

Software Metrics

Software Measurements

- Software Measurement is indicator of size, quantity amount or dimension of particular attribute of a product or process.
- It helps the project manager & entire software team to take decisions that lead to successful completion of the project by generating quantity result.

► Software measurements are of two categories:

1. Direct Measures: It include software processes like Cost & Effort applied, Lines of code produced, Execution speed & Total no. of errors that have been reported.

2. Indirect Measures: It include products like Functionality, Quality, Complexity, Reliability, Maintainability and many more.



Software Metrics

- Software Metrics provide measures, functions or formulas for various aspects of software process & product.
- It including measuring software performance, planning work items, measuring productivity, and many other uses.

Software metrics are of three categories:

1.Product Metrics: It estimate Size, Complexity, Quality & Reliability of software.

2. Process Metrics: It estimate Faults rate during development, Pattern of testing defect arrival. Time it takes for a fixed operation.

3. Project Metrics: It estimate Number of software developers. Cost, Scheduling and Productivity of software.



Software Measurement Principles

- 1. Formulation:** The derivation of software measures and metrics appropriate for the representation of the software that is being considered.
- 2. Collection:** The mechanism used to accumulate data required to derive the formulated metrics.
- 3. Analysis:** The computation of metrics and the application of mathematical tools.
- 4. Interpretation:** The evaluation of metrics resulting in insight into the quality of the representation.
- 5. Feedback:** Recommendation derived from the interpretation of product metrics transmitted to the software team.

Size Metrics: LOC

- Size-Oriented Metrics concentrates on the size of the program created.

1. LOC (Lines of Code):

- It is one of the earliest and simpler metrics for calculating the size of the computer program.
- It is generally used in calculating and comparing the productivity of programmers.

- LOC not count comment or blank line in code.

- **Example:** Total Line of Code (LOC)=09

```
//Import header file
#include <iostream>
int main() {
    int num=10;
    //Logic of even number
    if (num%2==0)
    {
        cout<<"It is even number";
    }
    return 0;
}
```

Size Metrics: LOC

Advantages:

1. Most used metric in cost estimation.
2. It is very easy in estimating the efforts.

Disadvantages:

1. It cannot measure the size of the specification.
2. Bad software design may cause an excessive line of code
3. It is language dependent
4. Users cannot easily understand it

Tamir Chahh

Project	LOC	Cost \$K	Efforts (Persons/ Month)	Documenta tion	Errors
A	10000	110	18	365	39
B	12000	115	20	370	45
C	15400	130	25	400	32

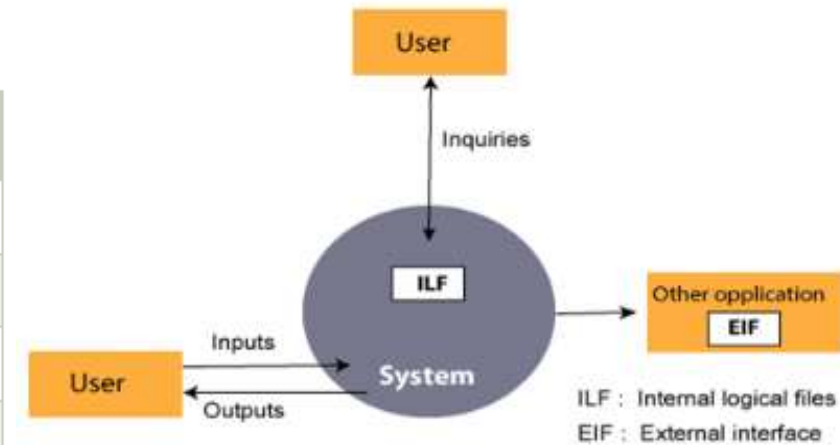
Size Metrics: FP

2. FP (Functional Point):

- Function-oriented software metrics measure functionality, what the system performs, is the measure of the system size.
- It find out by counting the number and types of functions used in the applications

Types of FP Attributes

Measurements Parameters	Examples
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.



FPA's Functional Units System

Size Metrics: FP

Advantages:

1. Need detail specification.
2. Not restricted to code.
3. Language Interdependent.

Disadvantages:

1. Ignore quality issues.
2. Subjective counting depend on estimation.

	A	B	C	D	E	F	G
	Info Domain	Optimistic	Likely	Pessim.	Est Count	Weight	FP count
1							
2	# of inputs	22	26	30	26	4	104
3	# of outputs	16	18	20	18	5	90
4	# of inquiries	16	21	26	21	4	84
5	# of files	4	5	6	5	10	50
6	# of external inter	1	2	3	2	7	14
7	UFC: Unadjusted Function Count						342
8			Complexity adjustment factor				1.17
9						FP	400

Differentiate between FP and LOC

FP	LOC
1. FP is specification based.	1. LOC is an analogy based.
2. FP is language independent.	2. LOC is language dependent.
3. FP is user-oriented.	3. LOC is design-oriented.
4. It is extendible to LOC.	4. It is convertible to FP (backfiring)

Software Project Estimation

- Effective software project estimation is important activity in software development project.
- Estimation in software engineering is a procedure that predicts the time and budget that required for completing a project.
- Project Estimation requires the use of complex tools & good mathematical as well as knowledge about planning.
- One of the main reasons software programs fail is our inability to accurately estimate software size.

► Responsible Persons:

- ☐ Software Manager
- ☐ Cognizant Engineers
- ☐ Software Estimators.



Factors affect on Project Estimation

- 1. Cost:** The project will fail if you do not have sufficient funds to complete it. So. At early stage estimate project cost & ensure you have enough money to complete the work.
- 2. Time:** Estimate both overall project duration & timing of individual tasks. It also enables you to manage client expectations for key deliverables.
- 3. Size & Scope:** All the tasks that must be completed in order to deliver product on time. You can ensure that you have right materials & expertise on the project by estimating how much work is involved and exactly what tasks must be completed.
- 4. Risk:** Estimating predicting risk, what events will occur during the project's life cycle and how serious they will be. Create risk management plans.
- 5. Resource's:** Resource management ensures that you have all the resources you require & make best use of them. Like, tools, people, materials, hardware, software & other resources

Steps of Software Project Estimation

1. Estimate Project Size
2. Estimate Efforts (Person per month/hr)
3. Estimate Project Schedule (Month/Year)
4. Estimate Project Cost (Currency / Dollars)

