CHAPTER-23
ESTIMATION
WHAT IS ESTIMATION?

A real need for software has been established stakeholders are on boards.

Software Engineers are ready to start and the project is about to begin but how do we proceed.

Software project planning encompasses five more activities:

# Estimation
# Scheduling
# Risk analysis
# Quality management planning
# Change management planning
Estimation is attempt to determine how much money, effort, resources & time it will take to build a specific software based system or project.
• Degree of uncertainty as a matter of course.
• planning involves estimation—your attempt to determine how much money, how much effort, how many resources, and how much time it will take to build a specific software-based system or product.
Why is it important?

• Would you build a house without knowing how much you were about to spend?
• Of course not, and since most computer-based systems
  • and products cost considerably more to build
  • than a large house, it would seem reasonable to
  • develop an estimate before you start creating the
• software.
Who does it?

• Software managers—using information solicited from customers and software engineers
• software metrics data collected from past projects.
WHAT ARE THE STEPS FOR ESTIMATION?

Estimation begins with a description of the scope of the product. The problem is then decomposed into a set of smaller problems and each of these is estimated using historical data and experience as guides.
Problem complexity and risk are considered before a final estimate is made.

Estimation risk is measured by the degree of uncertainty in the quantitative estimates established by the resources, cost and schedule.
Process-Based Estimation

- The most common technique for estimating a project is to base the estimate on the process that will be used. That is, the process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated.
- Like the problem-based techniques, process-based estimation begins with a delineation of software functions obtained from the project scope. A series of software process activities must be performed for each function. Functions and related software process activities may be represented as part of a table similar to the one presented.
Once problem functions and process activities are melded, the planner estimates the effort (e.g., person-months) that will be required to accomplish each software process activity for each software function. These data constitute the central matrix of the table in Figure 3.2. Average labour rates (i.e., cost/unit effort) are then applied to the effort estimated for each process activity. It is very likely the labour rate will vary for each task. Senior staff heavily involved in early activities are generally more expensive than junior staff involved in later design tasks, code generation, and early testing.
Problem-Based Estimation

- Lines of code and function points were described as measures from which productivity metrics can be computed. LOC and FP data are used in two ways during software project estimation: (1) as an estimation variable to "size" each element of the software and (2) as baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections.
LOC and FP estimation are distinct estimation techniques. Yet both have a number of characteristics in common. The project planner begins with a bounded statement of software scope and from this statement attempts to decompose software into problem functions that can each be estimated individually. LOC or FP (the estimation variable) is then estimated for each function. Alternatively, the planner may choose another component for sizing such as classes or objects, changes, or business
A typical Empirical model is derived using regression analysis on data collected from past software projects. The overall structure of such models takes the form
\[ E = A + B \times (ev)^{\text{power}C} \]
where \( A, B, \) and \( C \) are empirically derived constants, \( E \) is effort in person-months, and \( ev \) is the estimation variable.
DECOMPOSITION MODEL

The model which decompose the problem, recharacterizing it as a set of smaller (and hopefully, more manageable) problems is known as Decomposition Model.
An Example of Decomposition Model:

As an example of Decomposition Model problem-based estimation techniques, let us consider a software package to be developed for a computer-aided design application for mechanical components.

A review of the System Specification indicates that the software is to execute on an engineering workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high resolution colour display and laser printer.
COCOMO II

COCOMO stands for Constructive Cost Model. The original COCOMO model became one of the most widely used and discussed software cost estimation models in the industry. It has evolved into a more comprehensive estimation model, called COCOMO II.
AREAS OF COCMO II

Application composition model. Used during the early stages of software engineering.

Early design stage model. Used once requirements have been stabilized and basic software architecture has been established.

Post-architecture-stage model. Used during the construction of the software.
Using customer requirements, project managers estimate costs and materials needed to complete a task. Because a project estimate often results in a contract, project managers combine analysis, data and experience to construct a reasonable estimate. Inputs to a project estimate include project scope, schedule, a human resource plan and risk mitigation, according to the book "Project Management Body of Knowledge." Project cost estimates include both direct and indirect costs.
Project managers, with significant experience, can create valid estimates based on expert judgment. This method applies most often to simple projects or when providing a high-level estimate prior to detailed analysis. For example, a licensed electrician applies knowledge and experience to calculate the cost of a project to install ceiling lights in a gymnasium by looking at the space to be covered and the amount of illumination desired.
What is an Object point?

- The object point is determined by multiplying the original number of objects by the waiting factor and summing to obtain a total object point count.
- When component-based development or general software reused is to be applied, the percent of reused (%reuse) is estimated in the object point count is adjusted.

\[ NOP = (\text{object points}) \times [(100 - \text{%reuse}) \times 100] \]
What is an Object point? (cont....)

- To derive an estimate of effort based on computed NOP value, a “productivity rate” must be derived.

\[ \text{PROD} = \frac{\text{NOP}}{\text{person-month}} \]

- Once the productivity rate has been determined, an estimate of project effort can be derived as.

\[ \text{Estimated effort} = \frac{\text{NOP}}{\text{PROD}} \]
Productivity rates for object points

<table>
<thead>
<tr>
<th>Developer's experience/capability</th>
<th>Very low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very high</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment maturity/capability</td>
<td>Very low</td>
<td>Low</td>
<td>Nominal</td>
<td>High</td>
<td>Very high</td>
</tr>
<tr>
<td>PROD</td>
<td>4</td>
<td>7</td>
<td>13</td>
<td>25</td>
<td>50</td>
</tr>
</tbody>
</table>
Estimation for object-oriented project

Approaches:

1) Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.

2) Using object-oriented analysis modeling, develop use-case and determine a count.

3) From the analysis model, determine the number of key classes.
Creating a decision tree

Steps:

1) Build system X from scratch

2) Reuse existing “partial-experience” components to construct the system

3) Buy an available software product and modify it to meet local needs

4) Contract the software development to an outside vendor
Creating a decision tree (contd..)

**Expected value for cost:**

Expected Cost = ∑ (path probability)\(i\) \(\times\) (estimated path cost)\(i\)

**For the build path,**

Expected Cost\(\text{build}\) = 0.30 ($380K) + 0.70 ($450K) = $429K

**The expected costs for these paths are,**

Expected Cost\(\text{reuse}\) = 0.40 ($275K) + 0.60 [0.20($310K) + 0.80($490K)] = $382K

Expected Cost\(\text{buy}\) = 0.70($210K) + 0.30($400K) = $267K

Expected Cost\(\text{contract}\) = 0.60($350K) + 0.40($500K) = $410K
Decision tree

System X

- Build
  - Simple (0.30): $380,000
  - Difficult (0.70): $450,000
  - Minor changes (0.40): $275,000

- Reuse
  - Simple (0.20): $310,000
  - Complex (0.80): $490,000
  - Minor changes (0.70): $210,000

- Buy
  - Major changes (0.60)
    - Major changes (0.30): $400,000
    - Without changes (0.60): $350,000

- Contract
  - With changes (0.40): $500,000