2. Linear Software Project Estimation

Different Method of Cost Estimation:

COCOMO Model:

Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981.COCOMO is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

The necessary steps in this model are:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
2. Determine a set of 15 multiplying factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size. To determine the initial effort E_i in person-months the equation used is of the type is shown below

E_i=a

***(KDLOC)b**

The value of the constant a and b are depends on the project type.

In COCOMO, projects are categorized into three types:

1. Organic
2. Semidetached
3. Embedded

1.Organic: A development project can be treated of the organic type, if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar methods of projects. **Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.**

2. Semidetached: A development project can be treated with semidetached type if the development consists of a mixture of experienced and inexperienced staff. Team members may have finite experience in related systems but may be unfamiliar with some aspects of the order being developed. **Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.**

3. Embedded: A development project is treated to be of an embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational method exist. **For Example:** ATM, Air Traffic control.

For three product categories, Bohem provides a different set of expression to predict effort (in a unit of person month)and development time from the size of estimation in KLOC(Kilo Line of code) efforts estimation takes into account the productivity loss due to holidays, weekly off, coffee breaks, etc.

According to Boehm, software cost estimation should be done through three stages:

1. Basic Model
2. Intermediate Model
3. Detailed Model

2.1.1. Basic COCOMO Model:

The basic COCOMO model provide an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model:

Effort=a_1*(KLOC) a_2 PM
Tdev=b_1*(efforts)b_2 Months

Where

KLOC is the estimated size of the software product indicate in Kilo Lines of Code,

a_1,a_2,b_1,b_2 are constants for each group of software products,

Tdev is the estimated time to develop the software, expressed in months,

Effort is the total effort required to develop the software product, expressed in **person months (PMs)**.

Estimation of development effort

For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic: Effort = 2.4(KLOC) 1.05 PM

Semi-detached: Effort = 3.0(KLOC) 1.12 PM

Embedded: Effort = 3.6(KLOC) 1.20 PM

Estimation of development time

For the three classes of software products, the formulas for estimating the development time based on the effort are given below:

Organic: Tdev = 2.5(Effort) 0.38 Months

Semi-detached: Tdev = 2.5(Effort) 0.35 Months

Embedded: Tdev = 2.5(Effort) 0.32 Months

Some insight into the basic COCOMO model can be obtained by plotting the estimated characteristics for different software sizes. Fig shows a plot of estimated effort versus product size. From fig, we can observe that the effort is somewhat superliner in the size of the software product. Thus, the effort required to develop a product increases very rapidly with project size.

The development time versus the product size in KLOC is plotted in fig. From fig it can be observed that the development time is a sub linear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately. This can be explained by the fact that for larger products, a larger number of activities which can be carried out concurrently can be identified. The parallel activities can be carried out simultaneously by the engineers. This reduces the time to complete the project. Further, from fig, it can be observed that the development time is roughly the same for all three categories of products. For example, a 60 KLOC program can be developed in approximately 18 months, regardless of whether it is of organic, semidetached, or embedded type.

From the effort estimation, the project cost can be obtained by multiplying the required effort by the manpower cost per month. But, implicit in this project cost computation is the assumption that the entire project cost is incurred on account of the manpower cost alone. In addition to manpower cost, a project would incur costs due to hardware and software required for the project and the company overheads for administration, office space, etc.

It is important to note that the effort and the duration estimations obtained using the COCOMO model are called a nominal effort estimate and nominal duration estimate. The term nominal implies that if anyone tries to complete the project in a time shorter

than the estimated duration, then the cost will increase drastically. But, if anyone completes the project over a longer period of time than the estimated, then there is almost no decrease in the estimated cost value.

Example1: Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded.

Solution: The basic COCOMO equation takes the form:

Effort=a_1*(KLOC) a_2 PM
Tdev=b_1*(efforts)b_2 Months
Estimated Size of project= 400 KLOC

(i)Organic Mode

E = 2.4 * (400)1.05 = 1295.31 PM
D = 2.5 * (1295.31)0.38=38.07 PM

(ii)Semidetached Mode

E = 3.0 * (400)1.12=2462.79 PM
D = 2.5 * (2462.79)0.35=38.45 PM

(iii) Embedded Mode

E = 3.6 * (400)1.20 = 4772.81 PM
D = 2.5 * (4772.8)0.32 = 38 PM

Example2: A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the Effort, development time, average staff size, and productivity of the project.

Solution: The semidetached mode is the most appropriate mode, keeping in view the size, schedule and experience of development time.

