Chapter 8

Managing Software Projects

Recall SAD

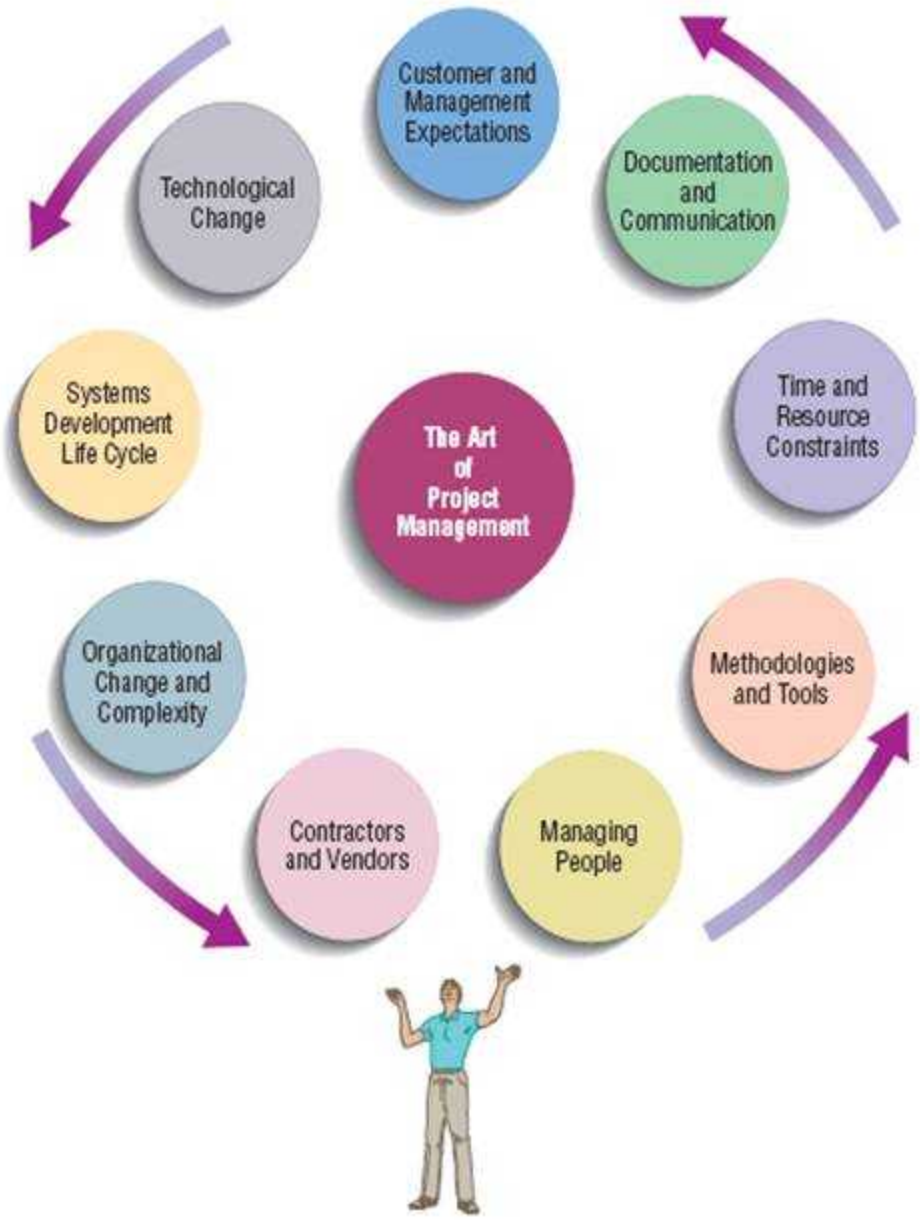
Managing the Information Systems Project

- Focus of Project Management
  - To assure that information system projects meet customer expectations
    - Delivered in a timely manner
    - Meet constraints and requirements
- PM's Motto:
  - High quality deliverables, on time, on budget

Managing the Information Systems Project

- Project Manager
  - Requires diverse set of skills
    - Planning
    - Organizing
    - Monitor and Control
    - Leadership
  - Customer relations
  - Technical problem solving
  - Conflict management
  - Communications
  - Risk management
  - Change management

FIGURE 3-5
A project manager juggles numerous activities.



