17 \text{ M}$$

$$P = \frac{E}{D} = \frac{115.44}{13.17} = 8.76 \Rightarrow 9 \text{ P}$$

Detailed COCOMO:-

↳ A Major shortcoming of both Basic & Intermediate COCOMO is considers a software product as a single homogeneous Entity

↳ The most large project are made up several similar sub projects

↳ 4 phases are used in Detailed COCOMO.

1) Requirements planning & product Design (RPD)

2) Detailed Design (DD)

3) Coding & Unit Testing (CUT)

4) Integration Testing (IT)

Effort	very Low	Low	Normal	High	Very High	Extra High
1) RPD	1.80	0.85	1.00	0.75	0.55	-
2) DD	1.35	0.85	1.00	0.90	0.75	-
3) CUT	1.35	0.85	1.00	0.90	0.75	-
4) IT	1.50	1.20	1.00	0.85	0.70	-

Equations:

$$E = a_b (KLOC)^{b_b} \times EAF (P-M)$$

$$D = c_b (E)^{d_b} (M)$$

$$P = E/d (P)$$

Q: Consider a problem project having 50,000 LOC which is in organic mode Software Area is RPD is high. Estimate the Effort, Development time and persons?

$$E = a_b (KLOC)^{b_b} \times EAF$$

$$= 2.4 (50)^{1.05} \times 0.75$$

$$= 109.44 \text{ P-M}$$

$$D = C_b (E)^{d_b}$$

$$= 2.5 (109.44)^{0.38}$$

$$= 14.88 \text{ M}$$

$$P = E/D$$

$$= \frac{109.44}{14.88} = 7.35 \approx 7 \text{ person}$$

Q: Consider project have 30KLOC which is semidetected mode. Software area is IT is ~~very~~ low. Estimate

$E, D, P?$

$$\text{Sol: } E = a_b (KLOC)^{b_b} \times EAF$$

$$= 3.0 (30)^{1.12} \times 1.20$$

$$= 162.43 \text{ P-M}$$

$$D = C_b (E)^{d_b}$$

$$= 2.5 (162.43)^{0.35}$$

$$= 14.84 \text{ P}$$

$$P = E/D$$

$$= \frac{162.43}{14.84} = 10.9 \approx 11 \text{ P}$$

1) Empirical Estimation Technique:

- ↳ Empirical Estimation Technique are based on based on making an educated guess of the project parameters
- ↳ By using this technique, prior experience with the development of similar products is helpful
- ↳ there are 2 popular Estimation Techniques. They are:

- * Expert Judgement Technique

- * Delphi Cost Estimation Technique.

Expert Judgement Technique:

- ↳ Expert makes an educated guess of the problem size after ^{observing} the problem thoroughly.
- ↳ The Expert estimates the cost of the different components (i.e., modules (or) subsystems) of the system and then combines them to arrive at the overall estimation.
- ↳ This technique is subject to human errors and individual bias as an expert's estimate may overlook some factors inadvertently.
- An expert making an estimate may not have experience & knowledge of all aspects of a project
- A more refined form of expert judgement is the estimation made by a group of experts minimizes factors such as individual oversight, personal bias, and the desire to win contrast through overly optimistic estimates.

Delphi Cost Estimation:

- Delphi Cost Estimation approach tries to overcome some of the shortcomings of the expert judgement approach.
- Delphi estimation is carried out by a team composed of a group of experts and a coordination.
- The co-ordinator provides each estimation with a copy of the s/w requirements specification (req) document and a form for recording his cost estimate.
- Estimators complete their individual estimates anonymously and submit it to the co-ordinator.
- The co-ordinator prepares and distributes the summary of the responses of all the estimators and includes any unusual rationale noted by any of the estimators.
- However, no discussion among the estimators is allowed during the entire estimation process.
- The idea behind this is that if any discussion is allowed among the estimators then more experienced seniors may easily get influenced by the less experienced seniors.
- After completion of several iterations of estimations the coordinator takes the responsibility of compiling the result of preparing the final estimation.

(iii) Analytical Estimation:

- Halshed has proposed an analytical estimation technique based upon certain assumptions about the project.

→ According to the properties of physics when the data (static properties) of some algorithms are merged with suitable assumptions. This provides an equation to estimate the duration required to write the source code of the algorithm.

→ The length of a code is used to measure the length of the source code program.

$$N = N_1 + N_2$$

N_1 → no. of operators

N_2 → No. of operands.

