Estimation Techniques

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Estimation techniques are of utmost importance in software development life cycle, where the time required to complete a particular task is estimated before a project begins. Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable. This tutorial discusses various estimation techniques such as estimation using Function Points, Use-Case Points, Wideband Delphi technique, PERT, Analogy, et

Audience

If you are an aspiring project manager or project leader, then this tutorial is definitely for you. It will take you through all the important estimation techniques.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of the Software Development Life Cycle (SDLC). In addition, you should have a basic understanding of software programming using any programming language.

Disclaimer & Copyright

Table of Contents

About the Tutorial i

Audience i

Prerequisites i

Disclaimer & Copyright i

Table of Contents ii

1. ESTIMATION TECHNIQUES – OVERVIEW 1

Observations on Estimation 1

General Project Estimation Approach 2

Estimation Accuracy 3

Estimation Issues 3

Estimation Guidelines 4

2. ESTIMATION TECHNIQUES – FUNCTION POINTS 6

ISO Standards 6

Object Management Group Specification for Automated Function Point 6

History of Function Point Analysis 6

Elementary Process (EP) 7

Functions 7

Definition of RETs, DETs, FTRs 9

3. ESTIMATION TECHNIQUES – FP COUNTING PROCESS 10

Step 1: Determine the Type of Count 10

Step 2: Determine the Boundary of the Count 11

Step 3: Identify Each Elementary Process Required by the User 11

Step 4: Determine the Unique Elementary Processes 11

Step 5: Measure Data Functions 11

Step 6: Measure Transactional Functions 13

Step 7: Calculate Functional Size (Unadjusted Function Point Count) **18**

Benefits of Function Points **21**

FP Repositories **22**

4. ESTIMATION TECHNIQUES – USE-CASE POINTS 23

Use-Case Points – Definition **23**

History of Use-Case Points **23**

Use-Case Points Counting Process **23**

Advantages and Disadvantages of Use-Case Points **29**

5. ESTIMATION TECHNIQUES – WIDEBAND DELPHI TECHNIQUE 30

Wideband Delphi Technique – Steps **30**

Advantages and Disadvantages of Wideband Delphi Technique **33**

6. ESTIMATION TECHNIQUES – THREE-POINT ESTIMATION 35

Standard Deviation **35**

Convert the Project Estimates to Confidence Levels **36**

7. ESTIMATION TECHNIQUES – PROJECT EVALUATION & REVIEW TECHNIQUE 37

Standard Deviation **37**

PERT Estimation Steps **38**

Convert the Project Estimates to Confidence Levels **38**

Differences between Three-Point Estimation and PERT **39**

8. ESTIMATION TECHNIQUES – ANALOGOUS ESTIMATION 40

Analogous Estimation – Definition **40**

Analogous Estimation Requirements **40**

Analogous Estimation Steps **41**

Advantages of Analogous Estimation **41**

9. ESTIMATION TECHNIQUES – WORK BREAKDOWN STRUCTURE 42

Representation of WBS **42**

Types of WBS **44**

Estimate Size **44**

Estimate Effort **44**

Scheduling **45**

Critical Path Method **45**

Task Dependency Relationships **45**

Gantt Chart **46**

Milestones **47**

Advantages of Estimation using WBS **47**

10. ESTIMATION TECHNIQUES – PLANNING POKER 48

Planning Poker Estimation **48**

Planning Poker Estimation Technique **48**

Benefits of Planning Poker Estimation **49**

11. ESTIMATION TECHNIQUES – TESTING 50

Percentage of Development Effort Method **50**

Testing Estimation Techniques **51**

1. Estimation Techniques – Overview

Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable.

Estimation determines how much money, effort, resources, and time it will take to build a specific system or product. Estimation is based on:

- Past Data/Past Experience
- Available Documents/Knowledge
- Assumptions
- Identified Risks

The four basic steps in Software Project Estimation are:

- Estimate the size of the development product.
- Estimate the effort in person-months or person-hours.
- Estimate the schedule in calendar months.
- Estimate the project cost in agreed currency.

Observations on Estimation

- Estimation need not be a one-time task in a project. It can take place during:
  - Acquiring a Project.
  - Planning the Project.
  - Execution of the Project as the need arises.

- Project scope must be understood before the estimation process begins. It will be helpful to have historical Project Data.

- Project metrics can provide a historical perspective and valuable input for generation of quantitative estimates.

- Planning requires technical managers and the software team to make an initial commitment as it leads to responsibility and accountability.

- Past experience can aid greatly.

- Use at least two estimation techniques to arrive at the estimates and reconcile the resulting values. Refer Decomposition Techniques in the next section to learn about reconciling estimates.