Hence E=3.0(200)1.12=1133.12PM
D=2.5(1133.12)0.35=29.3PM P = 176 LOC/PM

2. Intermediate Model:

The basic Cocomo model considers that the effort is only a function of the number of lines of code and some constants calculated according to the various software systems. The intermediate COCOMO model recognizes these facts and refines the initial estimates obtained through the basic COCOMO model by using a set of 15 cost drivers based on various attributes of software engineering.

Classification of Cost Drivers and their attributes:

(i) Product attributes -

- Required software reliability extent
- Size of the application database
- The complexity of the product

Hardware attributes -

- Run-time performance constraints
- Memory constraints
- The volatility of the virtual machine environment
- Required turnabout time

Personnel attributes -

- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience

Project attributes -

- Use of software tools
- Application of software engineering methods
- Required development schedule

The cost drivers are divided into four categories:

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	..

3.Detailed COCOMO Model:

Detailed COCOMO incorporates all qualities of the standard version with an assessment of the cost driver?s effect on each method of the software engineering process. The detailed model uses various effort multipliers for each cost driver property. In detailed cocomo, the whole software is differentiated into multiple modules, and then we apply COCOMO in various modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

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

The effort is determined as a function of program estimate, and a set of cost drivers are given according to every phase of the software lifecycle.

2.1.2 Delphi Cost Estimation

Delphi Method is a structured communication technique, originally developed as a systematic, interactive forecasting method which relies on a panel of experts. The experts answer questionnaires in two or more rounds. After each round, a facilitator

provides an anonymous summary of the experts' forecasts from the previous round with the reasons for their judgments. Experts are then encouraged to revise their earlier answers in light of the replies of other members of the panel.

It is believed that during this process the range of answers will decrease and the group will converge towards the "correct" answer. Finally, the process is stopped after a predefined stop criterion (e.g. number of rounds, achievement of consensus, and stability of results) and the mean or median scores of the final rounds determine the results.

Delphi Method was developed in the 1950-1960s at the RAND Corporation.

Wideband Delphi Technique

In the 1970s, Barry Boehm and John A. Farquhar originated the Wideband Variant of the Delphi Method. The term "wideband" is used because, compared to the Delphi Method, the Wideband Delphi Technique involved greater interaction and more communication between the participants.

In Wideband Delphi Technique, the estimation team comprise the project manager, moderator, experts, and representatives from the development team, constituting a 3-7 member team. There are two meetings –

- Kickoff Meeting
- Estimation Meeting

Wideband Delphi Technique – Steps

Step 1 − Choose the Estimation team and a moderator.

Step 2 − The moderator conducts the kickoff meeting, in which the team is presented with the problem specification and a high level task list, any assumptions or project constraints. The team discusses on the problem and estimation issues, if any. They also decide on the units of estimation. The moderator guides the entire discussion, monitors time and after the kickoff meeting, prepares a structured document containing problem specification, high level task list, assumptions, and the units of estimation that are decided. He then forwards copies of this document for the next step.

Step 3 − Each Estimation team member then individually generates a detailed WBS, estimates each task in the WBS, and documents the assumptions made.

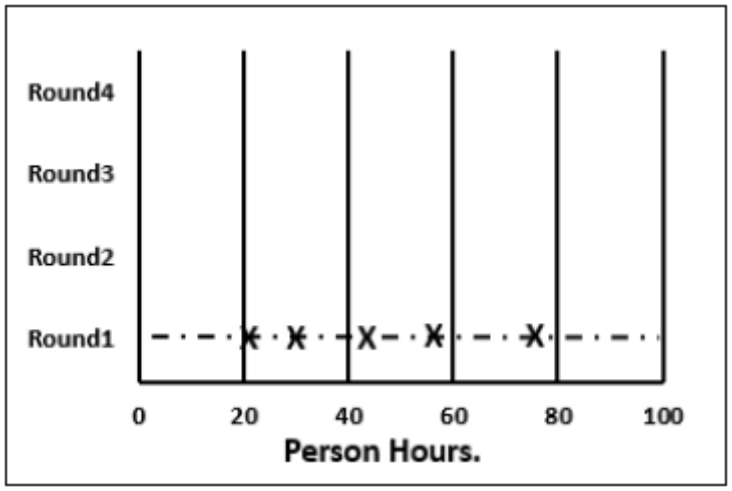
Wideband Delphi Estimation Sheet						
Project: <Project Name>				**Estimation Units:**	Person Hours	
Estimation Team Member: <Name>				**Date:**	<MM-DD-YY>	
Task	**Initial Estimate**	**Change 1**	**Change 2**	**Change 3**	**Change 4**	**Final**
Task1	n_1					
Task2	n_2					
Task3	n_3					
Task4	n_4					
Task5	n_5					
Task6	n_6					
Task7	n_7					
Task8	n_8					
Net Change						
Total	$\sum n_i$					

Step 4 − The moderator calls the Estimation team for the Estimation meeting. If any of the Estimation team members respond saying that the estimates are not ready, the moderator gives more time and resends the Meeting Invite.