→ The volume metric is used to measure the amount of storage space required for the program

$$V = N \log(n_1 + n_2) \quad \begin{matrix} N = \text{length} \\ n_1 = \text{operator} \end{matrix} \quad n_2 = \text{operands.}$$

→ Analytical estimation is mainly used for estimating s/w efforts in maintenance, scheduling, reporting projects

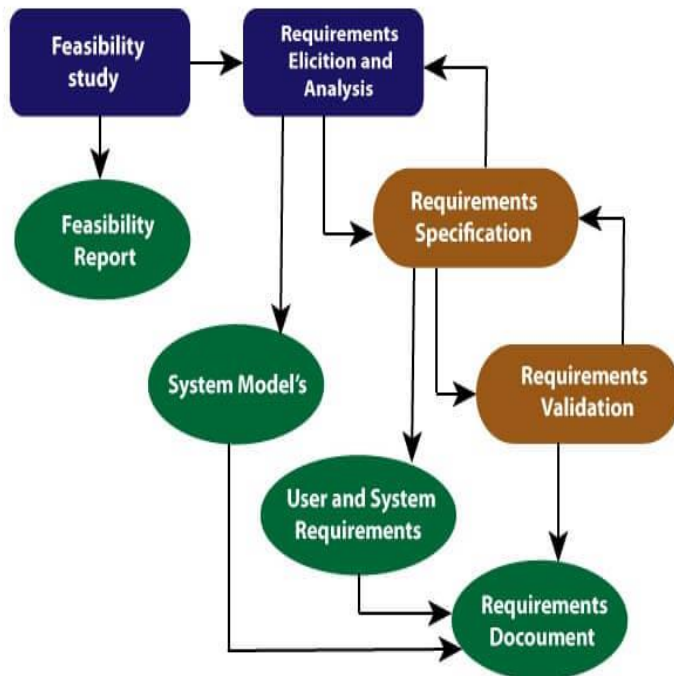
→ It highly considers the programming structure of source code

→ It is the best suited in industries to identify defects in the program.

REQUIREMENT ENGINEERING

- The process to gather the software requirements from client, analyze and document them is known as requirement engineering.
- The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requirements Specification' document.

REQUIREMENT ENGINEERING PROCESS



Requirement Engineering Process

- It is a four step process, which includes –
 - ✓ Feasibility Study
 - ✓ Requirement Gathering
 - ✓ Software Requirement Specification
 - ✓ Software Requirement Validation
- Let us see the process briefly -

FEASIBILITY STUDY

- When the client approaches the organization for getting the desired product developed, it comes up with rough idea about what all functions the software must perform and which all features are expected from the software.
- Referencing to this information, the analysts does a detailed study about whether the desired system and its functionality are feasible to develop.

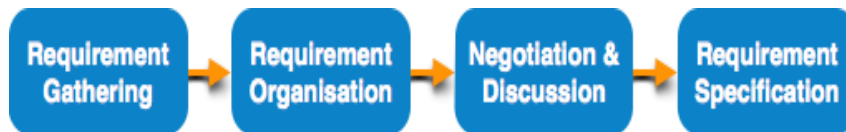
- This feasibility study is focused towards goal of the organization. This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, productivity and integration ability.
- The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

REQUIREMENT GATHERING

- If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user. Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

REQUIREMENT ELICITATION PROCESS(REQUIREMENT GATHERING)

- Requirement elicitation process can be depicted using the following diagram:



- ✓ **Requirements gathering** - The developers discuss with the client and end users and know their expectations from the software.
- ✓ **Organizing Requirements** - The developers prioritize and arrange the requirements in order of importance, urgency and convenience.
- ✓ **Negotiation & discussion** - If requirements are ambiguous or there are some conflicts in requirements of various stakeholders, if they are, it is then negotiated and discussed with stakeholders. Requirements may then be prioritized and reasonably compromised.
The requirements come from various stakeholders. To remove the ambiguity and conflicts, they are discussed for clarity and correctness. Unrealistic requirements are compromised reasonably.
- ✓ **Documentation** - All formal & informal, functional and non-functional requirements are documented and made available for next phase processing.

REQUIREMENT ELICITATION TECHNIQUES

- Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users and others who have a stake in the software system development. There are various ways to discover requirements

INTERVIEWS

- ✓ Interviews are strong medium to collect requirements. Organization may conduct several types of interviews such as:
 - Structured (closed) interviews, where every single information to gather is decided in advance, they follow pattern and matter of discussion firmly.
 - Non-structured (open) interviews, where information to gather is not decided in advance, more flexible and less biased.
 - Oral interviews
 - Written interviews
 - One-to-one interviews which are held between two persons across the table.