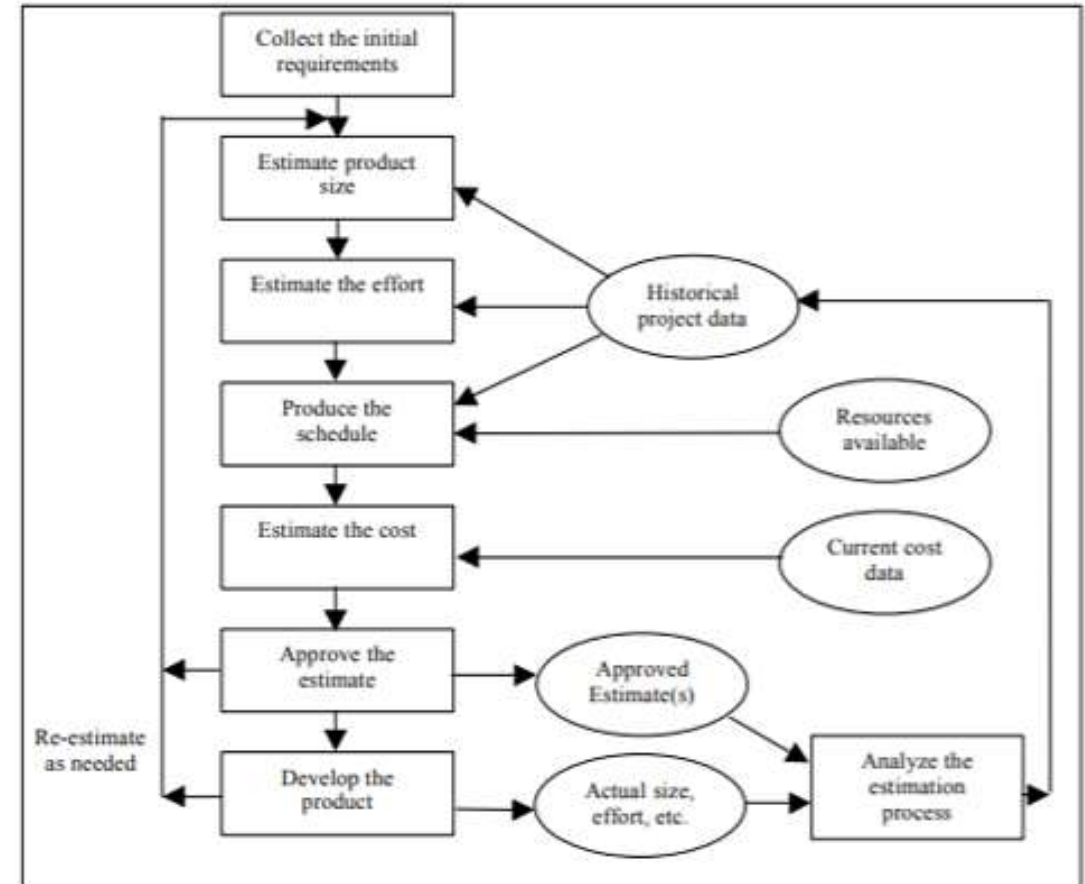
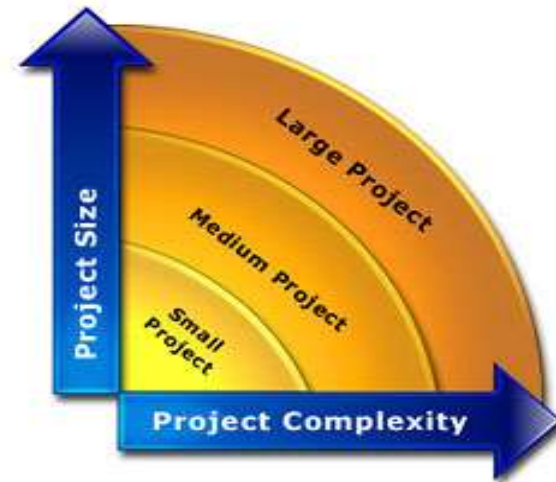


Figure 1 – The Basic Project Estimation Process

Steps of Software Project Estimation

1. Estimate Product Size (LOC & FP):

- It is the very first step to make an effective estimation of the project.
- A Customer's requirements, SRS Document & System design document used for estimating the size of a software.
- Estimate project size can be through similar project developed in past. This is called estimation by Analogy.
- The system is divided into several subsystems depending on functionality & size of each subsystem is calculated.



Steps of Software Project Estimation



2. Estimate Project Efforts:

- The estimation of effort can be made from the organizational specifics of SDLC.
- **Software development project involves Design, Coding. Testing.** Writing, Reviewing documents, Implementing prototypes & Deliverable it decide overall project efforts.
- The project effort estimate requires you to identify and estimate & then sum up all the activities you must perform to build a product of the estimated size.

There are two main ways:

1. The best way to estimate effort is based on the organization's own historical data of development process. Suppose you have similar SDLC, Project size, Development methodology, Tools, Team with similar skills and experience for the new project.
2. If the project is in different nature which requires the organization to adopt different strategy or different models based on algorithmic approach. Ex. COCOMO Model.

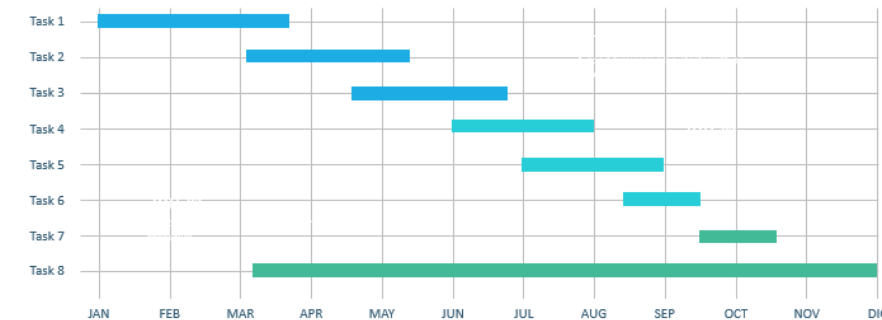
Steps of Software Project Estimation

3. Estimate Project Schedule:

- It involves estimating number of people who will work on the project & what they will work on (the Work Breakdown Structure).
- Also, when they will start working on project & when they will finish.
- Once you have this information, you need to lay it out into a calendar schedule.
- Efforts in man-month are translated to calendar months thumb rule is used.

$$\text{Schedule in calendar months} = 3.0 * (\text{man-months})^{1/3}$$

- The parameter 3.0 is variable, used depending on the situation which works best for the organization.



Steps of Software Project Estimation

4. Estimate Project Cost:

The cost of a project is derived not only from the estimates of person efforts and size.

Other parameters such as purchase hardware. software. travel for meeting. telecommunication costs (long distance phone calls, video-conferences), training, office space etc.

Exactly how you estimate total project cost will depend on how your organization allocates costs.

The simplest labor cost can be obtained by multiplying the project's effort estimate (in hours) by a general labor rate (\$ per hour).

A more accurate labor cost would result from using a specific labor rate for each staff position (e.g.. Technical, QA, Project Management, Documentation, Support, etc.)

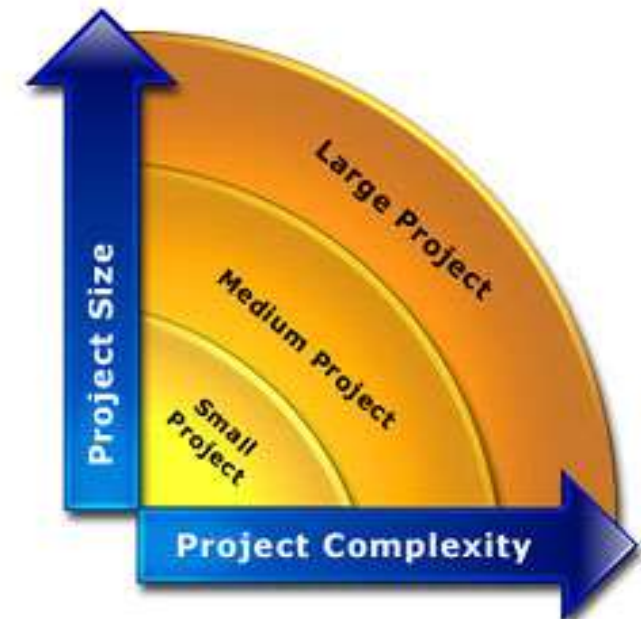


Decomposition Techniques

- **Software Project Estimation** is a form of problem solving (To estimate cost, time & efforts in software project.)
- **Decomposition Technique** is divide & conquer approach of Software Project Estimation.
- By decomposition a project into major functions like software engineering related activities, cost, schedule & efforts estimation can be performed in stepwise manner.