- Plans should be iterative and allow adjustments as time passes and more details are known.

General Project Estimation Approach

The Project Estimation Approach that is widely used is **Decomposition Technique**. Decomposition techniques take a divide and conquer approach. Size, Effort and Cost estimation are performed in a stepwise manner by breaking down a Project into major Functions or related Software Engineering Activities.

Step 1: Understand the scope of the software to be built.

Step 2: Generate an estimate of the software size.

- Start with the statement of scope.
- Decompose the software into functions that can each be estimated individually.
- Calculate the size of each function.
- Derive effort and cost estimates by applying the size values to your baseline productivity metrics.
- Combine function estimates to produce an overall estimate for the entire project.

Step 3: Generate an estimate of the effort and cost. You can arrive at the effort and cost estimates by breaking down a project into related software engineering activities.

- Identify the sequence of activities that need to be performed for the project to be completed.
- Divide activities into tasks that can be measured.
- Estimate the effort (in person hours/days) required to complete each task.
- Combine effort estimates of tasks of activity to produce an estimate for the activity.
- Obtain cost units (i.e., cost/unit effort) for each activity from the database.
- Compute the total effort and cost for each activity.
- Combine effort and cost estimates for each activity to produce an overall effort and cost estimate for the entire project.

Step 4: Reconcile estimates: Compare the resulting values from Step 3 to those obtained from Step 2. If both sets of estimates agree, then your numbers are highly reliable. Otherwise, if widely divergent estimates occur conduct further investigation concerning whether:

- The scope of the project is not adequately understood or has been misinterpreted.
- The function and/or activity breakdown is not accurate.

- Historical data used for the estimation techniques is inappropriate for the application, or obsolete, or has been misapplied.

Step 5: Determine the cause of divergence and then reconcile the estimates.

Estimation Accuracy

Accuracy is an indication of how close something is to reality. Whenever you generate an estimate, everyone wants to know how close the numbers are to reality. You will want every estimate to be as accurate as possible, given the data you have at the time you generate it. And of course you don't want to present an estimate in a way that inspires a false sense of confidence in the numbers.

Important factors that affect the accuracy of estimates are:

- The accuracy of all the estimate's input data.
- The accuracy of any estimate calculation.
- How closely the historical data or industry data used to calibrate the model matches the project you are estimating.
- The predictability of your organization's software development process.
- The stability of both the product requirements and the environment that supports the software engineering effort.
- Whether or not the actual project was carefully planned, monitored and controlled, and no major surprises occurred that caused unexpected delays.

Following are some guidelines for achieving reliable estimates:

- Base estimates on similar projects that have already been completed.
- Use relatively simple decomposition techniques to generate project cost and effort estimates.
- Use one or more empirical estimation models for software cost and effort estimation.

Refer to the section on Estimation Guidelines in this chapter.

To ensure accuracy, you are always advised to estimate using at least two techniques and compare the results.

Estimation Issues

Often, project managers resort to estimating schedules skipping to estimate size. This may be because of the timelines set by the top management or the marketing team. However,

whatever the reason, if this is done, then at a later stage it would be difficult to estimate the schedules to accommodate the scope changes.

While estimating, certain assumptions may be made. It is important to note all these assumptions in the estimation sheet, as some still do not document assumptions in estimation sheets.

Even good estimates have inherent assumptions, risks, and uncertainty, and yet they are often treated as though they are accurate.

The best way of expressing estimates is as a range of possible outcomes by saying, for example, that the project will take 5 to 7 months instead of stating it will be complete on a particular date or it will be complete in a fixed no. of months. Beware of committing to a range that is too narrow as that is equivalent to committing to a definite date.

- You could also include uncertainty as an accompanying probability value. For example, there is a 90% probability that the project will complete on or before a definite date.

- Organizations do not collect accurate project data. Since the accuracy of the estimates depend on the historical data, it would be an issue.

- For any project, there is a shortest possible schedule that will allow you to include the required functionality and produce quality output. If there is a schedule constraint by management and/or client, you could negotiate on the scope and functionality to be delivered.

- Agree with the client on handling scope creeps to avoid schedule overruns.

- Failure in accommodating contingency in the final estimate causes issues. For e.g., meetings, organizational events.

- Resource utilization should be considered as less than 80%. This is because the resources would be productive only for 80% of their time. If you assign resources at more than 80% utilization, there is bound to be slippages.

Estimation Guidelines

One should keep the following guidelines in mind while estimating a project:

- During estimation, ask other people's experiences. Also, put your own experiences at task.

- Assume resources will be productive for only 80 percent of their time. Hence, during estimation take the resource utilization as less than 80%.

