UNIT 1 SOFTWARE PROJECT PLANNING

Structure **Page Nos.**

1.0 INTRODUCTION

Historically, software projects have dubious distinction of overshooting project schedule and cost. Estimating duration and cost continues to be a weak link in software project management. The aim of this unit is to give an overview of different project planning techniques and tools used by modern day software project managers.

It is the responsibility of the project manager to make as far as possible accurate estimations of effort and cost. This is particularly what is desired by the management of an organisation in a competitive world. This is specially true of projects subject to competition in the market where bidding too high compared with competitors would result in losing the business and a bidding too low could result in financial loss to the organisation. This makes software project estimation crucial for project managers.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- understand different software estimation techniques;
- understand types of metrics used for software project estimation;
- learn about models used for software estimation, and
- learn about different automated tools used for estimation.

1.2 DIFFERENT TYPES OF PROJECT METRICS

It is said that if you cannot measure, then, it is not engineering. Effective management of software development process requires effective measurement of software development process. Often, from the input given by project leaders on the estimation of a software project, the management decides whether to proceed with the project or not.

The process of software estimation can be defined as the set of techniques and procedures that an organisation uses to arrive at an estimate. An important aspect of software projects is to know the cost, time, effort, size etc.

Need for Project metrics : Historically, the process of software development has been witnessing inaccurate estimations of schedule and cost, overshooting delivery target and productivity of software engineers in not commensurate with the growth of demand. Software development projects are quite complex and there was no scientific method of measuring the software process. Thus effective measurement of the process was virtually absent. The following phrase is aptly describing the need for measurement:

If you can not measure it, then, you can not improve it.

This is why measurement is very important to software projects. Without the process of measurement, software engineering cannot be called engineering in the true sense.

Definition of metrics : Metrics deal with measurement of the software process and the software product. Metrics quantify the characteristics of a process or a product. Merics are often used to estimate project cost and project schedule.

Examples of metrics : Lines of code(LOC), pages of documentation, number of man-months, number of test cases, number of input forms.

Types of Project metrics

Metrics can be broadly divided into two categories namely, product metrics and process metrics.

Product metrics provides a measure of a software product during the process of development which can be in terms of lines of code (either source code or object code), pages of documentation etc.

Process metrics is a measure of the software development process such as time, effort etc.

Another way of classification of metrics are primitive metrics and derived metrics.

Primitive metrics are directly observable quantities like lines of code (LOC), number of man-hours etc.

Derived metrics are derived from one or more of primitive metrics like lines of code per man-hour, errors per thousand lines of code.

Now, let us briefly discuss different types of product metrics and process metrics.

Product metrics

Lines of Code(LOC) : LOC metric is possibly the most extensively used for measurement of size of a program. The reason is that LOC can be precisely defined. LOC may include executable source code and non-executable program code like comments etc.

Looking at the following table, we can know the size of a module.

Module	**Effort in man-months**	**LOC**
Module 1	3	24,000
Module 2	4	25,000

Looking at the data above we have a direct measure of the size of the module in terms of LOC. We can derive a productivity metrics from the above primitive metrics i.e., LOC.

Productivity of a person = LOC / man-month

Quality = No. of defects / LOC

It is evident that the productivity of the developer engaged in Module 1 is more than the productivity of the developer engaged in Module 2. It is important to note here how derived metrics are very handy to project managers to measure various aspects of the projects.

Although, LOC provides a direct measure of program size, at the same time, these metrics are not universally accepted by project managers. Looking at the data in the table below, it can be easily observed that LOC is not an absolute measure of program size and largely depends on the computer language and tools used for development activity.

Consider the following table:

Module	**Effort in man-months**	**LOC in COBOL**	**LOC in Assembly**	**LOC in 4 GL**
Module- 1	3	24,000	800,000	400
Module- 2	4	25,000	100,000	500

The LOC of same module varies with the programming language used. Hence, just LOC cannot be an indicator of program size. The data given in the above table is only assumed and does not correspond to any module(s).

There are other attributes of software which are not directly reflected in Lines of Code (LOC), as the complexity of the program is not taken into account in LOC and it penalises well designed shorter program. Another disadvantage of LOC is that the project manager is supposed to estimate LOC before the analysis and design is complete.

Function point : Function point metrics instead of LOC measures the functionality of the program. Function point analysis was first developed by Allan J. Albrecht in the 1970s. It was one of the initiatives taken to overcome the problems associated with LOC.