- Group interviews which are held between groups of participants. They help to uncover any missing requirement as numerous people are involved.

SURVEYS

- ✓ Organization may conduct surveys among various stakeholders by querying about their expectation and requirements from the upcoming system.

QUESTIONNAIRES

- ✓ A document with pre-defined set of objective questions and respective options is handed over to all stakeholders to answer, which are collected and compiled.
- ✓ A shortcoming of this technique is, if an option for some issue is not mentioned in the questionnaire, the issue might be left unattended.

TASK ANALYSIS

- ✓ Team of engineers and developers may analyze the operation for which the new system is required. If the client already has some software to perform certain operation, it is studied and requirements of proposed system are collected.

DOMAIN ANALYSIS

- ✓ Every software falls into some domain category. The expert people in the domain can be a great help to analyze general and specific requirements.

BRAINSTORMING

- ✓ An informal debate is held among various stakeholders and all their inputs are recorded for further requirements analysis.

PROTOTYPING

- ✓ Prototyping is building user interface without adding detail functionality for user to interpret the features of intended software product. It helps giving better idea of requirements. If there is no software installed at client's end for developer's reference and the client is not aware of its own requirements, the developer creates a prototype based on initially mentioned requirements. The prototype is shown to the client and the feedback is noted. The client feedback serves as an input for requirement gathering.

OBSERVATION

- ✓ Team of experts visit the client's organization or workplace. They observe the actual working of the existing installed systems. They observe the workflow at client's end and how execution problems are dealt. The team itself draws some conclusions which aid to form requirements expected from the software.

SOFTWARE REQUIREMENT SPECIFICATION

- SRS is a document created by system analyst after the requirements are collected from various stakeholders.
- SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.
- The requirements received from client are written in natural language. It is the responsibility of system analyst to document the requirements in technical language so that they can be comprehended and useful by the software development team.
- SRS should come up with following features:
 - ✓ User Requirements are expressed in natural language.
 - ✓ Technical requirements are expressed in structured language, which is used inside the organization.
 - ✓ Design description should be written in Pseudo code.
 - ✓ Format of Forms and GUI screen prints.
 - ✓ Conditional and mathematical notations for DFDs etc.

SOFTWARE REQUIREMENTS

- We should try to understand what sort of requirements may arise in the requirement elicitation phase and what kinds of requirements are expected from the software system.
- Broadly software requirements should be categorized in two categories:

FUNCTIONAL REQUIREMENTS

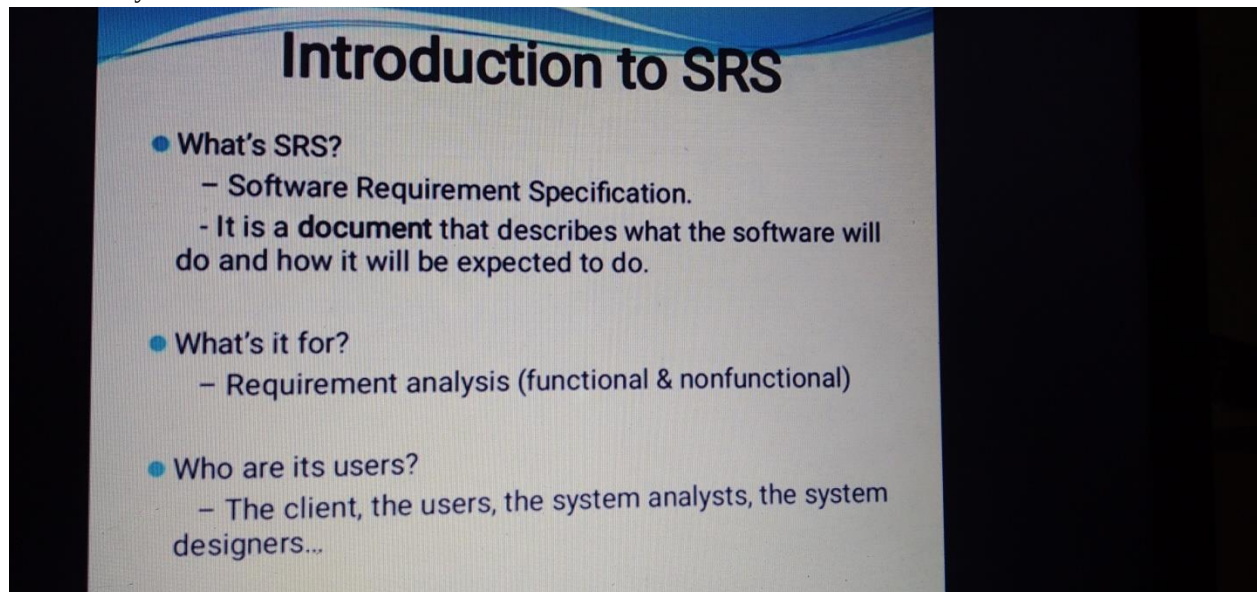
- Requirements, which are related to functional aspect of software fall into this category.
- They define functions and functionality within and from the software system.

