Software Project Planning

Unit 1

Red: indicates important

Contents

- Project Matrices
- Software Project Estimation
- Models for estimation
- Automated tools for estimation

Software Estimation and project metrics

- Software estimation is set of techniques and procedures that an organization uses to arrive at an estimate.
- Metrics : It deals with measurement of the software process and the software product. Metrics quantify the characteristics of a process or a product. Metrics are used to estimate project cost and project schedule.
- Examples of metrics: lines of code(LOC), pages of documentation, number of months, number of text cases, number of input forms.
- Types of project metrics: Metrics can be broadly divided into two categories: 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, pages of documentation etc..
- Process metrics is a measure of the software development process such as time, effort etc..

- Another classification of metrics : 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 primitive metrics like lines of code per man-hour.
- Types of product metrics:

1. 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..

   Productivity of a person= LOC/man-month

   Quality= No. of defects/LOC

- LOC provides a direct measure of program size. These metrics are not universally accepted by project managers. LOC depends on the computer language and tools used for development.
- Disadvantage of LOC is that the project manager is supposed to estimate LOC before the analysis and design is complete.

2. Function point

- It was an initiative to overcome the problems associated with LOC.
- It measures the functionality of the program.
- Was developed by Allan J. Albrecht in the 1970s.
- Features considered in FP / action points:
  - External inputs: A process by which data crosses the boundary of the system. Data may be used to update one or more logical files. Data may be 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 purpose only. These data files remain completely outside the application boundary and are maintained by external applications.

- Its independent of technology, computer language and development method.
- The above 5 action types are ranked according to their complexity: low, average and high. For transactions like external input, external output and user inquiry ranking is based on number of file updated for external inputs or number of files referenced for external input and external inquiries.
- External inquiry, external input and external output can be assigned numerical values like rating based on complexity.

- External logical files and external interface files are assigned numeric values depending on element type and number of data elements.
- Organizations develop criteria for determining whether a particular entry is Low, Average or High. Once the number of function points have been identified and their significance has been arrived at, the total function point can be calculated. Once the total of function point is determined, other derived metrics are calculated:
- Productivity of person=total of function point/man-month

  Quality=No. of defects/ total of function points
- Benefits of using function point:
  - It 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.

- It can be used to track and monitor projects.
- It can be calculated at various stages of software development process and can be compared.

- Another type of metrics is quality matrics. This type of matrics include:
  - 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.

Software Project Estimation

- It's the process of estimating various resources required for the completion of a project.
- Its an important activity in any software development project.
- Steps in software project estimation:
  - Estimating the size of project. There are many ways 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 he project depending on the above and other resources.
  - Software project estimation(Fig 1.1,page:10, block 2) and (Fig 1.3 page 11, block 2)

- Software project estimation :

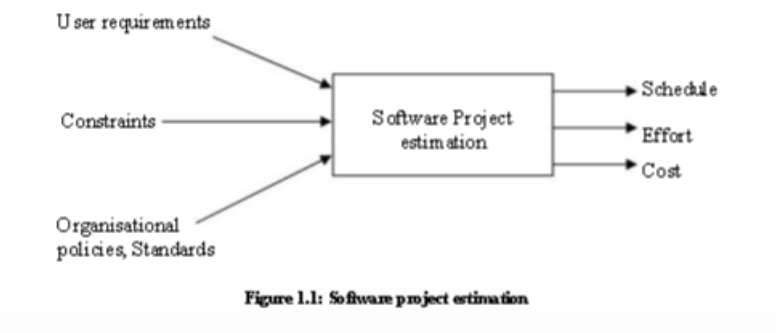

Figure 1.1: Software project estimation

Project estimation process :