In a Function point analysis, the following features are considered:

- **External inputs** : A process by which data crosses the boundary of the system. Data may be used to update one or more logical files. It may be noted that data here means either business or control information.

- **External outputs** : A process by which data crosses the boundary of the system to outside of the system. It can be a user report or a system log report.

- **External user inquires** : A count of the process in which both input and output results in data retrieval from the system. These are basically system inquiry processes.

- **Internal logical files** : A group of logically related data files that resides entirely within the boundary of the application software and is maintained through external input as described above.

- **External interface files** : A group of logically related data files that are used by the system for reference purposes only. These data files remain completely outside the application boundary and are maintained by external applications.

As we see, function points, unlike LOC, measure a system from a functional perspective independent of technology, computer language and development method. The number of function points of a system will remain the same irrespective of the language and technology used.

For transactions like external input, external output and user inquiry, the ranking of high, low and medium will be based on number of file updated for external inputs or number of files referenced for external input and external inquiries. The complexity will also depend on the number of data elements.

Consider the following classification for external inquiry:

No. of Data elements	**Number of file references**		
	0 to 2	**3 to 4**	**5 and above**
1 to 5	Low	Low	Low
6 to 10	Low	Medium	Medium
10 to 20	Medium	Medium	High
More than 20	Medium	High	High

Also, External Inquiry, External Input and External output based on complexity can be assigned numerical values like rating.

Component type	**Values**		
	Low	**Medium**	**High**
External Output	4	6	8
External Input	2	4	6
External Inquiry	3	5	7

Similarly, external logical files and external interface files are assigned numerical values depending on element type and number of data elements.

Component type	**Values**		
	Low	**Medium**	**High**
External Logical files	6	8	10
External Interface	5	7	9

Organisations may develop their own strategy to assign values to various function points. Once the number of function points have been identified and their significance has been arrived at, the total function point can be calculated as follows.

Type of Component	**Complexity of component**			
	Low	**Medium**	**High**	**Total**
Number of External Output	X 4 =	X 6 =	X 8 =	
Number of External Input	X 2 =	X 4 =	X 6 =	
Number of External Inquiry	X3 =	X 5 =	X 7 =	
Number of Logical files	X 6 =	X 8 =	X 10 =	
Number of Interface file	X 5 =	X 7 =	X 9 =	
			Total of function point	

Total of function points is calculated based on the above table. Once, total of function points is calculated, other derived metrics can be calculated as follows:

Productivity of Person = Total of Function point / man-month
Quality = No. of defects / Total of Function points.

Benefits of Using Function Points

- Function points can be used to estimate the size of a software application correctly irrespective of technology, language and development methodology.
- User understands the basis on which the size of software is calculated as these are derived directly from user required functionalities.
- Function points can be used to track and monitor projects.
- Function points can be calculated at various stages of software development process and can be compared.

Other types of metrics used for various purposes are quality metrics which include the following:

- **Defect metrics** : It measures the number of defects in a software product. This may include the number of design changes required, number of errors detected in the test, etc.

- **Reliability metrics** : These metrics measure mean time to failure. This can be done by collecting data over a period of time.

☞ **Check Your Progress 1**

1) Lines of Code (LOC) is a product metric. True ☐ False ☐

2) _____ are examples of process metrics.

1.3 SOFTWARE PROJECT ESTIMATION

Software project estimation is the process of estimating various resources required for the completion of a project. Effective software project estimation is an important activity in any software development project. Underestimating software project and under staffing it often leads to low quality deliverables, and the project misses the target deadline leading to customer dissatisfaction and loss of credibility to the company. On the other hand, overstaffing a project without proper control will increase the cost of the project and reduce the competitiveness of the company.

Software project estimation mainly encompasses the following steps:

- Estimating the size of project. There are many procedures available for estimating the size of a project which are based on quantitative approaches like estimating Lines of Code or estimating the functionality requirements of the project called Function point.

- Estimating efforts based on man-month or man-hour. Man-month is an estimate of personal resources required for the project.

- Estimating schedule in calendar days/month/year based on total man-month required and manpower allocated to the project

  Duration in calendar month = Total man-months / Total manpower allocated.

- Estimating total cost of the project depending on the above and other resources.

