Unit 5 – Chapter 13

Software Project Management (Part 1)

Prerequisites:

Software development life cycle, software testing

Unit Outcomes:

The objective of this chapter is to introduce the concept of project management and software metric calculations. At the end of this chapter, students will be able to:

- Identify various project management activities such as project planning, monitoring, and control.
- Discuss estimation of various parameters such as cost, Efforts, and schedule/duration.
- Determine various size-oriented measures and understand Halstead's software science.
- Apply function point measures, cyclomatic complexity, and control flow graphs to determine the software quality.
- Analyze the constructive cost models (COCOMO) and, resource allocation models.

13.1 Software Project Management (SPM)

SPM is an important part of software engineering from the stage of a conceptual model to the developed software stage. Any project in software engineering is under the constraints of budget and schedule planned. The success of the project varies based on the application and users, but in general, the goals of the SPM are:

- Maintain the agreed timing of the software delivery.
- Keep the project activities under the allotted budget.
- Fulfill the client's specifications
- Make sure that the development team is functioning well in a happy atmosphere.

Project management in software engineering is different than other engineering practices. In civil or mechanical engineering, the progress of the project under development is visible and the project manager can see it. Whereas in software engineering the product is intangible. A review of the progress with team members is only possible in software development. When the projects are large the project manager's experience may not be sufficient to anticipate the problems in the software development. The software processes are dynamic, unlike electrical or civil engineering project tasks. When the software to be developed is a part of a large engineering project, it becomes impossible to determine whether there will be any software development problems. SPM is managed by the **project manager**. For any project, whether it is small or big, the project manager is employed for its completion.

The different duties of the project manager are as given below.

- Project planning
- Reporting
- Risk management
- People management
- Proposal writing

13.2 SPM Activities

There are two main activities in SPM as follows.

- Project planning,
- Project monitoring and project control

13.2.1 Project planning

In this stage, the project manager is responsible for planning the project and estimating and developing the project schedule. The different task allocation to team members is done by the project manager. The manager will monitor the project development progress within the given schedule and budget. Project planning is completed before the software development activity begins. Project planning includes the following important activities.

1) Estimation of project size, cost duration, and efforts: The size of the project determines the complexity of the project in terms of the complexity of efforts and time required for software development. The cost of the project refers to how much finance is necessary to develop a project. The duration refers to the time required to complete project development. The efforts refer to the manpower and other resources required for project completion.
2) Preparing a schedule of manpower and resources such as web, database, hardware, etc.
3) Organisation and appointment of staff: In this activity, analysis is done to find out if the existing staff is sufficient for the project or if new hiring is required.
4) Configuration management and quality assurance planning.

   The different activities can be conducted in the order as shown in Figure 12.1. The size of the project is determined first. The effort estimation is connected to project staffing. The size is connected to the time or duration required for completion and scheduling. Each activity is equally important in project planning.

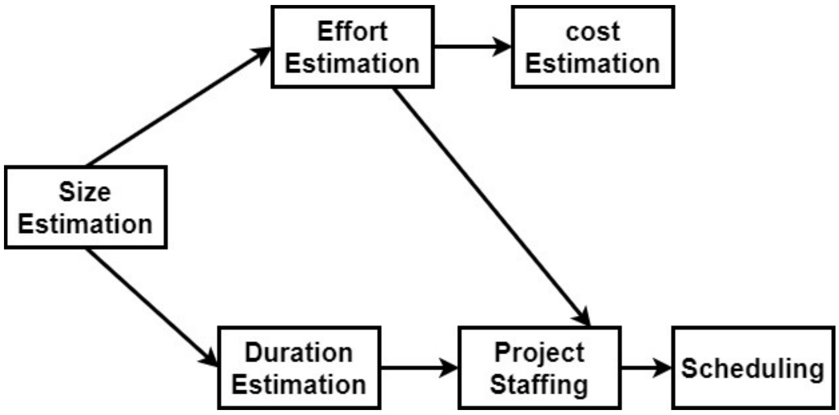

Figure 13.1 Ordering of activities in project planning

Software Project Management Plan (SPMP) Document

At the end of the project planning, a project management plan document is generated. The SPMP document includes a list of different items concerned with the activities mentioned in Table 13.1.