Project Management Process

- Project
  - Planned undertaking of related activities to reach an objective that has a start and an end
- Launched by a System Services Request
- Four Phases
  - Initiating
  - Planning
  - Executing
  - Closing down

Project Management Concept

- Project management involves the planning, monitoring, and control of people, process, and events that occur during software development.
- It is concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organizations developing and procuring the software.
- Project management is needed because software development is always subject to budget and schedule constraints that are set by the organization developing the software
- Project management is the Organizing, planning and scheduling software projects

CTD...

- **Managers must focus on the fours P's to be successful (people, product, process, and project) i.e., it defines the process and task to be conducted, the people who will do the work, and the mechanisms for assessing risks, controlling change and evaluating quality.**
- **A project plan is a document that defines the four P's in such a way as to ensure a cost effective, high quality software product.**

Management Spectrum

1) **People** (The PM-CMM's Key practice areas includes recruiting, selection, performance management, training, compensation, career development, organization, work design, team/culture development)
2) **Product** (product objectives, scope, alternative solutions)
3) **Process** (framework activities populated with tasks, milestones, work products, and QA points)
4) **Project** (planning, monitoring, controlling)

People

- **Players** (senior managers, technical managers, practitioners, customers, end-users)
- **People Management- Capability Maturity Model (PM-CMM)** enhance the readiness of software organizations to undertake increasingly complex applications by helping to attract, grow, motivate, deploy, and retain the talent needed to improve their software development capability.
- The most important thing you do for a project is selecting the staff.

The Product

- Software scope (context, information objectives, function, performance)
- Focus is on functionality to be delivered and the process used to deliver it
- Communication and coordination with the customer and other stakeholders must occur so that product scope and requirements are understood.
- Objectives identify the overall goal for the product (from the customers' point of view) without considering how these goals will be achieved.

Process

- Process model chosen must be appropriate for the: customers and developers, characteristics of the product, and project development environment
- Project planning begins with melding the product and the process .
- Each function to be engineered must pass though the set of framework activities defined for a software organization
- Work tasks may vary but the common process framework (CPF) is invariant (project size does not change the CPF)
- The job of the software engineer is to estimate the resources required to move each function though the framework activities to produce each work product
- Project decomposition begins when the project manager tries to determine how to accomplish each CPF activity.

Project

- The project plan defines the process and tasks to be conducted, the people who will do the work and the mechanism for assessing risks, controlling change and evaluating quality.
- To manage a successful software project, we must understand what can go wrong (so that problems can be avoided).

W5HH Principle

- Why is the system being developed?
- What will be done by When?
- Who is responsible for a function?
- Where are they organizationally located?
- How will the job be done technically and managerially?
- How much of each resource is needed

Project [CTD...]

John Reel defines 10 signs that indicate that an IS project is in Jeopardy:-

1. s/w people don't understand their customer's needs.
2. The product scope is poorly defined.
3. Changes are managed poorly.
4. The chosen technology changes.
5. Business needs change (or ill defined)
6. Deadlines are unrealistic
7. Users are resistant
8. Sponsorship is lost (or was never properly obtained)
9. The project team lacks people with appropriate skills.
10. Managers and practitioners avoid best practices and lessons learned.

Software Project Planning-Overview

- Software planning involves estimating how much time, effort, money, and resources will be required to build a specific software system. After the project scope is determined and the problem is decomposed into smaller problems, software managers use historical project data (as well as personal experience and intuition) to determine estimates for each.
- The resulting work product is called a project management plan.