Decomposition Techniques are:

1. Software Sizing
2. Problem based Estimation
3. Process based Estimation



1. Software Sizing

Sizing represents the project planner's first major challenge.

► **The accuracy of a software project estimate is predicated on a number of things:**

1. The degree to which the planner has properly estimated the size of the product to be built.
2. The ability to translate the size estimate into human effort, calendar time and dollars.
3. The degree to which the project plan reflects the abilities of the software team.
4. The stability of product requirements & environment that supports software engineering effort.

► **Approaches of Software Sizing:**

1. "Fuzzy Logic" Sizing
2. Function Point Sizing
3. Standard Component Sizing
4. Change Sizing

Approaches of Software Sizing

Putnam and Myers suggest four different approaches of Software Sizing Problem:

1. "Fuzzy Logic" Sizing:

- To apply this approach, the planner must identify the type of application.
- Although personal experience can be used, the planner should also have access to historical database of projects so that estimates can be compared to actual experience.

2. Function Point Sizing:

- The planner develops estimates of the information domain characteristics.

Approaches of Software Sizing

3. Standard Component Sizing:

- Standard components for an information system are subsystems, modules, screens, reports, interactive programs, batch programs, files. LOC and object-level instructions.
- Uses historical project data to determine the delivered size per standard component.

4. Change 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 reuse, adding code, changing code, deleting code of modifications 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

- Lines of code and function points were described as measures from which productivity metrics can be computed. (LOC/pm & FP/pm)
- **LOC and FP data are used in two ways during software project estimation:**
 1. As an estimation variable to "size" each element of the software.
 2. As baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections.

➤ A three-point or expected value can then be computed

$$S = (s_{opt} + 4s_m + s_{pess})/6$$

– S = expected-value for the estimation variable (size)

– s_{opt} = optimistic value

– s_m = most likely value

– s_{pess} = pessimistic value

Examples of LOC & FP

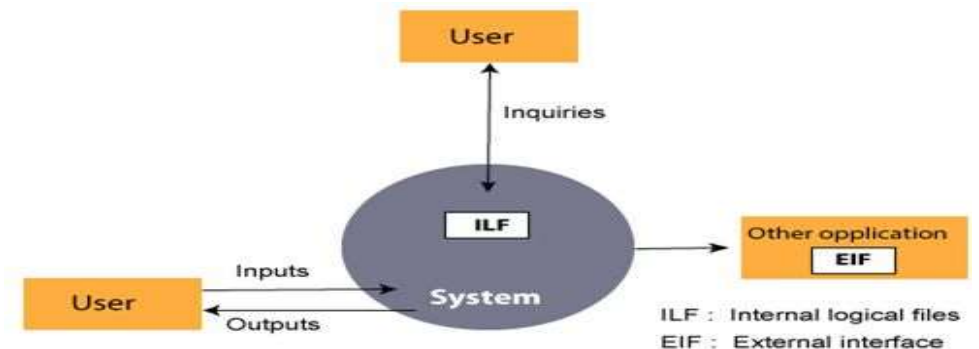
1. LOC (Lines of Code)

Project	LOC	Cost SK	Efforts (Persons/ Month)	Documentation	Errors
A	10000	110	18	365	39
B	12000	115	20	370	45
C	15400	130	25	400	32

Function	Estimated LOC
User interface and control facilities (UICF)	2,300
Two-dimensional geometric analysis (2DGA)	5,300
Three-dimensional geometric analysis (3DGA)	6,800
Database management (DBM)	3,350
Computer graphics display facilities (CGDF)	4,950
Peripheral control function (PCF)	2,100
Design analysis modules (DAM)	8,400
<i>Estimated lines of code</i>	<i>33,200</i>

2. FP(Functional Points)

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						320



3. Process based Estimation

- Most common technique for estimating project is to estimate on the process that will be used.
- The process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated.
- Once problem functions & process activities are decided, the planner estimates the effort (e.g., person-months) that will be required to accomplish each software process activity for each software function.
- Average labor rates (cost/unit effort) are then applied to the effort estimated for each process activity.
- It is very likely the labor rate will vary for each task.
- Senior staff heavily involved in early activities are generally more expensive than junior

Software Cost Estimation

- The Software Cost Estimation is a process of predict/estimate the approximate cost of the software project before the development starts.
- Cost estimation models are some mathematical algorithms or parametric equations that are used to estimate the cost of a product or a project.

► Factors considering for Project Budgeting:

1. Specification & Scope
2. Location
3. Duration
4. Project Team Efforts
5. Resources



Tools and Techniques of Software Cost Estimation

1. Expert Judgment:

- The experts are the people who have prior knowledge on similar kind of projects.
- So, they can suggest valuable insight based on their experience.
- You can also take their advice on various tools and techniques that can be used to estimate similar kind of project.

2. Analogous Estimation:

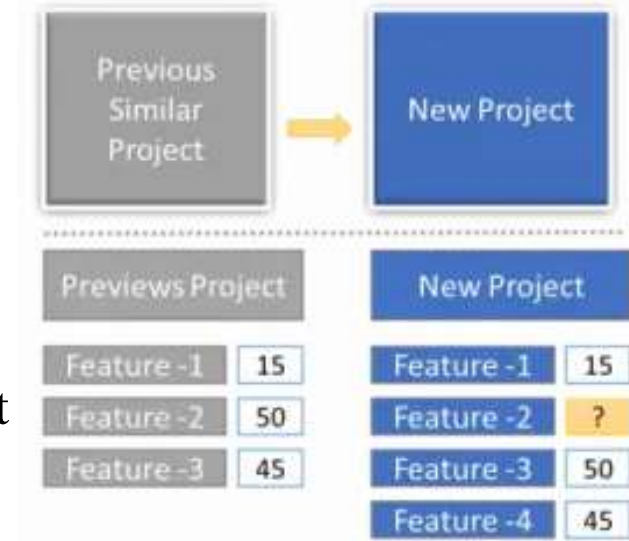
- It uses historical data from similar projects like scope, budget, duration, size, weight and complexity etc.
- It measures the current project on that basis and does the estimation.
- It used in the early phases of a project and is less costly, time & accurate than other methods.



Tools and Techniques of Software Cost Estimation

3. Parametric Estimation:

- It analyze a past project and get the man-hours for each unit.
- Then it estimate the total man-hour for the new project's activity.
- It use statistical modelling technique to identify both cost and time for current project using data from past projects.



4. Bottom-up Estimation:

- In WBS, All the project is divided into smaller work packages & cost of each work package is estimated one by one
- It estimate every single activity and rolling them up into an overall project estimate.
- It is a time consuming and costly technique but it provides the most accurate estimation.

Tools and Techniques of Software Cost Estimation

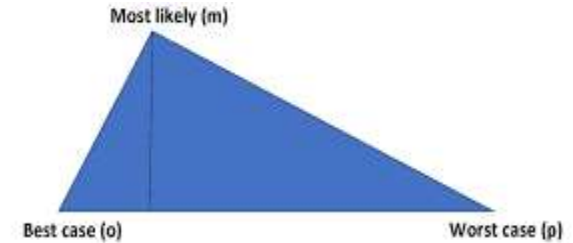
5. Three-Point Estimation:

- To deal with uncertainties and risk PERT method is used.
- Program Evaluation and Review Technique which provide three different estimates.