Examples -

- ✓ Search option given to user to search from various invoices.
- ✓ User should be able to mail any report to management.
- ✓ Users can be divided into groups and groups can be given separate rights.
- ✓ Should comply business rules and administrative functions.
- ✓ Software is developed keeping downward compatibility intact.

NON-FUNCTIONAL REQUIREMENTS

- Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software, which users make assumption of.
- Non-functional requirements include -
 - ✓ Security
 - ✓ Logging
 - ✓ Storage
 - ✓ Configuration
 - ✓ Performance
 - ✓ Cost
 - ✓ Interoperability
 - ✓ Flexibility
 - ✓ Disaster recovery
 - ✓ Accessibility



Things to be written in SRS

- Introduction
 - Purpose, scope, objectives, reference, ...
- Current System
 - Describe the current state of affairs
- Proposed System
 - Overview
 - Functional requirements
 - Nonfunctional requirements
 - System models (Scenarios, use cases, object model, dynamic models, user interface)

SYSTEM REQUIREMENTS SPECIFICATION for ONLINE RAILWAY RESERVATION SYSTEM



Table of Contents

1. Introduction
2. Overall Description
 - 2.1 Product Perspective
 - 2.2 Product Functions
 - 2.2.1 Administrators
 - 2.2.2 Normal Users
 - 2.3 Operating Environment
 - 2.4 User Characteristics
 - 2.5 Design and Implementation Constraints
3. Specific Requirements
 - 3.1 Functional Requirements
 - 3.2 Non-Functional Requirements
4. Diagrams
 - 4.1 E-R Diagram
 - 4.2 Use case Diagram
 - 4.3 Data Flow Diagram

INTRODUCTION

Railway Reservation System is basically an internet based application so that the users can able know the details and availability of the trains.

The project is specifically designed for the users to book tickets online. The product will work as a complete user interface for Railway Reservation System management process and usage from ordinary users.

OVERALL DESCRIPTION

- **Product Perspective**

It is a replacement for the traditional information management systems which depend on paper work for recording the booking of tickets and users' information.

- **Product Functions**

- **Administrators :**

- Admin should be able to insert, modify and delete train info.
- Can accept or reject a new user according to the Railway Reservation policy or payment methods.
- Can get the information (status report) of any member who has book tickets.

- **Users**

- Members are given a provision to check their account's information and change it.
- Members have the ability to search trains by categories
- Members can book tickets.
- Can change or cancel tickets according to the railway policy.
- The customer may give suggestions.

SPECIFIC REQUIREMENTS

Functional Requirements

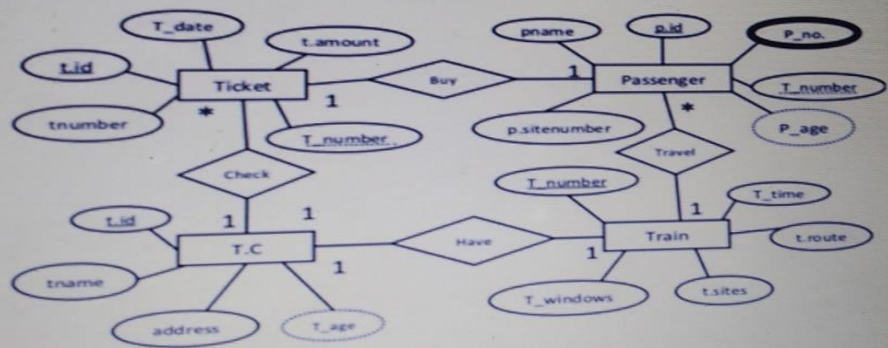
This section provides requirement overview of the system that can be implemented by the system will be –

- **Registration-**
- **Login-**
- **Enquiry and Booking-**
- **Payment-**
- **Logout-**
- **Report/Ticket Generation-**