Step 5 − The entire Estimation team assembles for the estimation meeting.

Step 5.1 − At the beginning of the Estimation meeting, the moderator collects the initial estimates from each of the team members.

Step 5.2 − He then plots a chart on the whiteboard. He plots each member's total project estimate as an X on the Round 1 line, without disclosing the corresponding names. The Estimation team gets an idea of the range of estimates, which initially may be large.



Step 5.3 − Each team member reads aloud the detailed task list that he/she made, identifying any assumptions made and raising any questions or issues. The task estimates are not disclosed.

The individual detailed task lists contribute to a more complete task list when combined.

Step 5.4 − The team then discusses any doubt/problem they have about the tasks they have arrived at, assumptions made, and estimation issues.

Step 5.5 − Each team member then revisits his/her task list and assumptions, and makes changes if necessary. The task estimates also may require adjustments based on the discussion, which are noted as +N Hrs. for more effort and –N Hrs. for less effort.

The team members then combine the changes in the task estimates to arrive at the total project estimate.

Wideband Delphi Estimation Sheet						
Project: <Project Name>				**Estimation Units:**	Person Hours	
Estimation Team Member:	<Name>			**Date:**	<MM-DD-YY>	
Task	**Initial Estimate**	**Change 1**	**Change 2**	**Change 3**	**Change 4**	**Final**
Task1	n_1	-1				
Task2	n_2	-2				
Task3	n_3	-4				
Task4	n_4	5				
Task5	n_5	0				
Task6	n_6	0				
Task7	n_7	2				
Task8	n_8	-3				
Net Change		**-3**				
Total	$\sum n_i$	$\sum n_i$**-3**				

Step 5.6 − The moderator collects the changed estimates from all the team members and plots them on the Round 2 line.

In this round, the range will be narrower compared to the earlier one, as it is more consensus based.

Step 5.7 − The team then discusses the task modifications they have made and the assumptions.

Step 5.8 − Each team member then revisits his/her task list and assumptions, and makes changes if necessary. The task estimates may also require adjustments based on the discussion.

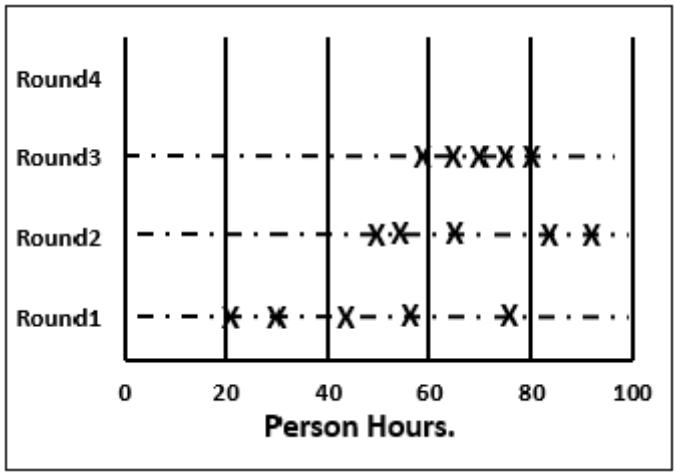
The team members then once again combine the changes in the task estimate to arrive at the total project estimate.

Step 5.9 − The moderator collects the changed estimates from all the members again and plots them on the Round 3 line.

Again, in this round, the range will be narrower compared to the earlier one.

Step 5.10 − Steps 5.7, 5.8, 5.9 are repeated till one of the following criteria is met −

- Results are converged to an acceptably narrow range.
- All team members are unwilling to change their latest estimates.
- The allotted Estimation meeting time is over.



tep 6 − The Project Manager then assembles the results from the Estimation meeting.

Step 6.1 − He compiles the individual task lists and the corresponding estimates into a single master task list.

Step 6.2 − He also combines the individual lists of assumptions.

Step 6.3 − He then reviews the final task list with the Estimation team.