1. Most Likely Estimate: Assuming a normal case in which everything goes as usual.

2. Pessimistic Estimate: Assuming everything goes wrong i.e. worst case scenario

3. Optimistic Estimate: Assuming everything goes as planned i.e. best case scenario



6. Reserve Analysis:

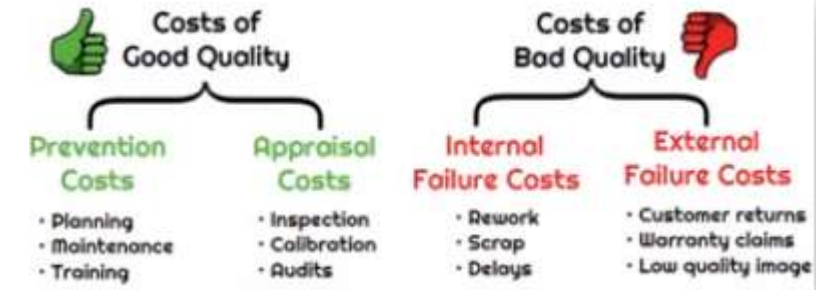
- It is important to keep the reserve budget to deal with uncertain events.
- Project manager need to take approval from the sponsors for management of reserve budget.



Tools and Techniques of Software Cost Estimation

7. Cost of Quality:

- It includes money spent during the project to avoid failures.
- It includes money spent during & after the project due to failures.
- These money included in the project cost estimate.



8. Vendor Bid Analysis:

- It is used to estimate what the project should cost by comparing the bids submitted by multiple vendors.
- There may be differences in their bids but you can get an idea considering the average bid values.



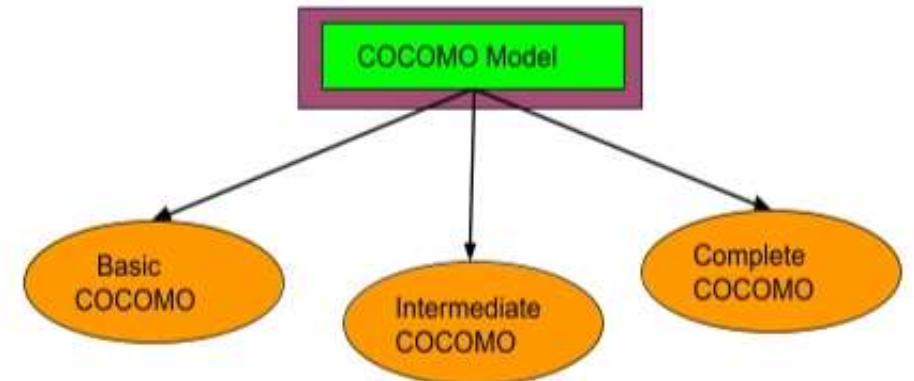
Typical Problems with IT Cost Estimation

1. To estimate large software project is complex task. There are various modifications are there But estimation process done at early stage of process not at various stages.
2. Many people doing estimation but they have little experience about it. Try to provide training & mentoring them.
3. Human beings are biased towards underestimation. Senior IT professionals might make estimates based on their own abilities and forget that many younger people will be working on a project.
4. Estimators might also forget to allow for extra costs needed for integration and testing on large IT projects.
5. Management might ask for an estimate but really want a more accurate number to help them create a bid to win a major contract or get internal funding.



About COCOMO Model

- It was developed by a scientist Barry Boehm in 1981.
- The COCOMO (Constructive Cost Model) is one of the most popularly used software cost estimation models.
- This model depends on the size means number of lines of code for software product development.
- It estimates Effort required for project, Total project cost & Scheduled time of project.



Software Project Types

- In COCOMO, projects are categorized into three types:

1. Organic Type:

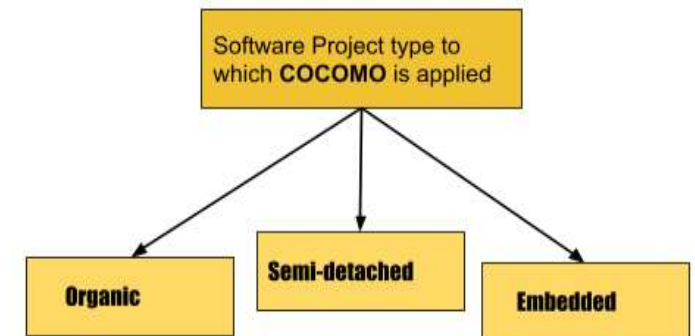
Project is small and simple. (2-50 KLOC)

Few Requirements of projects.

Project team is small with prior experience.

The problem is well understood and has been solved in the past.

Examples: Simple Inventory Management Systems & Data Processing Systems.



Software Project Types

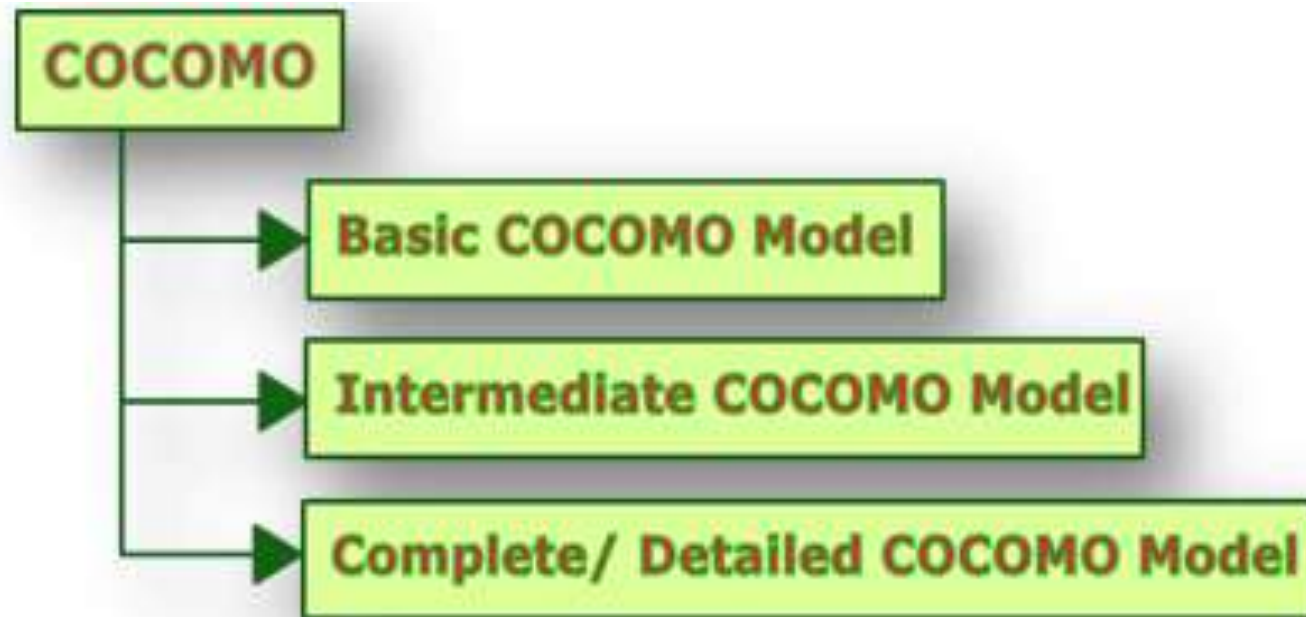
2. Semidetached Type:

- Medium size and has mixed rigid requirements. (50-300 KLOC) Project has Complexity not too high & low.
- Both experienced and inexperienced members in Project Team.
- Project are few known and few unknown modules.
- Examples: Database Management System, Difficult inventory management system.