Name	Details
Introduction	Objectives, Functions, Performance issues, Constraints

Project estimates	Historical data used, Estimation techniques details, Cost, duration, effort estimates
Project Schedule	Work breakdown using Gantt and PERT chart
Project resource	Manpower, Hardware, and Software, highly skilled professionals
Staff organization	Team formation and structure, Management reporting
Risk Management	Risk analysis: Risk identification and Risk abatement methods
Project tracking and control plan	This plan is used for the continuous process of tracking and reviewing the project's performance.
Miscellaneous activities	Software process, testing, verification and validation plan, testing, and configuration plan

Table 13.1 Organization of SPMP Document

Some of the important SPM activities are discussed below.

13.3 Project Size Estimation:

The cost, schedule, and time of the project can be determined using the size of the project. This is the reason the size of the project must be determined very accurately. There are two important metrics commonly used in project size estimation:

13.3.1 Lines of code (LOC)

This is the simplest metric used where the project size is calculated by counting the number of source instructions in the program. The comments and the header file statements are excluded from the count. For estimation of the project size at the initial stages of development, the project size can be divided into modules and submodules, and so on.

To estimate the accurate LOC, the experience in the past and the skill of the software developer are considered. Programming skills can be divided into 3 types as shown in Table 13.2.

Skill Level	Developed Code
Efficient	Optimistic (Sopt)

Average	Most Likely (Sm)
Poor	Pessimistic (Spess)

Table 13.2 Division of Programming Skills

The LOC can be calculated using the formula:

$$S = \left(\frac{\boldsymbol{Sopt + 4Sm + Spess}}{\boldsymbol{6}}\right)$$

Example: Consider a software system, at the initial stage it is estimated that there are 5200 Sopt, 7000 Sm, and 9000 Spess. Find the following for this application.

1) The expected LOC of software
2) Consider a team of two members for software development. Determine the productivity rate. The productivity rate is given as size/effort.
3) If software development effort is expected to be two man-years, then what is the productivity rate?
4) If the duration for the software development is estimated as 6 months, then what is the average manpower?
5) If software development cost is estimated as Rs 2500 / man-month, what is the development cost of the project?

Solution:

1) $\boldsymbol{S} = \left(\frac{\boldsymbol{Sopt+4Sm+Spess}}{\boldsymbol{6}}\right)$

   = (5200 + 4* 7000 + 9000)/6

   = 7033 LOC

2) 2 men year = 24 months

   7033 LOC -> 24 months

   7033/24 -> 1 month

   = 293 LOC/man/month

3) Manpower = Effort / Duration

   24/6 = 4 months

4) Development Cost = Development Effort * Cost/man month

   = 24 * 2500

   = 60000

13.3.2 Function point (FP)

Function point was proposed in 1983 and it overcomes many pitfalls of the LOC metric. Unlike the LOC method, FP can be used to estimate LOC right from the stage of problem specification.

The main concept in the FP method is that size of the software depends on the number of functions or features included in the project. The more the number of functions or tasks in the project, the higher the size of the project. Other parameters to estimate the size are the number of files and interfaces to the project, and input/output data values.

Important Notations

FP = Function Point

UFP = Unadjusted function point

VAF = Value adjusted factor

TDI = Total degree of influence

DI = Degree of Influence

FP calculation is shown by the following steps

- Compute the Unadjusted Function Points (UFP).
- Find Total Degree of Influence (TDI).
- Compute Value Adjustment Factor (VAF).
- Find the Function Point Count (FPC).

  This means:

1. UFP = (EI*AWF) + (EO*AWF) + (EQ*AWF) +(ILF*AWF) +(EIF*AWF)

   (AWF: Average Weighing Factor)
2. TDI = 14*DI
3. VAF = 0.65+(0.01*TDI)
4. FP = UFP * VAF

Weighting Factor Table (generally average calculation is used)			
Function Unit	LOW	AVG	HIGH

Number of external Inputs (EI)	3	4	6
Number of external outputs (EO)	4	5	7
Number of external inquiries (EQ)	3	4	6
Number of Internal Files (IF)	7	10	15
Number of external and internal interfaces (EIF)	5	7	10

Degree of Influence (DI)

Degree of Influence (DI)	Value
No Influence	0
Incidental	1
Moderate	2
Average	3
Significant	4
Essential	5

Example for finding FP count

Consider a software project with the following functional units.

1) Number of user inputs = 20
2) Number of user outputs = 40
3) Number of inquiries = 5
4) Number of files = 4
5) Number of external interfaces = 2
6) Degree of influences is given as = 3,1,1,3,2,5,4,3,3,3,1,2,5,4
7) Weight Factors are given as = Average for which the values are 4,5,4,10,7

Calculate the function point estimation of the project.