Advantages and Disadvantages of Wideband Delphi Technique

Advantages

- Wideband Delphi Technique is a consensus-based estimation technique for estimating effort.
- Useful when estimating time to do a task.
- Participation of experienced people and they individually estimating would lead to reliable results.
- People who would do the work are making estimates thus making valid estimates.
- Anonymity maintained throughout makes it possible for everyone to express their results confidently.
- A very simple technique.
- Assumptions are documented, discussed and agreed.

Disadvantages

- Management support is required.
- The estimation results may not be what the management wants to hear.

2.2 Function Point Analysis

Function Point Analysis

Function Point Analysis was initially developed by Allan J. Albercht in 1979 at IBM and it has been further modified by the International Function Point Users Group (IFPUG).

Initial Definition given by Allan J. Albrecht:

PA provides standardized method to functionally size the software work product. This work product is the output of software new development and improvement projects for subsequent releases. It is the software which is relocated to the production application at project implementation. It measures functionality from the users point of view i.e. on the basis of what the user requests and receives in return.

Function Point Analysis (FPA) is a method or set of rules of Functional Size Measurement. It assesses the functionality delivered to its users, based on the user's external view of the functional requirements. It measures the logical view of an application not the physically implemented view or the internal technical view.

The Function Point Analysis technique is used to analyse the functionality delivered by software and *Unadjusted Function Point (UFP)* is the unit of measurement.

Objectives of FPA:

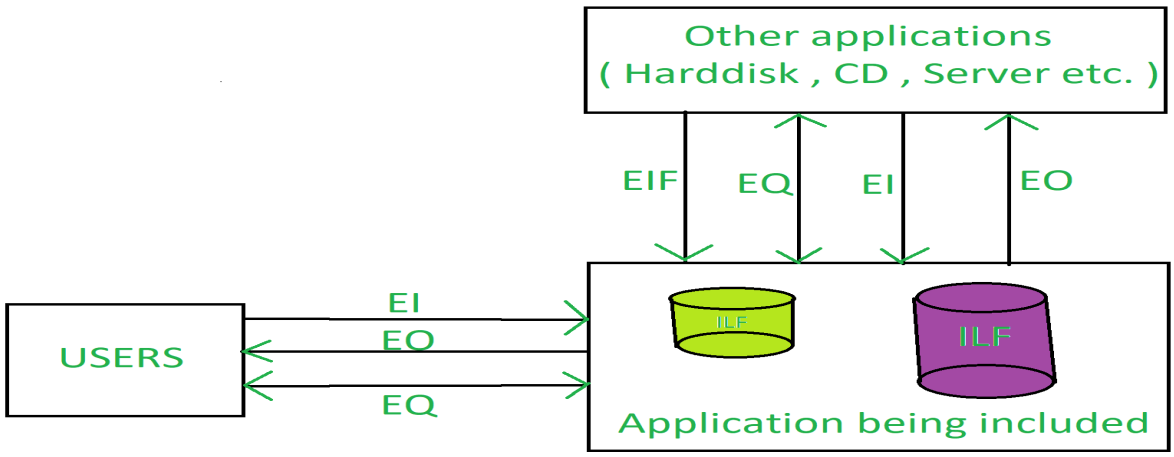
1. The objective of FPA is to measure functionality that the user requests and receives.
2. The objective of FPA is to measure software development and maintenance independently of technology used for implementation.
3. It should be simple enough to minimize the overhead of the measurement process.
4. It should be a consistent measure among various projects and organizations.

Types of FPA:

1. **Transactional Functional Type –**
    - **(i) External Input (EI):** EI processes data or control information that comes from outside the application's boundary. The EI is an elementary process.
    - **(ii) External Output (EO):** EO is an elementary process that generates data or control information sent outside the application's boundary.
    - **(iii) External Inquiries (EQ):** EQ is an elementary process made up of an input-output combination that results in data retrieval.
2. **Data Functional Type –**

- **(i) Internal Logical File (ILF):** A user identifiable group of logically related data or control information maintained within the boundary of the application.

(ii) External Interface File (EIF): A group of user recognizable logically related data allusion to the software but maintained within the boundary of another software



Benefits of FPA:

- FPA is a tool to determine the size of a purchased application package by counting all the functions included in the package.
- It is a tool to help users discover the benefit of an application package to their organization by counting functions that specifically match their requirements.
- It is a tool to measure the units of a software product to support quality and productivity analysis.
- It s a vehicle to estimate cost and resources required for software development and maintenance.
- It is a normalization factor for software comparison.