3. Embedded Type:

- Large project with fixed requirements of resources. (More then 300 KLOC)
- Larger team size with little previous experience.
- Highest level of complexity, creativity and experience requirement.
- Examples: ATM, Air Traffic control, Banking software.

Types of COCOMO Model



Type 1: Basic COCOMO Model

- It is type of static model to estimates software development effort quickly and roughly.
- It mainly deals with the number of lines of code in project.

Formula:

- **Effort (E)** = $a \cdot (\text{KLOC})^b$ MM
- **Scheduled Time (D)** = $c \cdot (E)^d$ Months(M)

Where,

- **E** = Total effort required for the project in Man-Months (MM).
- **D**=Total time required for project development in Months (M).
- **KLOC** = The size of the code for the project in Kilo lines of code.
- **a, b, c, d** = The constant parameters for a software project.

Software Project	A	B	C	D
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Type 1: Basic COCOMO Model Example

Problem Statement:

► Consider a software project using semi-detached mode with 300 Kloc. Find out Effort estimation, Development time and Person estimation.

Solution:

$$\text{Effort (E)} = a * (\text{KLOC})^b = 3.0 * (300)^{1.12} = 1784.42$$

$$\text{PM Development Time (D)} = c(E)^d = 2.5(1784.42)^{0.35} = 34.35 \text{ Months(M)}$$

$$\text{Person Required (P)} = E/D = 1784.42/34.35 = 51.9481 \text{ Persons} \sim 52 \text{ Persons}$$

Type 2: Intermediate COCOMO Model

- Extension of Basic COCOMO model which enhance more accuracy to cost estimation model result.
- It include cost drivers (Product, Hardware, Resource & project Parameter) of project.

Formula:

- **Effort (E)** = $a * (\text{KLOC}) * \text{EAF MM}$
- **Scheduled Time (D)** = $c(E)d \text{ Months(M)}$

Where,

- E = Total effort required for the project in Man-Months (MM).
- D=Total time required for project development in Months (M).
- KLOC=The size of the code for the project in Kilo lines of code.
- a, b, c, d =The constant parameters for the software project.

Software Project	A	B	C	D
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Type 2: Intermediate COCOMO Model

- **EAF**: Effort Adjustment Factor, which is calculated by multiplying the parameter values of different cost driver parameters. For ideal, the value is 1.

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very High	Extra High
Product Attributes						
RELY	0.75	0.88	1.00	1.15	1.40	..
DATA	..	0.94	1.00	1.08	1.16	..
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
TIME	1.00	1.11	1.30	1.66
STOR	1.00	1.06	1.21	1.56
VIRT	..	0.87	1.00	1.15	1.30	..
TURN	..	0.87	1.00	1.07	1.15	..

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Personnel Attributes						
ACAP	1.46	1.19	1.00	0.86	0.71	..
AEXP	1.29	1.13	1.00	0.91	0.82	..
PCAP	1.42	1.17	1.00	0.86	0.70	..
VEXP	1.21	1.10	1.00	0.90
LEXP	1.14	1.07	1.00	0.95
Project Attributes						
MODP	1.24	1.10	1.00	0.91	0.82	..
TOOL	1.24	1.10	1.00	0.91	0.83	..
SCED	1.23	1.08	1.00	1.04	1.10	..

Type 2: Intermediate COCOMO Model Example

Problem Statement:

- For a given Semidetached project was estimated with a size of 300 KLOC. Calculate the Effort, Scheduled time for development by considering developer having high application experience and very low experience in programming.

Solution:

- $EAF = 0.82 * 1.14 = 0.9348$
- Effort (E) = $a * (KLOC)^b * EAF = 3.0 * (300)^{1.12} * 0.9348 = 1668.07 \text{ MM}$
- Scheduled Time (D) = $c * (E)^d = 2.5 * (1668.07)^{0.35} = 33.55 \text{ Months (M)}$

Type 3: Detailed / Complete COCOMO Model

- The model incorporates all qualities of both Basic COCOMO and Intermediate COCOMO strategies on each software engineering process.
- The whole software is divided into different modules and then apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model

Type 3: Detailed / Complete COCOMO Model

Problem Statement:

A distributed Management Information System (MIS) product for an organization having offices at several places across the country can have the following sub-components:

- Database part
- Graphical User Interface (GUI) part
- Communication part

Solution:

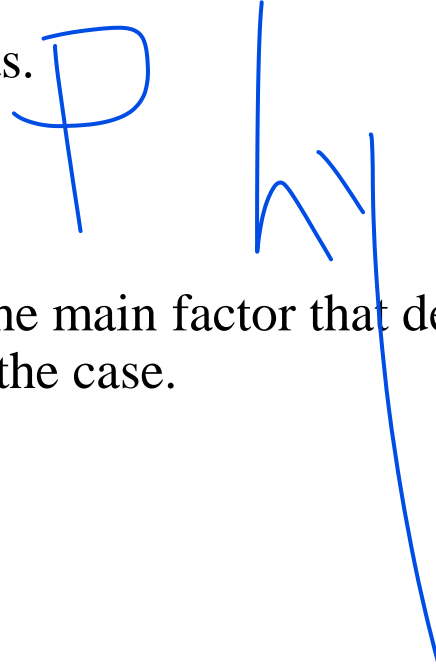
- ✓ The communication part can be considered as Embedded software.
- ✓ The database part could be Semi-detached software.
- ✓ The GUI part Organic software.

The costs for these three components can be estimated separately and summed up to give the overall cost of the system.

Advantages of COCOMO Model

1. Provides a systematic way to estimate the cost and effort of a software project.
2. Estimate cost and effort of software project at different stages of the development process.
3. Helps in identifying the factors that have the greatest impact on the cost and effort of a software project.
4. Provide ideas about historical projects.
5. Easy to implement with various factors.

Disadvantages of COCOMO Model

1. It ignores requirements, customer skills and hardware issues.
 2. It limits the accuracy of the software costs.
 3. It is based on assumptions and averages.
 4. It mostly depends on time factors.
 5. Assumes that the size of the software is the main factor that determines the cost and effort of a software project, which may not always be the case.
- 
- Handwritten blue scribbles, including a large 'P' and a wavy line, are present on the right side of the list.

Source of Risk

► What is Risk?

✓ "Risk is an uncertain future event with a probability of occurrence and potential for loss"

► Source of Risk:

1. Misunderstanding of customer requirements.
2. Uncontrolled & continuous changing of customer requirements.
3. Unrealistic promises given to the customers.
4. Misunderstanding of real impact of new methodologies.
5. Miscalculation of robustness & extensibility of software design.
6. Miscalculation of Team work & group effectiveness.
7. Wrong budget estimation



Risk Management

- Risk Management is an important part of project planning activities.
- It involves identifying and estimating the probability of risks with their order of impact on the project.