In a commercial and competitive environment, Software project estimation is crucial for managerial decision making. The following *Table* give the relationship between various management functions and software metrics/indicators. Project estimation and tracking help to plan and predict future projects and provide baseline support for project management and supports decision making.

Consider the following table:

Activity	Tasks involved
Planning	Cost estimation, planning for training of manpower, project scheduling and budgeting the project
Controlling	Size metrics and schedule metrics help the manager to keep control of the project during execution
Monitoring/improving	Metrics are used to monitor progress of the project and wherever possible sufficient resources are allocated to improve.

Figure 1.1 depicts the software project estimation.

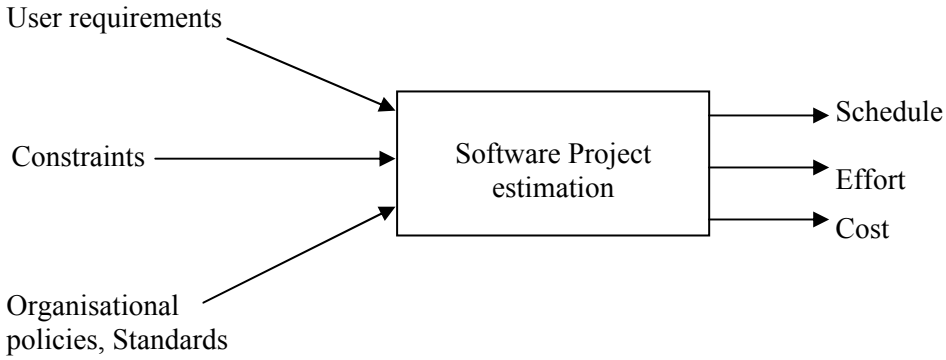

Figure 1.1: Software project estimation

1.3.1 Estimating the size

Estimating the size of the software to be developed is the very first step to make an effective estimation of the project. Customer's requirements and system specification forms a baseline for estimating the size of a software. At a later stage of the project, system design document can provide additional details for estimating the overall size of a software.

- The ways to estimate project size can be through past data from an earlier developed system. This is called estimation by analogy.
- The other way of estimation is through product feature/functionality. The system is divided into several subsystems depending on functionality, and size of each subsystem is calculated.

1.3.2 Estimating effort

Once the size of software is estimated, the next step is to estimate the effort based on the size. The estimation of effort can be made from the organisational specifics of software development life cycle. The development of any application software system is more than just coding of the system. Depending on deliverable requirements, the estimation of effort for project will vary. Efforts are estimated in number of man-months.

- The best way to estimate effort is based on the organisation's own historical data of development process. Organizations follow similar development life cycle for developing various applications.
- If the project is of a different nature which requires the organisation to adopt a different strategy for development then different models based on algorithmic approach can be devised to estimate effort.

1.3.3 Estimating Schedule

The next step in estimation process is estimating the project schedule from the effort estimated. The schedule for a project will generally depend on human resources involved in a process. Efforts in man-months are translated to calendar months.

Schedule estimation in calendar-month can be calculated using the following model [McConnell]:

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

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

1.3.4 Estimating Cost

Cost estimation is the next step for projects. The cost of a project is derived not only from the estimates of effort and size but from other parameters such as hardware, travel expenses, telecommunication costs, training cost etc. should also be taken into account.

Figure 1.2 depicts the cost estimation process.

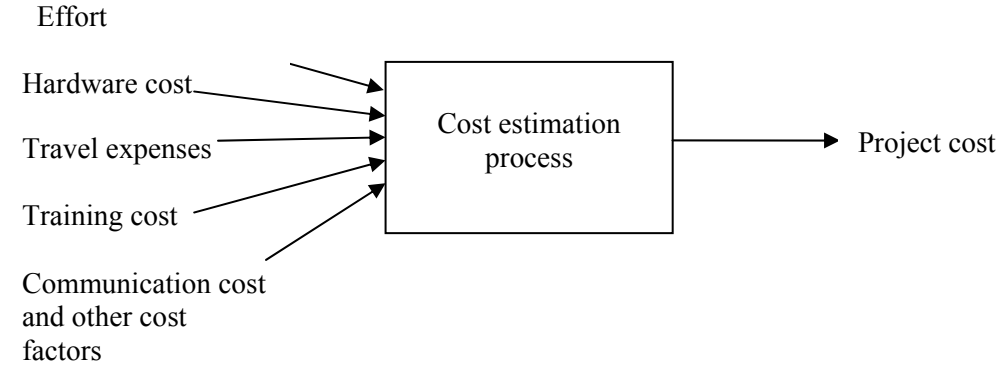

Figure 1.2 : Cost estimation process

Figure 1.3 depicts project estimation process.