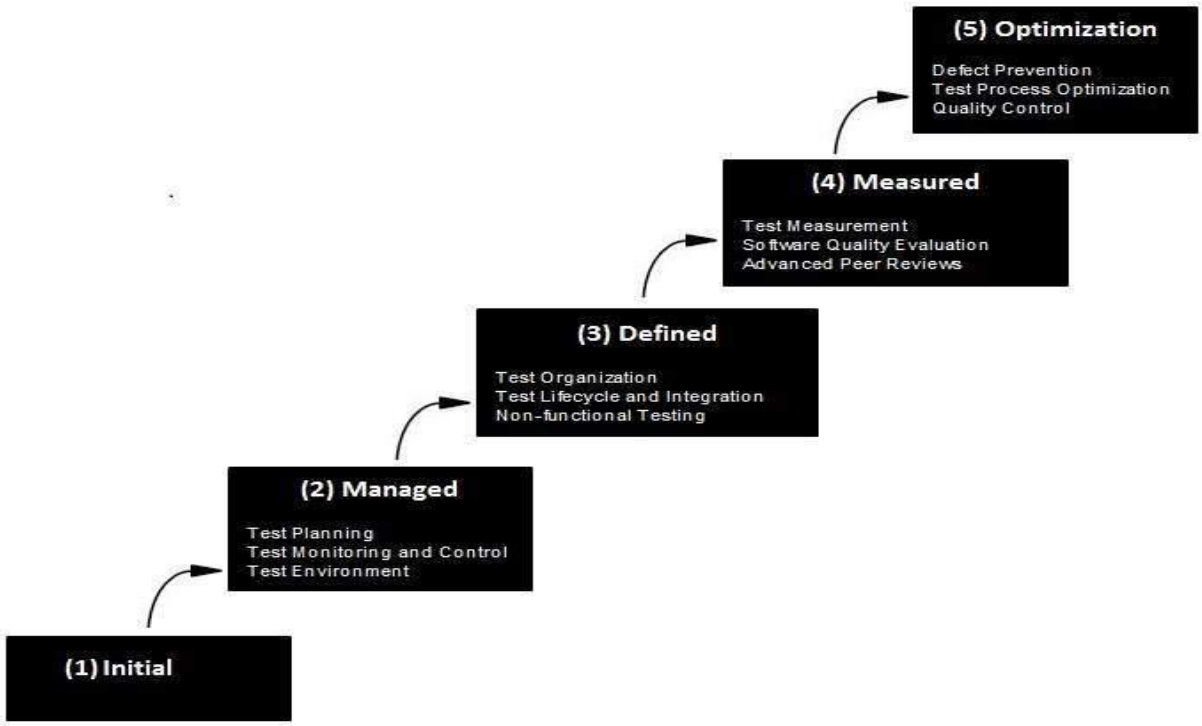
2.3 The SEI Capability Maturity Model CMM

What is Capability Maturity Model?

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level,

the better the software development process, hence reaching each level is an expensive and time-consuming process.

Levels of CMM



- **Level One :Initial** - The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.
- **Level Two: Repeatable** - This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.
- **Level Three: Defined** - The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an approved, tailored version of the organization's standard software process for developing,testing and maintaining the application.
- **Level Four: Managed** - Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance. At

this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.

- **Level Five: Optimizing** - The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