Risk Identification

- The project organizer needs to find out risk in the project as early as possible.
- So, the impact of risk can be reduced by making effective risk management planning.
- By doing Brainstorming, SWOT Analysis, Casual Mapping & Flowchart methods are used.

There are different types of risks which can affect a software project:

1. Technology risks: Software or Hardware technologies that are used to develop the system.
2. People risks: Risks that are connected with the person in the development team.
3. Organizational risks: Organizational environment where the software is being developed.
4. Tools risks: Software tools and other support software used to create the system.
5. Requirement risks: Changes to customer requirement & process of managing the requirements change.
6. Estimation risks: Management of estimation resources required to build the system

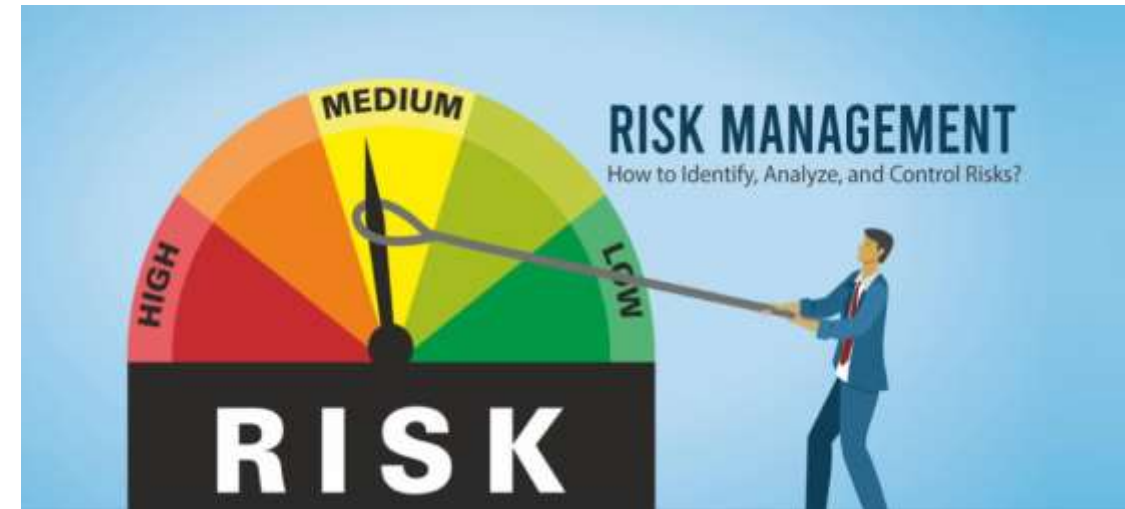
Risk Analysis & Projection or Prioritization

In Risk Analysis Process:

1. Identifying the problems causing risk in projects.
2. Identifying the probability of occurrence of problem.
3. Identifying the impact of problem.

► **The probability of a risk can be categorized as:**

1. **Very Low (0-10%):** Tolerable Risk (No harm)
2. **Low (10-25%):** Low Risk (Minor effect)
3. **Moderate (25-50%):** Medium Risk (Impact on Time)
4. **High (50-75%):** High Risk (Impact on Time & Budget)
5. **Very high (+75%):** Intolerable Risk (Impact on Output, Time, Budget & Performance)



Risk Control

- It is the process of managing risks to achieve desired outcomes.

1. Risk Planning:

- The risk planning technique considers all of the significant risks that have been identified and develop strategies to mitigate them.



► There are three main methods to plan for risk management:

- 1. Avoid the risk:** Discussing with client to change requirements, Decrease scope of work, Giving incentives to engineers to avoid the risk of human resources turnover etc.
- 2. Transfer the risk:** The risky element developed by third party. Buying insurance cover etc.
- 3. Risk reduction:** This means planning method to include the loss due to risk. If there is a risk that some key personnel might leave new recruitment can be planned.

Risk Control

2. Risk Monitoring:

- This is an ongoing process throughout the project and requires continuous evaluation and assessment of potential risks.
- Any changes in the assumptions made about risks should be identified and appropriate actions are taken to manage those risks.

3. Risk Resolution:

- This process ensures that the project stays on track and risks are controlled within acceptable levels.
- The effectiveness of risk resolution depends on the accuracy of risk identification, analysis, and planning of risk solving.
- It has ability to respond promptly and effectively to any issues that arise during the project.

About RMMM

- A risk management technique is usually seen in the software Project plan.
- This can be divided into Risk Mitigation, Monitoring and Management Plan (RMMM).
- The project manager typically uses this RMMM plan as part of the overall project plan.
- Risk is documented with the help of a Risk Information Sheet (RIS).
- RIS maintain Risk ID. Date. Probability, Impact. Description, Avoidance, Monitoring, Management plan & Current status.
- This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching and other analysis.



Risk Mitigation

- Risk Mitigation is a technique for avoiding risks (Risk Avoidance).
- It is proactive approach. Apply before risk have generate.

►Steps for mitigating the risks as follows:

1. Communicate with concerned staff to find off probable risk.
2. Removing all causes that are the reason for risk creation.
3. Develop a policy in an organization which will help to continue the project.
4. Controlling the corresponding documents from time to time.
5. Conducting timely reviews to speed up the work.



Risk Monitoring

- Risk monitoring is an activity used to track a project's progress.
- Performed by Project Manager.

► Objectives of Risk Monitoring Process:

1. To check if predicted risks occur or not.
2. To ensure proper application of risk avoidance are apply or not.
3. To gather information for future risk assessments.
4. To determine which risks generate which problems throughout the project.

```
If (RISK is avoidable)
{
  ➤ Risk Mitigation.      //(1)
  ➤ Risk Monitoring.     //(2)
}
else                        //(i.e. RISK is unavoidable)
{
  ➤ Risk Management.    //(3)
}
```

Risk Management

- It is reactive approach, Apply after risk have generate.
- It assumes that the mitigation activity failed, and the risk generate in reality.
- This task is done by Project manager. They will solve generated risk.
- If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks

Example:

Consider a scenario that many people leaving the organization but

- If sufficient additional staff is available
- If current development activity knows to everybody in the team
- If systematic documentation available

Then any new employee easily understand & start current development activity.



RMMM Plan Example

- Risk generated: Late Delivery of Project

1. Mitigation (Avoid Risk):

- Before development apply some precautionary measures.
- Developer already knew Project will be complete in 20 days. But he told customer Project will complete in 30 days.