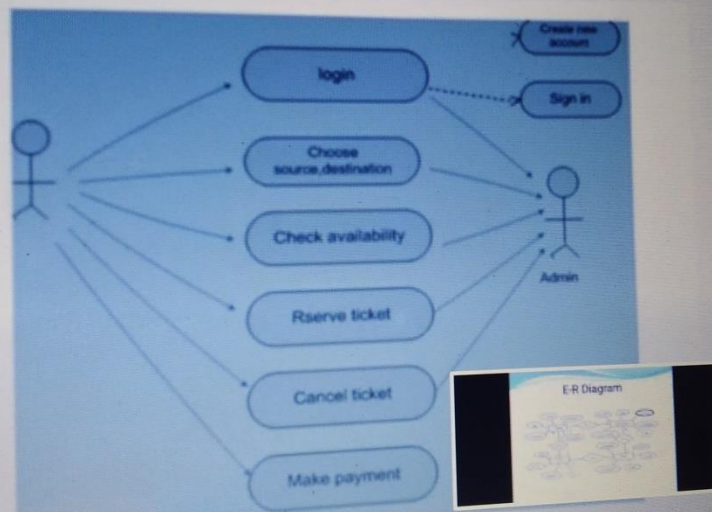
Non Functional Requirement

- **Security-**
- **Reliability-**
- **Availability-**
- **Maintainability-**
- **Portability-**

E-R Diagram



USE CASE DIAGRAM



Data Flow Diagram

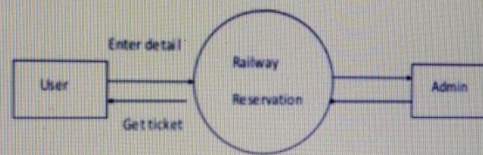


Fig 3- Level 0 DFD

Level 0: Describes the Dataflow with Login and Logout having processes of reservation and cancellation in between.

Data Flow Diagram

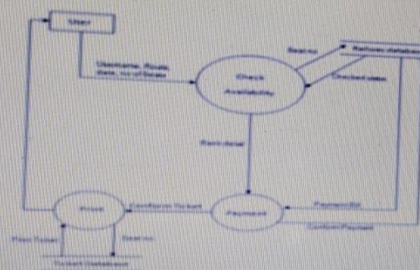


Fig - Level 1 DFD

Level 1: Describes the data flow from Login to Logout consisting of Seat availability and payment management, Admin's Report Generation Tasks etc.

Data Flow Diagram

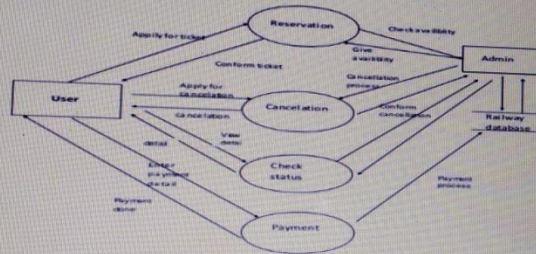


Fig – Level 2 DFD

Level 2: Describes the data flow from Login to Logout consisting details of Seat availability and payment management, Admin's Report Generation Tasks etc.

SOFTWARE REQUIREMENT VALIDATION

- After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal, impractical solution or experts may interpret the requirements incorrectly. This results in huge increase in cost if not nipped in the bud. Requirements can be checked against following conditions -
 - ✓ If they can be practically implemented
 - ✓ If they are valid and as per functionality and domain of software
 - ✓ If there are any ambiguities
 - ✓ If they are complete
 - ✓ If they can be demonstrated

SOFTWARE REQUIREMENTS CHARACTERISTICS

- Gathering software requirements is the foundation of the entire software development project. Hence they must be clear, correct and well-defined.
- A complete Software Requirement Specifications must be:
 - ✓ Clear
 - ✓ Correct
 - ✓ Consistent
 - ✓ Coherent
 - ✓ Comprehensible
 - ✓ Modifiable
 - ✓ Verifiable
 - ✓ Prioritized
 - ✓ Unambiguous
 - ✓ Traceable
 - ✓ Credible source
 - ✓

SOFTWARE SYSTEM ANALYST(Organization)

- System analyst in an IT organization is a person, who analyzes the requirement of proposed system and ensures that requirements are conceived and documented properly & correctly. Role of an analyst starts during Software Analysis Phase of SDLC. It is the responsibility of analyst to make sure that the developed software meets the requirements of the client.
 - ✓ System Analysts have the following responsibilities:
 - ✓ Analyzing and understanding requirements of intended software
 - ✓ Understanding how the project will contribute in the organization objectives
 - ✓ Identify sources of requirement
 - ✓ Validation of requirement
 - ✓ Develop and implement requirement management plan
 - ✓ Documentation of business, technical, process and product requirements
 - ✓ Coordination with clients to prioritize requirements and remove and ambiguity
- Finalizing acceptance criteria with client and other stakeholders