Answer:

- FP = UFP * VAF
- UFP = (EI*AWF) + (EO*AWF) + (EQ*AWF) +(ILF*AWF) +(EIF*AWF)
- VAF = 0.65+(0.01*TDI)
- TDI = 14*DI Calculate UFP

First, let us calculate UFP by referring to the table

UFP Calculation			
Measurement Factor	Count	Weight Factor	Total
Number of external Inputs (EI)	20	4	80
Number of external outputs (EO)	40	5	200
Number of external inquiries (EQ)	4	4	16
Number of Internal Files (IF)	5	10	50
Number of external and internal interfaces (EIF)	2	7	14
Total UFP			**360**

Table 1

Calculation of DI = Average of given complex factor = average of 3,1,1,3,2,5,4,3,3,3,1,2,5,4 as given in the problem

DI = (3+1+1+3+2+5+4+3+3+3+1+2+5+4)/14 = 40/14 = 2.857

Calculate TDI = 14 * DI = 39.998

Calculate VAF

VAF = 0.65 + (0.01 * 39.998) = 1.0499 = 1.05

FP = UFP * VAF

FP = 360 * 1.05 (The value of UFP is taken as the average value from Table 1)

FP = 378

Advantages and drawbacks of FP Metrics

Unlike LOC, FP can be used at the initial software development stage. It does not depend on the programming language. Using FP, the productivity of the software can be determined using just the problem definition. With the FP method, the analysis of the same problem using different languages can be done and the best one can be determined. It can be used where there is a large amount of interface such as GUI.

Some of the drawbacks of the FP metric is it depends on subjective evaluations with a lot of judgment. This model is not suitable for real-time system software development. In many applications, it is necessary to convert FP to LOC for measuring software metrics.

13.4 Project Estimation Techniques

The most fundamental parameters such as the size of the project, efforts, duration, and cost are determined in the project estimation technique. These parameters can be determined using three main categories of estimation techniques:

1. Empirical estimation techniques
2. Heuristic techniques
3. Analytical estimation techniques.

13.4.1 Empirical estimation techniques

In the empirical estimation techniques, an educated guess of project parameters is used. This guess is done using common sense and experience in the field.

Commonly used empirical project estimation techniques are:

a) Expert judgment technique: In this method, experts estimate the cost of different components of the software. The components refer to the modules and subsystems of the software. The drawback of this method is it may include human error or bias. If the expert is not familiar with the project or biased due to political considerations, or the decision may be dominated by other aggressive members, then this method is not effective

b) Delphi estimation techniques: In this method, a coordinator and group of experts. The coordinator provides the SRS document to more than one estimator. Each estimator uses several iterations to come up with the estimate. The coordinator distributes the estimate responses amongst estimators. After many rounds, the coordinator compiles and prepares the final estimate. This technique avoids personal errors or bias due to any considerations.

13.4.2 Heuristic Techniques

In this technique, project parameters are modeled using suitable mathematical expressions. There are two heuristic models for project size estimation.

The first heuristic technique is single variable estimation models. This model is represented as:

The terms in the equation are given as follows:

$$\text{Estimated Parameter} = c_1 * e^{d_1}$$

e = Characteristic of the software estimated known as the independent variable. Estimated Parameters include staff size, efforts, project duration, etc.

c1 and d1 are constants. The values of c1 and c2 are determined using the values from the previous projects. This is done using a constructive cost estimation (COCOMO) model.

Another project estimation model is the multivariable cost estimation model. It takes the following form. The multivariable model is more accurate than a single variable as includes more amount of historical data. The mathematical equation in this model is given as:

$$\text{Estimated Resource} = c_1 * e_1^{d_1} + c_2 * e_2^{d_2} + \ldots$$

e1, e2, …: Independent characteristics of the software already submitted.

c1, c2, d1, d2, …: constants using historical data.

13.5 COCOMO Model

COCOMO model is proposed by Barry Boehm in 1981 after considering a study of 63 projects. The different parameters such as size, effort, cost, time, and quality are estimated in this model. According to Boehm, the software projects can be divided as follows.

1) **Organic**: In this type of project is well understood with a small development team having good experience in a similar project. These projects are simple, and the project size is in the range (of 2KLOC –50 KLOC).
2) **Semidetached**: In this type of project, the development team is a mixture of experienced and inexperienced team members. These projects are used for creating utility programs in the range of (50KLOC –300 KLOC)
3) **Embedded:** In this type of project, the software to be developed is strongly connected to the complex hardware. A development team needs to know embedded system-based software of this type. These are complex projects with restricted regulations and a size >300 KLOC.