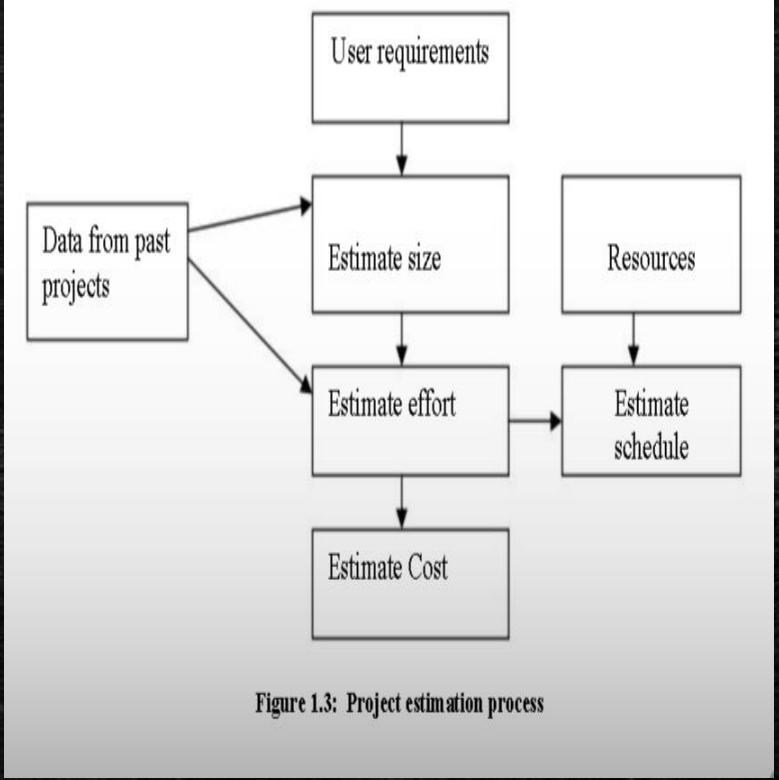

Figure 1.3: Project estimation process

Estimating the size

- It's the first step to make an effective estimation of the project.
- Customer's requirements and system specification becomes a baseline for this step.
- At later stage system design document provides more details for estimating the overall size.
- 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 the size of each subsystem is calculated.

Estimating effort

- The nest step is to estimate the effort based on the size.
- Its made from the organizational specifics of software development life cycle.
- It depends on deliverable requirements.
- Effort is measured in number of man-months.
- The best way to estimate effort is based on the organization's own historical data of development process.
- If the project is of different nature which requires the organization to adopt a different strategy for development then different models based o algorithmic approach can be devised to estimate effort.

Estimating Schedule

- Next step is estimating the project schedule from the effort estimated.
- It depends on human resources involved in a process. Efforts in man-months are translated to calendar months.
- This can be done using the McConnell model:

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

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

Estimating cost

- It's the next step.
- It is derived not only from the effort and size but from other parameters such as hardware, travel expense, training cost etc..
- Cost estimation process (Fig 1.2,Page 11, block 2)
- Accuracy of project estimation depends on:
  - Accuracy of historical data used to project the estimation.
  - Accuracy of input data to various estimates.
  - Maturity of organization's software development process.
- Reasons why cost estimation is difficult:
  - It requires a significant amount of effort. Sufficient time is not allocated for planning.
  - Its done hurriedly, without an appreciation for the actual effort required.
  - 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.

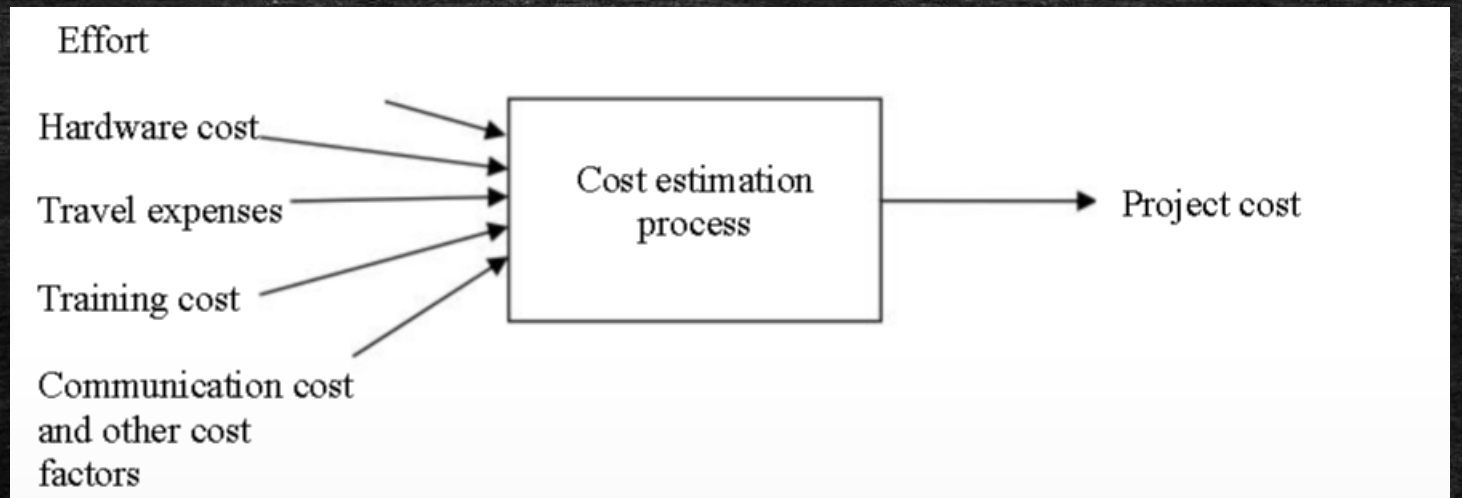

Figure 1.2 : Cost estimation process

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.

Problems with estimates

- Estimating size is often skipped and a schedule is estimated which is of more relevance to the management.
- Estimating size is the most difficult step which has a bearing on all other estimates.
- Good estimates are only projections and subject to various risks.
- Organizations give less importance to collection of historical data. It's 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 organization's past projects.
- Allow sufficient time for project estimation especially for bigger projects.
- Prepare realistic developer-based estimate.
- Use software estimation tools.
- Re-estimate the project during the life cycle of development process.
- Analyze past mistakes in the estimation of projects.