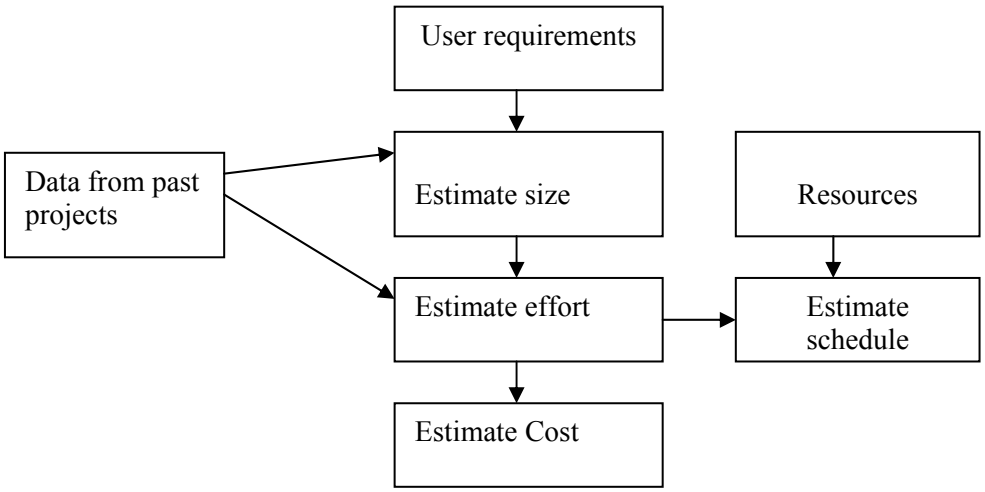

Figure 1.3: Project estimation process

Now, once the estimation is complete, we may be interested to know how accurate the estimates are to reality. The answer to this is "we do not know until the project is complete". There is always some uncertainty associated with all estimation techniques. The accuracy of project estimation will depend on the following:

- Accuracy of historical data used to project the estimation
- Accuracy of input data to various estimates
- Maturity of organisation's software development process.

The following are some of the reasons which make the task of cost estimation difficult:

- Software cost estimation requires a significant amount of effort. Sufficient time is not allocated for planning.
- Software cost estimation is often done hurriedly, without an appreciation for the actual effort required and is far from real issues.
- Lack of experience for developing estimates, especially for large projects.
- An estimator uses the extrapolation technique to estimate, ignoring the non-linear aspects of software development process

The following are some of the reasons for poor and inaccurate estimation:

- Requirements are imprecise. Also, requirements change frequently.
- The project is new and is different from past projects handled.
- Non-availability of enough information about past projects.
- Estimates are forced to be based on available resources.

Cost and time tradeoffs

If we elongate the project, we can reduce overall cost. Usually, long project durations are not liked by customers and managements. There is always shortest possible duration for a project, but it comes at a cost.

The following are some of the problems with estimates:

- Estimating size is often skipped and a schedule is estimated which is of more relevance to the management.
- Estimating size is perhaps the most difficult step which has a bearing on all other estimates.
- Let us not forget that even good estimates are only projections and subject to various risks.
- Organisations often give less importance to collection and analysis of historical data of past development projects. Historical data is the best input to estimate a new project.
- Project managers often underestimate the schedule because management and customers often hesitate to accept a prudent realistic schedule.

Project estimation guidelines

- Preserve and document data pertaining to organisation's past projects.
- Allow sufficient time for project estimation especially for bigger projects.
- Prepare realistic developer-based estimate. Associate people who will work on the project to reach at a realistic and more accurate estimate.
- Use software estimation tools.
- Re-estimate the project during the life cycle of development process.
- Analyse past mistakes in the estimation of projects.

☞ **Check Your Progress 2**

1) What is the first step in software project estimation?

……

2) What are the major inputs for software project estimation?