- Resources working on multiple projects take longer to complete tasks because of the time lost switching between them.

- Include management time in any estimate.

- Always build in contingency for problem solving, meetings and other unexpected events.

- Allow enough time to do a proper project estimate. Rushed estimates are inaccurate, high-risk estimates. For large development projects, the estimation step should really be regarded as a mini project.

- Where possible, use documented data from your organization's similar past projects. It will result in the most accurate estimate. If your organization has not kept historical data, now is a good time to start collecting it.

- Use developer-based estimates, as the estimates prepared by people other than those who will do the work will be less accurate.

- Use several different people to estimate and use several different estimation techniques.

- Reconcile the estimates. Observe the convergence or spread among the estimates. Convergence means that you have got a good estimate. Wideband-Delphi technique can be used to gather and discuss estimates using a group of people, the intention being to produce an accurate, unbiased estimate.

- Re-estimate the project several times throughout its life cycle.

2. Estimation Techniques – Function Points

Function Point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure software size. They are widely accepted as an industry standard for functional sizing.

For sizing software based on FP, several recognized standards and/or public specifications have come into existence. As of 2013, these are:

ISO Standards

- **COSMIC**: ISO/IEC 19761:2011 Software engineering. A functional size measurement method.
- **FiSMA**: ISO/IEC 29881:2008 Information technology - Software and systems engineering - FiSMA 1.1 functional size measurement method.
- **IFPUG**: ISO/IEC 20926:2009 Software and systems engineering - Software measurement - IFPUG functional size measurement method.
- **Mark-II**: ISO/IEC 20968:2002 Software engineering - Ml II Function Point Analysis - Counting Practices Manual.
- **NESMA**: ISO/IEC 24570:2005 Software engineering - NESMA function size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis.

Object Management Group Specification for Automated Function Point

Object Management Group (OMG), an open membership and not-for-profit computer industry standards consortium, has adopted the Automated Function Point (AFP) specification led by the Consortium for IT Software Quality. It provides a standard for automating FP counting according to the guidelines of the International Function Point User Group (IFPUG).

Function Point Analysis (FPA) technique quantifies the functions contained within software in terms that are meaningful to the software users. FPs consider the number of functions being developed based on the requirements specification.

Function Points (FP) Counting is governed by a standard set of rules, processes and guidelines as defined by the International Function Point Users Group (IFPUG). These are published in Counting Practices Manual (CPM).

History of Function Point Analysis

The concept of Function Points was introduced by Alan Albrecht of IBM in 1979. In 1984, Albrecht refined the method. The first Function Point Guidelines were published in 1984. The International Function Point Users Group (IFPUG) is a US-based worldwide organization of Function Point Analysis metric software users. The **International Function Point Users Group (IFPUG)** is a non-profit, member-governed organization founded in 1986. IFPUG owns Function Point Analysis (FPA) as defined in ISO standard 20296:2009 which specifies the definitions, rules and steps for applying the IFPUG's functional size measurement (FSM) method. IFPUG maintains the Function Point Counting Practices Manual (CPM). CPM 2.0 was released in 1987, and since then there have been several iterations. CPM Release 4.3 was in 2010.

The CPM Release 4.3.1 with incorporated ISO editorial revisions was in 2010. The ISO Standard (IFPUG FSM) - Functional Size Measurement that is a part of CPM 4.3.1 is a technique for measuring software in terms of the functionality it delivers. The CPM is an internationally approved standard under ISO/IEC 14143-1 Information Technology – Software Measurement.

Elementary Process (EP)

Elementary Process is the smallest unit of functional user requirement that:

- Is meaningful to the user.
- Constitutes a complete transaction.
- Is self-contained and leaves the business of the application being counted in a consistent state.

Functions

There are two types of functions:

- Data Functions
- Transaction Functions

Data Functions

There are two types of data functions:

- Internal Logical Files
- External Interface Files

Data Functions are made up of internal and external resources that affect the system.