2. Monitoring:

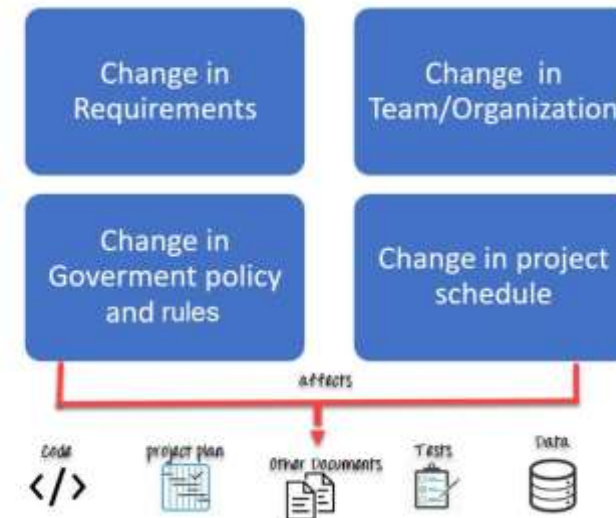
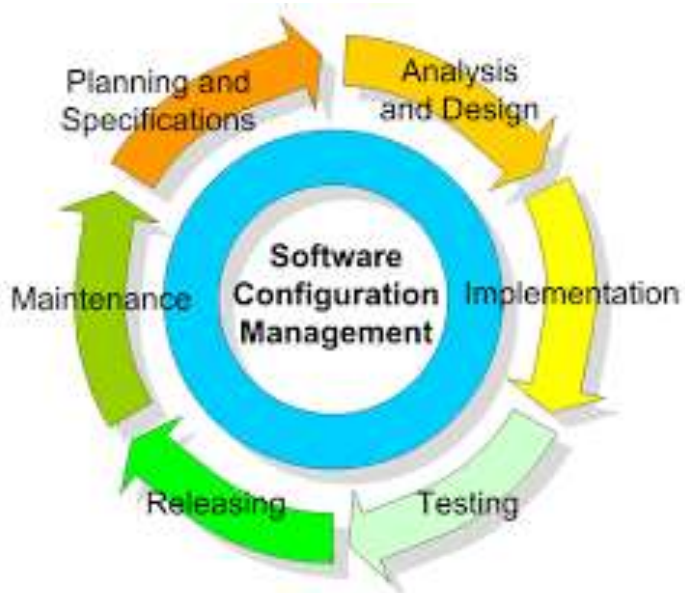
- To develop project schedule. Mentioned start & end date of Project. Within 20 to 30 days.

3. Management:

- Project not completed within deadline then negotiate with customers.
- Ask some extra time or give some additional features etc.

What is SCM?

- Software Configuration Management (SCM) is process to systematically manage, organize, & control changes in documents, codes & other entities during the Software Development Life Cycle.
- The primary goal is to increase productivity with minimal mistakes.
- SCM Tools: Puppet, ConfigHub, Saltstack, Ansible, Git, BitBucket, Docker & CHEF etc.

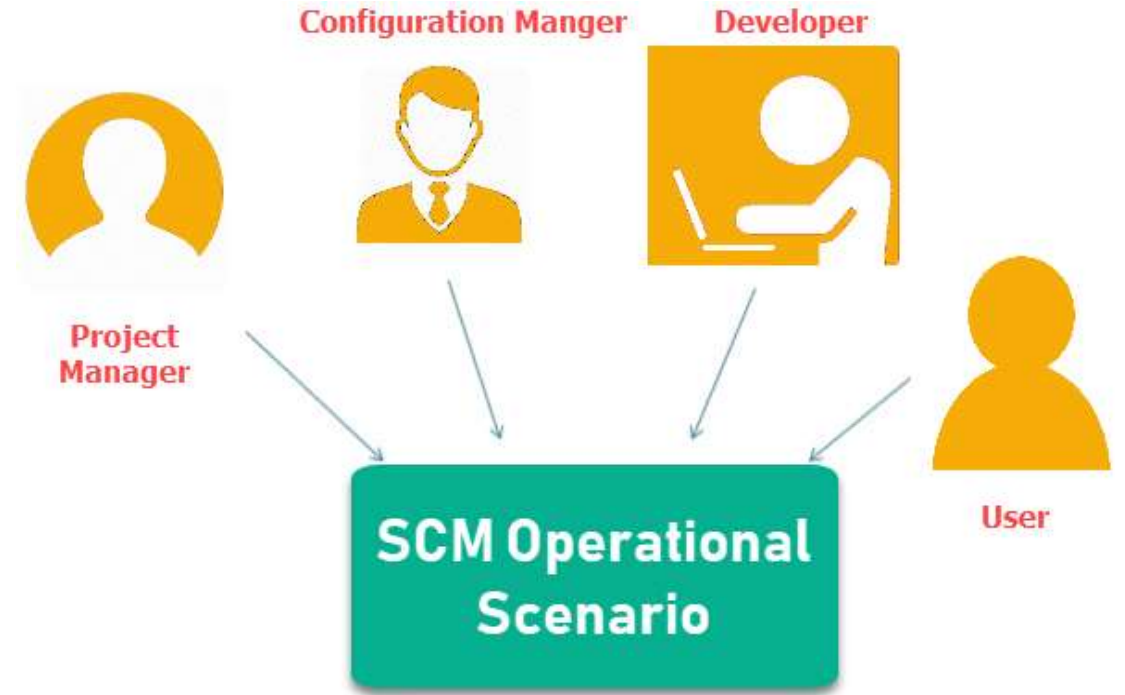


Importance of SCM

1. It ensure changes to software system are properly planned, tested & integrated into the final product.
2. Helps teams to collaborate & coordinate their work, everyone working from the same version of the software system.
3. It manage & track different versions of the system and to revert to earlier versions if necessary.
- 4 . It ensure that software systems can be easily replicated & distributed to other environments such as test, production & customer sites.
5. It improve quality & reliability of software systems, as well as increase efficiency and reduce the risk of errors.



SCM Process



SCM Process

1. Planning & Identification:

- This method determining the **scope** of the software system.
- This is accomplished by having **meetings and brainstorming sessions** with your team.
- Identifying items like test cases, specification requirements, modules & schedule time.
- Identifying each computer software configuration items in the process.
- Group basic details of why, when and what changes will be made and who will be in charge of making them

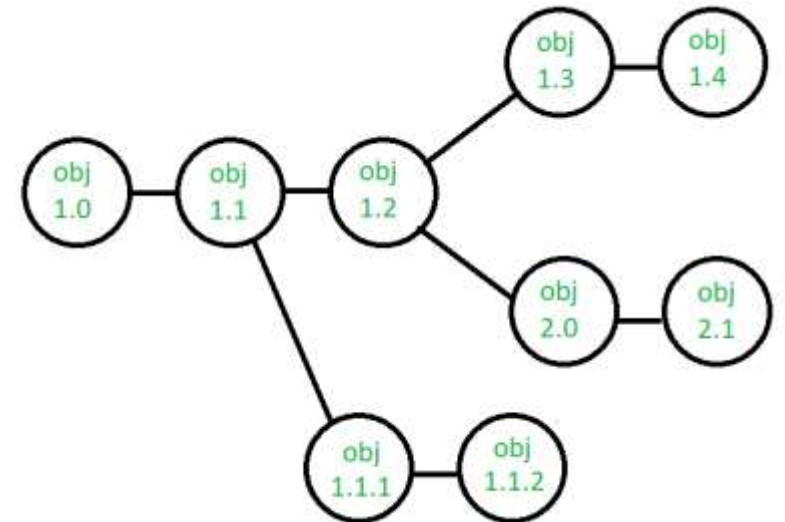
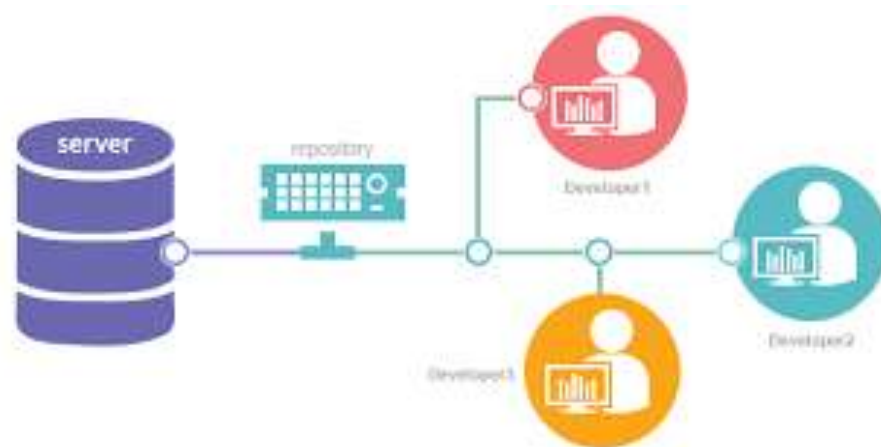
Examples:

- 1. Instead of naming a File login.php it should be named login v1.2.php where v1.2 stands for the version number of the file
- 2. Instead of naming folder "Code" it should be named "Code_D" where D represents code should be backed up daily.

