UNIT 2: Software Project Management Concepts

# **Software Project Management (SPM**)

is a proper way of planning and leading software projects. It is a part of project management in which software projects are planned, implemented, monitored, and controlled.

## Need for Software Project Management:

Software is a non-physical product. Software development is a new stream in business and there is very little experience in building software products. Most of the software products are made to fit clients’ requirements. The most important is that the basic technology changes and advances so frequently and rapidly that experience of one product may not be applied to the other one. Such type of business and environmental constraints increase risk in software development hence it is essential to manage software projects efficiently. It is necessary for an organization to deliver quality products, keep the cost within the client’s budget constrain and deliver the project as per schedule. Hence in order, software project management is necessary to incorporate user requirements along with budget and time constraints.

# Software Project Management consists of Several Different Types of Management

1. **Conflict Management:** Conflict management is the process to restrict the negative features of conflict while increasing the positive features of conflict. The goal of conflict management is to improve learning and group results including efficacy or performance in an organizational setting. Properly managed conflict can enhance group results.
2. **Risk Management:** Risk management is the analysis and identification of risks that is followed by synchronized and economical implementation of resources to minimize, operate and control the possibility or effect of unfortunate events or to maximize the realization of opportunities.
3. **Requirement Management:** It is the process of analyzing, prioritizing, tracking, and documenting requirements and then supervising change and communicating to pertinent stakeholders. It is a continuous process during a project.
4. **Change Management:** Change management is a systematic approach for dealing with the transition or transformation of an organization’s goals, processes, or technologies. The purpose of change management is to execute strategies for effecting change, controlling change, and helping people to adapt to change.
5. **Software Configuration Management:** Software configuration management is the process of controlling and tracking changes in the software, part of the larger cross-disciplinary field of configuration management. Software configuration management includes revision control and the inauguration of baselines
6. **Release Management:** Release Management is the task of planning, controlling, and scheduling the build-in deploying releases. Release management ensures that the organization delivers new and enhanced services required by the customer while protecting the integrity of existing services

# The Management Spectrum | 4 P’s in Software Project Planning

For properly building a product, there’s a very important concept that we all should know in software project planning while developing a product. There are 4 critical components in software project planning which are known as the 4P’s namely

* Product
* Process
* People
* Project

These components play a very important role in your project that can help your team meet its goals and objectives. Now, Let’s dive into each of them a little in detail to get a better understanding:

### P**eople**

The most important component of a product and its successful implementation is human resources. In building a proper product, a well-managed team with clear-cut roles defined for each person/team will lead to the success of the product. We need to have a good team in order to save our time, cost, and effort. Some assigned roles in software project planning are **project manager, team leaders, stakeholders, analysts, and other IT professional**s. Managing people successfully is a tricky process which a good project manager can do.

### Product

As the name inferred, this is the deliverable or the result of the project. The project manager should clearly define the product scope to ensure a successful result, control the team members, as well technical hurdles that he or she may encounter during the building of a product. The product can consist of both tangible or intangible things such as shifting the company to a new place or getting new software in a company.

### Process

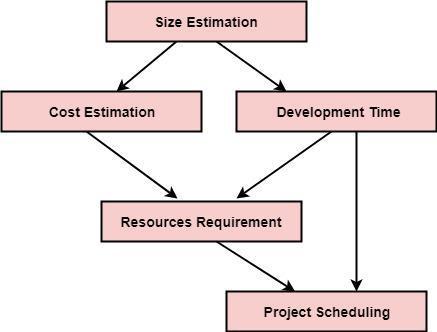
In every planning, a clearly defined process is the key to the success of any product. It regulates how the team will go about its development in the respective time period. The Process has several steps involved like, documentation phase, implementation phase, deployment phase, and interaction phase.

## Project

The last and final P in software project planning is Project. It can also be considered as a blueprint of process. In this phase, the project manager plays a critical role. They are responsible to guide the team members to achieve the project’s target and objectives, helping & assisting them with issues, checking on cost and budget, and making sure that the project stays on track with the given deadlines.

# 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.



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.

# Software Cost Estimation

For any new software project, it is necessary to know how much it will cost to develop and how much development time will it take. These estimates are needed before development is initiated, but how is this done? Several estimation procedures have been developed and are having the following attributes in common.

1. Project scope must be established in advanced.
2. Software metrics are used as a support from which evaluation is made.
3. The project is broken into small PCs which are estimated individually. To achieve true cost & schedule estimate, several option arise.
4. Delay estimation Used symbol decomposition techniques to generate project cost and schedule estimates.
5. Acquire one or more automated estimation tools.

# Cocomo

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e **number of Lines of Code**. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

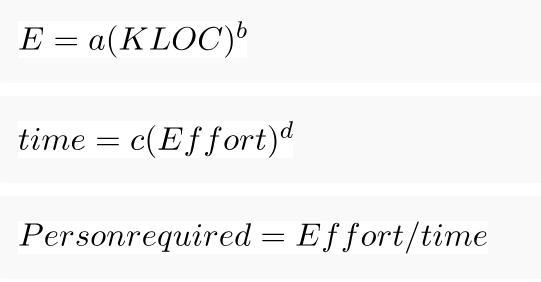
* **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
* **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is

measured in the units of time such as weeks, and months. Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of the constant to be used in subsequent calculations.

These characteristics pertaining to different system types are mentioned below. Boehm’s definition of organic, semidetached, and embedded systems:

1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered Semi-Detached types.
3. **Embedded –** A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.
4. Basic COCOMO Model
5. Intermediate COCOMO Model
6. Detailed COCOMO Model

**1. Basic Model –**



The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a,b,c, and d for the Basic Model for the different categories of the system:

| **Software Projects** | **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 |
|  |  |  |  |  |

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. The development time is measured in months. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, and expertise is taken into account, henceforth the estimate is rough.

1. **Intermediate Model –** The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system’s effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, and Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes
2. such drivers for cost estimation. Classification of Cost Drivers and their Attributes:

**(i) Product attributes –**

∙ Required software reliability extent ∙ Size of the application database

∙ The complexity of the product

∙ Run-time performance constraints ∙ Memory constraints

∙ The volatility of the virtual machine environment ∙ Required turnabout time

∙ Analyst capability

∙ Software engineering capability ∙ Applications experience

∙ Virtual machine experience

∙ Programming language experience ∙ Use of software tools

∙ Application of software engineering methods ∙ Required development schedule

1. **Detailed Model –** Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver’s impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply **C**OCOMO 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

**Advantages of the COCOMO model:**

1. Provides a systematic way to estimate the cost and effort of a software project.
2. Can be used to estimate the cost and effort of a 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. Can be used to evaluate the feasibility of a software project by estimating

the cost and effort required to complete it.

1. 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.
2. Does not take into account the specific characteristics of the development team, which can have a significant impact on the cost and effort of a software project.
3. Does not provide a precise estimate of the cost and effort of a software project, as it is based on assumptions and averages.