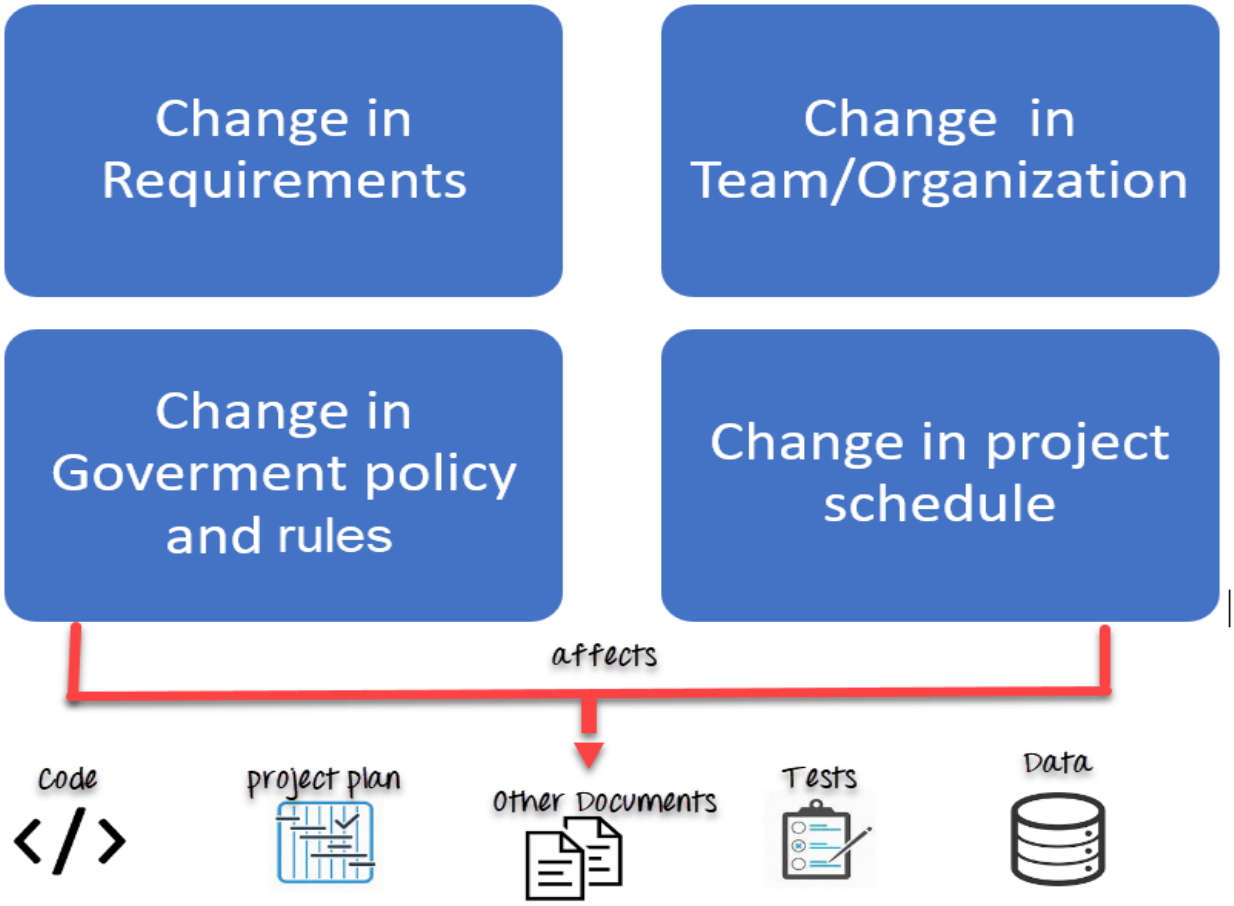
2.4 Software Configuration management

In Software Engineering, **Software Configuration Management(SCM)** is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

Why do we need Configuration management?

The primary reasons for Implementing Technical Software Configuration Management System are:

- There are multiple people working on software which is continually updating
- It may be a case where multiple version, branches, authors are involved in a software config project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders
- SCM process is also beneficial to control the costs involved in making changes to a system



Any change in the software configuration Items will affect the final product. Therefore, changes to configuration items need to be controlled and managed.

Tasks in SCM process

- Configuration Identification
- Baselines
- Change Control
- Configuration Status Accounting
- Configuration Audits and Reviews

Configuration Identification:

Configuration identification is a method of determining the scope of the software system. With the help of this step, you can manage or control something even if you don't know what it is. It is a description that contains the CSCI type (Computer Software Configuration Item), a project identifier and version information.

Activities during this process:

- Identification of configuration Items like source code modules, test case, and requirements specification.
- Identification of each CSCI in the SCM repository, by using an object-oriented approach
- The process starts with basic objects which are grouped into aggregate objects. Details of what, why, when and by whom changes in the test are made
- Every object has its own features that identify its name that is explicit to all other objects
- List of resources required such as the document, the file, tools, etc.

Example:

Instead of naming a File login.php its should be named login_v1.2.php where v1.2 stands for the version number of the file

Instead of naming folder "Code" it should be named "Code_D" where D represents code should be backed up daily.

Baseline:

A baseline is a formally accepted version of a software configuration item. It is designated and fixed at a specific time while conducting the SCM process. It can only be changed through formal change control procedures.

Activities during this process:

- Facilitate construction of various versions of an application
- Defining and determining mechanisms for managing various versions of these work products
- The functional baseline corresponds to the reviewed system requirements
- Widely used baselines include functional, developmental, and product baselines

In simple words, baseline means ready for release.