COCOMO model is divided into three types.

1) Basic COCOMO Model
2) Intermediate COCOMO model
3) Complete COCOMO model

1) Basic COCOMO Model

The basic COCOMO model gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

$$\textbf{Effort} = a_1 \times (\text{KLOC})^{a_2}\ \text{PM}$$

$$\textbf{Tdev} = b_1 \times (\text{Effort})^{b_2}\ \text{Months}$$

- **KLOC**: Kilo Lines of Code as an estimated size of the project
- **a1, a2, b1, b2**: Constant values for each category of the software products
- **Tdev:** estimated time to develop the software, expressed in months
- **PMs**: Total effort required to develop the software product, expressed in person-months.

The constant values are given in the following table.

Project	a1	a2	b1	b2
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

1) Estimation of development effort:

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

Organic : $\textbf{Effort} = 2.4(KLOC)^{1.05}$ **PM**
Semi-detached : $\textbf{Effort} = 3.0(KLOC)^{1.12}$ **PM**
Embedded : $\textbf{Effort} = 3.6(KLOC)^{1.20}$ **PM**

2) Estimation of development time:

Organic : $\textbf{Tdev} = 2.5(Effort)^{0.38}$ **Months**
Semi-detached : $\textbf{Tdev} = 2.5(Effort)^{0.35}$ **Months**
Embedded : $\textbf{Tdev} = 2.5(Effort)^{0.32}$ **Months**

Example: Assume that the size of an organic type of software product has been estimated to be 32,000 lines of source code. Assume that the average salary of software engineers is Rs. 15,000/- per month. Determine the effort required to develop the software product and the nominal development time.

Solution:

From the basic COCOMO estimation formula for organic software:

Effort = **2.4 x (32)1.05 = 91 PM**

Nominal development time = **2.5 x (91)0.38 = 14 months**

Cost required to develop the product = **14 x 15,000**

= **Rs. 210,000/-**

2) **Intermediate COCOMO model**

The basic COCOMO model determines efforts using LOC which is only a function of the size/ number of lines of code and some. But in practice, only LOC cannot be used to determine any system's efforts and schedule. The other factors such as experience, capability, and reliability must be considered should be used in estimating efforts. These factors are known as **multipliers or cost drivers.** There are 15 such drivers used in cost estimation.

The cost drivers are classified based on the attributes of the following items.

1. **Product:** The complexity and reliability of the product are considered in this attribute.
2. **Computer:** The configuration of the computer such as execution speed, memory storage, etc. is considered in this attribute.
3. **Personnel:** The experience of the software developer, knowledge, and analytical ability is considered in this attribute**.**
4. **Development Environment:** The facilities available for software development are considered in this attribute. One of the important features that are required in this attribute is CASE tools used for the readymade availability of the software components.

3) **Complete COCOMO model**

In the basic and intermediate COCOMO model, the software product is considered a homogeneous entity. Most of the large systems consist of many smaller sub-systems. These subsystems may be organic type, some may be semidetached or embedded. The complexity and reliability of these subsystems may be different. The complete COCOMO model considers these characteristics and estimates the efforts and development time as the sum of the estimates for the individual subsystems. The cost of each subsystem is done separately. Each subsystem may have widely different characteristics. For example, some subsystems may be considered as organic type, some semidetached, and some are embedded. The complexity and reliability requirements of each subsystem may be different. The complete COCOMO model considers these differences in characteristics of the subsystems and estimates the effort and development time as the sum of the estimates for the individual subsystems. The cost of each subsystem is estimated separately. The margin of errors in the final estimate is less in this model. The cost driver's impact on each step of the software engineering process is considered.

The COCOMO is applied in each phase to estimate effort and then the efforts are added. 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

Example of Complete COCOMO Model

Consider a distributed Management Information System (MIS) product for an organization. The organization is spread across the country with offices in several places. Each office can have sub-components such as a database, graphical user interface (GUI), and communication part. Here the communication is of the type of embedded software. The database part is semi-detached software and GUI is of the type organic software type. The overall cost of the system is estimated as the sum of the three subcomponents.

13.6 Analytical Estimation Techniques

In this technique, the given task is divided into basic components If there is time calculation available from some other source, then the same is applied to all the components. If such time is not available, then the estimation of work is done using the experience of the work. This is done using Halstead's software science.