……

1.4 MODELS FOR ESTIMATION

Estimation based on models allows us to estimate projects ignoring less significant parameters and concentrating on crucial parameters that drive the project estimate. Models are analytic and empirical in nature. The estimation models are based on the following relationship:

$$E = f(v_i)$$

E = different project estimates like effort, cost, schedule etc.

v_i = directly observable parameter like LOC, function points

1.4.1 COCOMO Model

COCOMO stands for Constructive Cost Model. It was introduced by Barry Boehm. It is perhaps the best known and most thoroughly documented of all software cost estimation models. It provides the following three level of models:

- **Basic COCOMO** : A single-value model that computes software development cost as a function of estimate of LOC.
- **Intermediate COCOMO** : This model computes development cost and effort as a function of program size (LOC) and a set of cost drivers.
- **Detailed COCOMO** : This model computes development effort and cost which incorporates all characteristics of intermediate level with assessment of cost implication on each step of development (analysis, design, testing etc.).

This model may be applied to three classes of software projects as given below:

- Organic : Small size project. A simple software project where the development team has good experience of the application
- Semi-detached : An intermediate size project and project is based on rigid and semi-rigid requirements.
- Embedded : The project developed under hardware, software and operational constraints. Examples are embedded software, flight control software.

In the COCOMO model, the development effort equation assumes the following form:

$$E = aS^b m$$

where **a** and **b** are constraints that are determined for each model.

E = Effort
S = Value of source in LOC
m = multiplier that is determined from a set of 15 cost driver's attributes.

The following are few examples of the above cost drivers:

- Size of the application database
- Complexity of the project
- Reliability requirements for the software
- Performance constraints in run-time
- Capability of software engineer
- Schedule constraints.

Barry Boehm suggested that a detailed model would provide a cost estimate to the accuracy of ± 20 % of actual value

1.4.2 Putnam's model

L. H. Putnam developed a dynamic multivariate model of the software development process based on the assumption that distribution of effort over the life of software development is described by the Rayleigh-Norden curve.

$$P = Kt \exp(t^2/2T^2) / T^2$$

P = No. of persons on the project at time 't'
K = The area under Rayleigh curve which is equal to total life cycle effort
T = Development time

The Rayleigh-Norden curve is used to derive an equation that relates lines of code delivered to other parameters like development time and effort at any time during the project.

$$S = C_k K^{1/3} T^{4/3}$$

S = Number of delivered lines of source code (LOC)
C_k = State-of-technology constraints
K = The life cycle effort in man-years
T = Development time.

1.4.3 Statistical Model

From the data of a number of completed software projects, C.E. Walston and C.P. Felix developed a simple empirical model of software development effort with respect to number of lines of code. In this model, LOC is assumed to be directly related to development effort as given below:

$$E = a\ L^b$$

Where L = Number of Lines of Code (LOC)

E = total effort required

a and **b** are parameters obtained from regression analysis of data. The final equation is of the following form:

$$E = 5.2\ L^{0.91}$$

The productivity of programming effort can be calculated as

$$P = L/E$$

Where P = Productivity Index

1.4.4 Function Points

It may be noted that COCOMO, Putnam and statistical models are based on LOC. A number of estimation models are based on function points instead of LOC. However, there is very little published information on estimation models based on function points.

1.5 AUTOMATED TOOLS FOR ESTIMATION

After looking at the above models for software project estimation, we have reason to think of software that implements these models. This is what exactly the automated estimation tools do. These estimation tools, which estimate cost and effort, allow the project managers to perform "What if analysis". Estimation tools may only support size estimation or conversion of size to effort and schedule to cost.

There are more than dozens of estimation tools available. But, all of them have the following common characteristics:

- Quantitative estimates of project size (e.g., LOC).
- Estimates such as project schedule, cost.
- Most of them use different models for estimation.

Figure 1.4 depicts a typical structure of estimation tools.