Internal Logical Files

Internal Logical File (ILF) is a user identifiable group of logically related data or control information that resides entirely within the application boundary. The primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted. An ILF has the inherent meaning that it is internally maintained, it has some logical structure and it is stored in a file. (Refer Figure 1)

External Interface Files

External Interface File (EIF) is a user identifiable group of logically related data or control information that is used by the application for reference purposes only. The data resides entirely outside the application boundary and is maintained in an ILF by another application. An EIF has the inherent meaning that it is externally maintained, an interface has to be developed to get the data from the file. (Refer Figure 1)

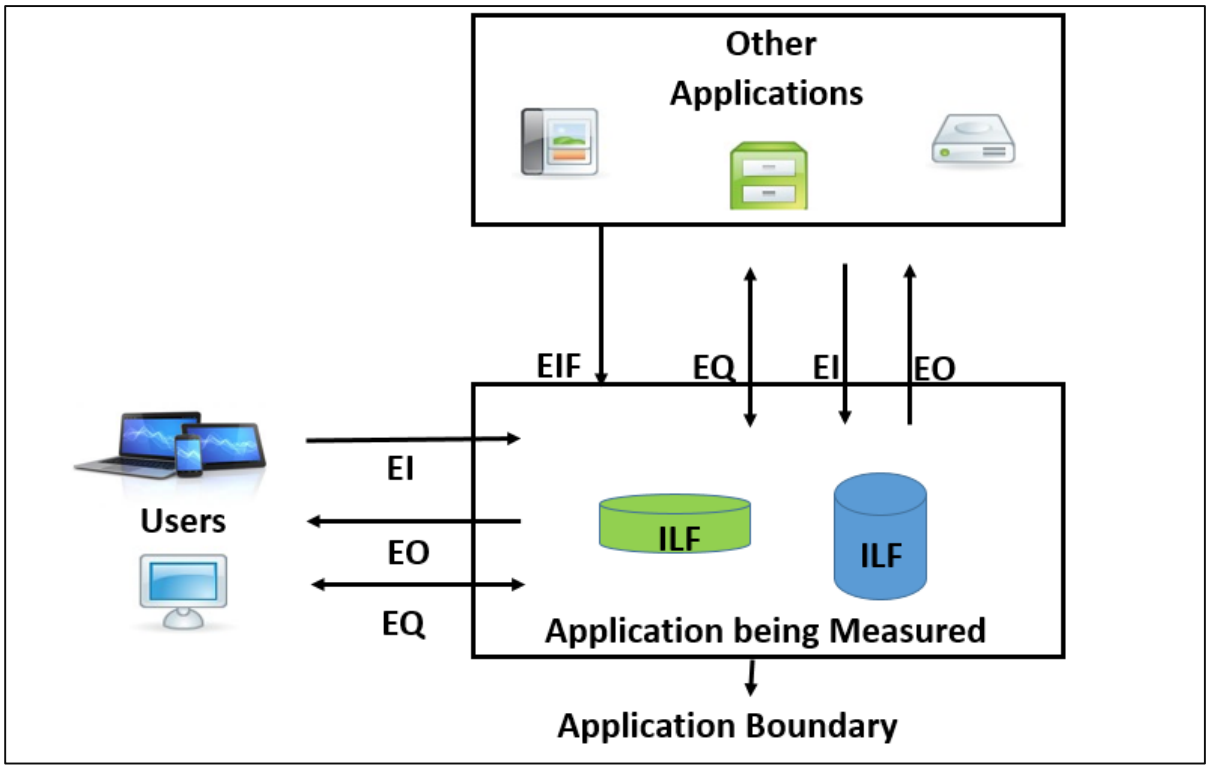

Figure 1: Application Boundary, Data Functions, Transaction Functions

Transaction Functions

There are three types of transaction functions.

- External Inputs
- External Outputs
- External Inquiries

Transaction functions are made up of the processes that are exchanged between the user, the external applications and the application being measured.

External Inputs

External Input (EI) is a transaction function in which Data goes "into" the application from outside the boundary to inside. This data is coming external to the application.

- Data may come from a data input screen or another application.
- An EI is how an application gets information.
- Data can be either control information or business information.
- Data may be used to maintain one or more Internal Logical Files.
- If the data is control information, it does not have to update an Internal Logical File. (Refer Figure 1)

External Outputs

External Output (EO) is a transaction function in which data comes "out" of the system. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications. (Refer Figure 1)

External Inquiries

External Inquiry (EQ) is a transaction function with both input and output components that result in data retrieval. (Refer Figure 1)

Definition of RETs, DETs, FTRs

Record Element Type

A Record Element Type (RET) is the largest user identifiable subgroup of elements within an ILF or an EIF. It is best to look at logical groupings of data to help identify them.

Data Element Type

Data Element Type (DET) is the data subgroup within an FTR. They are unique and user identifiable.

File Type Referenced

File Type Referenced (FTR) is the largest user identifiable subgroup within the EI, EO, or EQ that is referenced to.

The transaction functions EI, EO, EQ are measured by counting FTRs and DETs that they contain following counting rules. Likewise, data functions ILF and EIF are measured by counting DETs and RETs that they contain following counting rules. The measures of transaction functions and data functions are used in FP counting which results in the functional size or function points.