Halstead's Software Science – An Analytical Technique

Halstead's software science is an analytical technique used to measure the size of the software, development effort, and software products. Halstead's method uses a few program parameters. For a given program, let:

- $\eta 1$ be the number of unique operators used in the program,
- $\eta 2$ be the number of unique operands used in the program,
- N1 is the total number of operators used in the program,
- N2 is the total number of operands used in the program.

The different parameters developed using Halstead are as follows:

1. Length and Vocabulary

   The length of a program is considered the sum of all operators and operands in the program.

   Length = N = N1 + N2

   The unique operators and operands used in the program are given as program vocabulary and it is given as follows:

   $\eta = \eta_1 + \eta_2$

2. Program Volume

   When the programming style is different, then using only operators and operands is not enough. The project size can be estimated using program volume. Program volume is measured as the minimum number of bits needed to encode a program. It is given as:

   $V = N\log2\eta$

   Here η refers to the different identifiers and N is the length of the program.

3. Potential Minimum Volume

   The volume can be defined as potential minimum volume V*. It refers to that part of the program which is clear and relates to the actual problem and is coded.

   V* is given as:

   $V^* = (2 + \eta 2) \log2(2 + \eta 2)$

4. Program level

The program level is used to measure the level of abstraction provided by the programming language. There can ve different levels in the program and it is given as:

$$L = V^*/V$$

5. Effort

The effort unit is used to read and understand the program. It refers to the number of mental efforts required to convert an algorithm to a computer program. The effort is given as:

Effort $E = V/L$,
$E = V^2/V^*$ (since $L = V^*/V$)

6. Time

The time required by the programmer is given as:

$$T = E/S$$

where S is the speed of a person’s brain’s processing rate. It is determined by phycological reasoning.

7. Length Estimation

Halstead’s method recommends the use of unique operators and operands to determine the length of the program. The length is given as:

$$N = \eta 1\log 2\eta 1 + \eta 2\log 2\eta 2$$

In summary, Halstead’s theory includes three important quality parameters in the software metric calculations. These parameters are the complexity of the program, ease of understanding, and abstraction levels.

Example: Software Metric using Halstead’s Calculations

- Let us consider the following C program:

```
main ()
 {
   int x, y, z, avg;
   scanf(“%d %d %d”, &x, &y, &z);
   avg = (x+y+z)/3
  printf(“avg = %d”, avg);
```

}

- The unique operators(η1) are:

  main, (),{},int,scanf,&,“,”,“;”,=,+,/, printf

- The unique operands(η2) are:

  x, y, z, &x, &y, &z, x+y+z, avg, 3, “%d %d %d”, “avg = %d”

- Therefore,

  η1 = 12, η2 = 11

- Estimated Length = (12*log12 + 11*log11), (since N = η1log2η1 + η2log2η2)

  = (12*3.58 + 11*3.45)

  = (43+38) = 81

- Volume = Length*log (23) , (since *Volume* V = Nlog2η)

  = 81*4.52 = 366

13.7 Multiple Choice Questions (MCQs)

1. LOC and FP calculation is used for:
   a) Manpower
   b) Duration
   c) Cost
   d) Software Size
2. The correctness is measured using:
   a) Cost per LOC
   b) Documents for LOC
   c) Defects per LOC
   d) Time for LOC
3. The software size metric is used for
   a) Number of modules
   b) Number of functions
   c) Number of inputs
   d) Number of lines of code
4. Project size and complexity are part of the
   a) Process Metrics
   b) Product Metrics

c) Software Reliability

d) None of the above

5. Software project management goal does not refer to:

a) Software delivery in allotted time

b) Keeping good environment in the team

c) Avoiding customer clients

d) Managing costs within the budget

13.8 Review Questions

1) Discuss the activities in project planning
2) List the different fields in software project management document
3) Describe the key features of a COCOMO for cost estimation.
4) Discuss in detail about software evolution.
5) What is function point? Explain its importance.
6) Explain the different parameters in function point analysis.
7) Explain two metrics which are used to measure the software in detail. Clearly discuss the advantages and disadvantages of these metrics
8) Explain the different software metrics used in Halstead's Software Science.
9) Suppose that a project was estimated to be 400 KLOC. Given the constant values as follows.

Project	a1	a2	b1	b2
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

Calculate effort & time for each of 3 modes of development for all the three types of projects using COCOMO.

10) Discuss the different types of projects classified in COCOMO.

13.9 Keys in Multiple Choice Questions (MCQs)

1: d	2: c	3: d	4: b	5: c