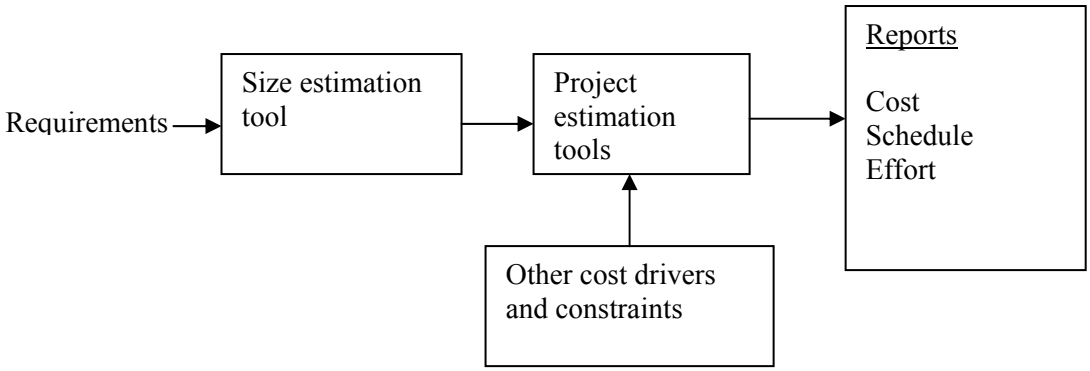

Figure 1.4: Typical structure of estimation tools

No estimation tool is the solution to all estimation problems. One must understand that the tools are just to help the estimation process.

Problems with Models

Most models require an estimate of software product size. However, software size is difficult to predict early in the development lifecycle. Many models use LOC for sizing, which is not measurable during requirements analysis or project planning. Although, function points and object points can be used earlier in the lifecycle, these measures are extremely subjective.

Size estimates can also be very inaccurate. Methods of estimation and data collection must be consistent to ensure an accurate prediction of product size. Unless the size metrics used in the model are the same as those used in practice, the model will not yield accurate results (Fenton, 1997).

The following *Table* gives some estimation tools:

Automated Estimation Tools

Tool	Tool vendor site	Functionality/Remark
EstimatorPal	http://www.effortestimator.com/	EstimatorPal© is a software tool that assists software developers estimate the effort required to be spent on various activities. This tool facilitates use of the following Estimation techniques – Effort estimation tools which supports Function Point Analysis Technique., Objects Points Technique, Use Case Points Technique and Task-Based Estimation Technique
Estimate Easy Use Case	Duvessa Software http://www.duvessa.com/	Effort estimation tool based on use cases
USC COCOMO II	USC Center for Software Engineering http://sunset.usc.edu/research/COCOMOII/index.html	Based on COCOMO
ESTIMACS	Computer Associates International Inc. http://www.cai.com/products/estimacs.htm	Provides estimates of the effort, duration, cost and personnel requirements for maintenance and new application development projects.
Checkpoint	Software Productivity Research Inc. http://www.spr.com/	Guides the user through the development of a software project estimate and plan.
Function Point Workbench	Software Productivity Research Inc. http://www.spr.com/	Automated software estimation tools, implementing function point sizing techniques, linking project estimation to project management initiatives, and collecting historical project data to improve future estimation efforts
ESTIMATE Professional	Software Productivity Center http://www.spc.com	Based on Putnam, COCOMO II
SEER-SEM	Galorath http://www.galorath.com/	Predicts, measures and analyzes resources, staffing, schedules, risk and cost for software projects
ePM.Ensemble	InventX http://www.inventx.com/	Support effort estimation, among other things.
CostXpert	Marotz, Inc. http://www.costxpert.com/	Based on COCOMO

1.6 SUMMARY

Estimation is an integral part of the software development process and should not be taken lightly. A well planned and well estimated project is likely to be completed in time. Incomplete and inaccurate documentation may pose serious hurdles to the success of a software project during development and implementation. Software cost estimation is an important part of the software development process. Metrics are important tools to measure software product and process. Metrics are to be selected carefully so that they provide a measure for the intended process/product. Models are used to represent the relationship between effort and a primary cost factor such as software product size. Cost drivers are used to adjust the preliminary estimate provided by the primary cost factor. Models have been developed to predict software cost based on empirical data available, but many suffer from some common problems. The structure of most models is based on empirical results rather than theory. Models are often complex and rely heavily on size estimation. Despite problems, models are still important to the software development process. A model can be used most effectively to supplement and corroborate other methods of estimation.

1.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1. True
2. Time, Schedule

Check Your Progress 2

1. Estimating size of the Project
2. User requirements, and project constraints.

1.8 FURTHER READINGS

1) *Software Engineering,* Ian Sommerville; *Sixth Edition, 2001*, Pearson Education.

2) *Software Engineering – A Practitioner's Approach*, Roger S. Pressman; McGraw-Hill International Edition.

Reference websites

http://www.rspa.com
http://www.ieee.org
http://www.ncst.ernet.in