SCM Process

2. Version Control Process or Baselines:

- The aim of this step is to **control the alteration and modification** done to the product.
- It handle different version of configuration objects that are generated during the software process.
- Also **focuses on developing way to track the hierarchy of different versions of the software.**
- Developing standardized label scheme for all products, revisions and files so that everyone is on the same page.
- **Examples:**



SCM Process

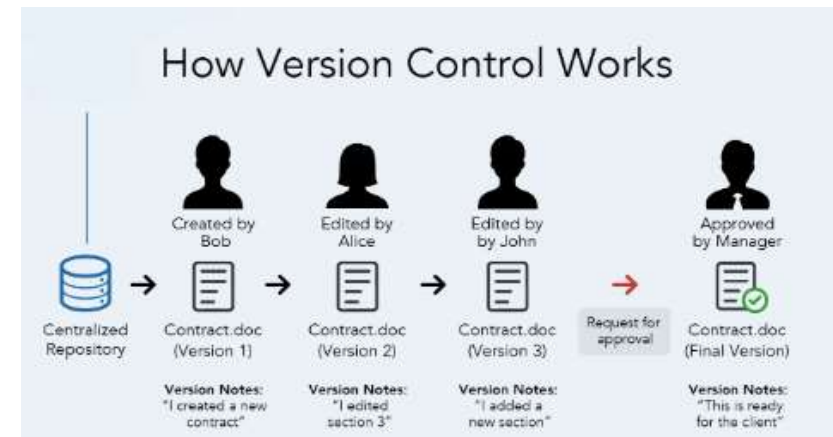
3. Change Control Process:

This method used to ensure that any changes that are made are consistent with the rest of the project.

Examples: To add or edit various configuration items, Change user permissions or Changing requirements of clients.

Process:

1. Software Team send changes to the Software Configuration Manager (SCM).
2. SCM checking examining the overall impact they will have on the project.
3. Making approved changes or explaining why change requests were denied to the team.
4. If it is approved them implement all necessary changes.
5. After that Review or Reporting it.



SCM Process

4. Configuration Auditing Process:

- This process is used to ensure that application will develop as per the project plan and test/verify the application as per scope.
- The audit confirms the completeness, correctness and consistency of modified items in the SCM system and track action items from the audit to closure.
- It mentioned what is new in each version and why the changes were necessary.
- It ensures that what is built is what is delivered.



SCM Process

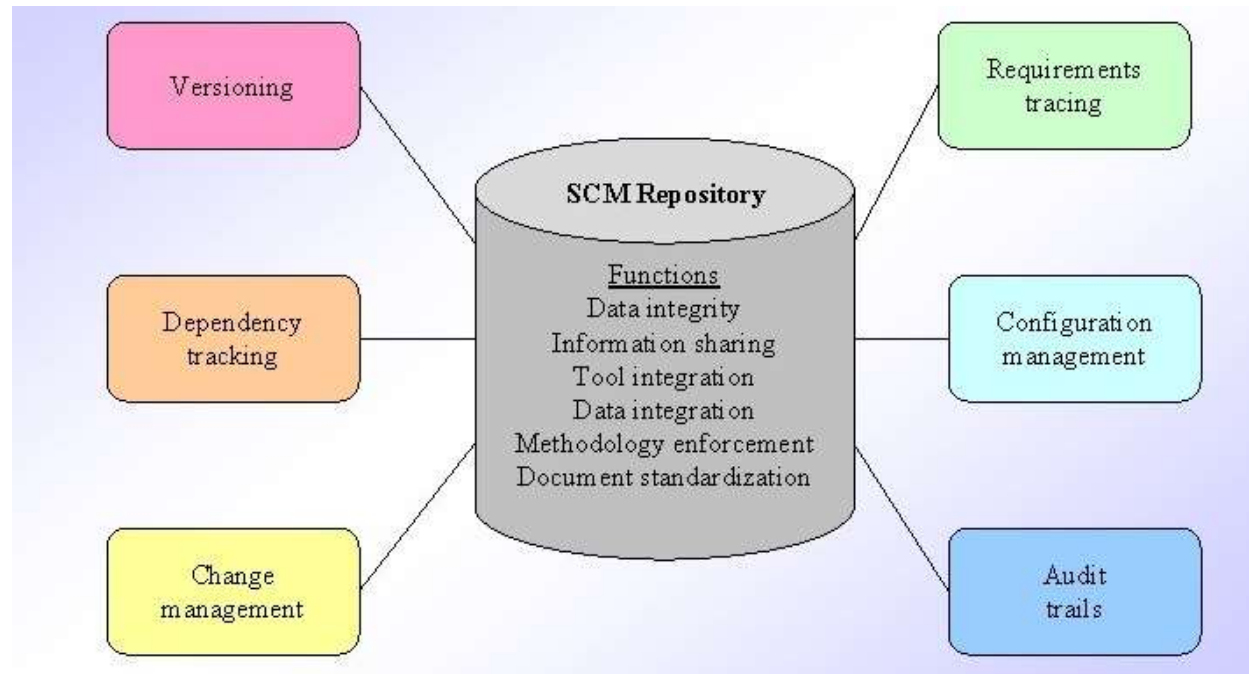
5. Review and Status Reporting Process:

- It is a technical review on the Application workflow, Process, Configuration items and Change requests etc.
- It generate the accurate status report in every phase of SDLC process.
- Configuration Status report provide to the Developers, Testers, End users, Customers and Stakeholders.
- It develop some application-related documents like User manual, Installation process guide, Configuration guide, Do's and Don't Do's etc.



SCM Repository

- SCM Repository is a set of control process & data structure that allow software team to manage change in effective manner.
- It manage version control, change control & release control process.



SCM Repository Functions

1. **Data Sharing:** Ensure consistent & accurate information. Validates entries.
2. **Information Sharing:** Manage & Control multi-user access, Share with among developers.
3. **Tool Integration:** Establish data models by using SE tools.
4. **Data Integration:** SCM task can be perform using one or more resources.
5. **Methodology Enforcement:** Define Entity Relationship model of repository.
6. **Document Standardization:** There is standard approach for creating Software Engineering documents.

SCM Repository Tools

- 1. Versioning:** Save & retrieve all repository object based on there version number.
- 2. Dependency Tracking & Change Management:** Track & manage to the changes in relationship of all objects in repository. Ex. UML diagram.
- 3. Requirement Tracing:** Trace design & construction of components & there deliverable result (Forward Tracing). As per the work product, which requirement is responsible for which feature. (Backward Tracing).
- 4. Configuration Management:** Track a series of configuration representing specific project milestone or production release.
- 5. Audit Trails:** Establish information when, why & by whom changes are made in the repository