Models for estimation

- 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

V_i: directly observable parameter like LOC, function points

COCOMO Model

- Stands for Constructive Cost Model
- Introduced by Barry Boehm
- It's the best known and most thoroughly documented of all software cost estimation model.
- 3 levels of model:
  1. Basic COCOMO: A single-valued model that computes software development cost as a function of estimate of LOC.
  2. Intermediate COCOMO: This module computes development cost and effort as a function of program size(LOC) and a set of cost drivers.
  3. 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 can 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.
- Development effort equation:

$E=aS^bm$

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.

- Examples of 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
- Cost drivers are used to adjust the preliminary estimate provided by the primary cost factor.
- Barry Boehm suggested that a detailed model will provide a cost estimate to the accuracy of ±20% of actual value.

Putnam's model

- The Putnam estimation model is a dynamic multivariate model for effort estimation.
- Was developed by L.H.Putnam.
- The distribution of effort over the life of software is modeled by Rayleigh-Norden curve.

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

P: Number 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 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-hours

T: Development time

Statistical Model

- It's a simple empirical model of software development effort with respect to number of lines of code.
- Developed by C.E.Walston and C.P.Felix
- LOC is directly related to development effort :

$E=aL^b$

Where L =number of lines of code

E:total effort required

a and b are parameters obtained fro, regression analysis of data.

Final equation:

$E=5.2\ L^{0.91}$

- Productivity of programming effort:

P=L/E

P: productivity index

Automated tools for estimation

- Characteristics of estimation tools:
  - Quantitative estimates of project size (eg. LOC)
  - Estimates such as project schedule, cost etc.
  - Most of them use different model for estimation.
- Typical structure of estimation tools

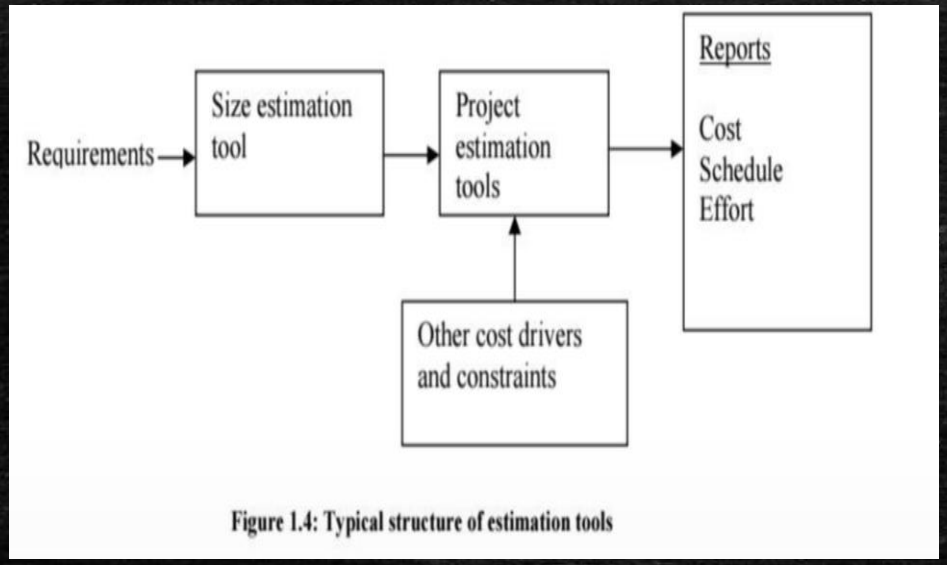

Figure 1.4: Typical structure of estimation tools

- Problems with models:
  - Most models require an estimate of software product size. Software size is difficult to predict early in the development lifecycle.
  - Size estimates can also be very inaccurate.

Automated Estimation Tools

- EstimatorPal:
  - It's a software tool that assists software developers estimate the effort required to be spent on various activities.
  - It facilitates the use of following estimation techniques: Effort estimation tools (supports Function Point Analysis technique, Objects Point technique, Use Case Points technique and Task Based Estimation technique)
- Estimate Easy Use Case:
  - Its an effort estimation tool based on use cases.
- USC COCOMO II:
  - Its an estimation model based on COCOMO.
- ESTIMACS:
  - It provides estimates of the effort, duration, cost and personnel requirements for maintenance and new application development projects.

- Checkpoint:
  - It guides the user through the development of a software project estimate and plan.
- Function Point Workbench:
  - 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:
  - Based on Putnam, COCOMO II
- SEER-SEM:
  - Predicts, measures and analyzes resources, staffing, schedules, risk and cost for software projects.
- ePM.Ensemble
  - It supports effort estimation among other things.

- CostXpert:
  - Based on COCOMO