Project Planning Objectives

- To provide a framework that enables software manager to make a reasonable estimate of resources, cost, and schedule.
- Project outcomes should be bounded by 'best case' and 'worst case' scenarios.
- Estimates should be updated as the project progresses

Types of Project Plan

Plan	Description
Quality Plan	Describes the quality procedures and standards that will be used in a project.
Validation Plan	Describes the approach, resources and schedules used for system validation.
Confirmation management Plan	describes the configuration management procedures and structures and structures to be used.
Maintenance plan	Predicts the maintenance requirements of the system, maintenance cost, and effort required.
Staff development plan	Describes how the skills and experience of the project team members will be developed.

Software Project Planning activities

Step1: Determination of software scope and feasibility study

Step 2: Estimation of the resources

Step1: Determination of software scope

- Describes the data to be processed and produced, control parameters, function, performance (wrt processing and response time requirement), constraints (wrt limits placed on the software by external hardware, available memory, or other existing system), external interfaces, and reliability.

Software scope is defined by answering following questions.

- Context: How does the s/w to be built fit into a larger system, product or business context?
- Information objectives: what customer visible data objects are produced as output from the software? What data objects are required for input?
- Function and Performance: what functions does the software perform to transform input data into outputs?

CTD...

Customer Communication and Scope

Necessary information is obtained by conducting a preliminary meeting or interview with customer, where Analyst prepare a set of context free questions in order to....

- **Determine the customer's overall goals for the proposed system and any expected benefits.**
- **Evaluate the effectiveness of the customer meeting**

FAST (Facilitated Application Specification Techniques)

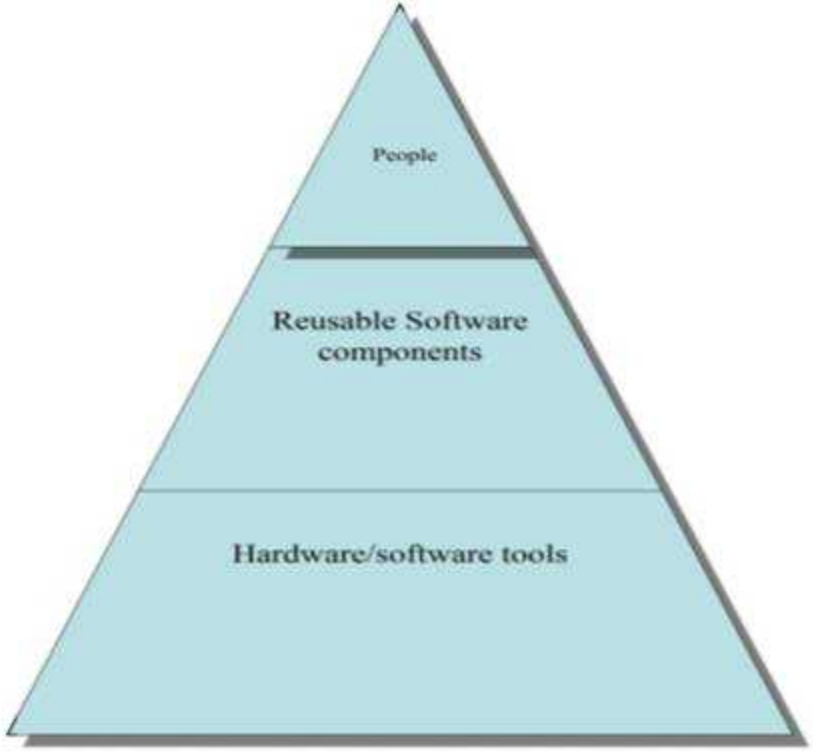
It is a team-oriented approach to requirement gathering that can be applied to help establish the scope of a project. It encourages the creation of a joint team of customer and development who work together to identify the problem, propose elements of the solution, negotiate different approaches & specify a preliminary set of requirements.

Feasibility Study

Do yourself...steps involved, types and figure

Step 2: Estimation of Resources

- Human Resources (number of people required and skills needed to complete the development project)
- Reusable Software Resources (off-the-shelf components, full-experience components, new components)
- Development Environment (hardware and software required to be accessible by software team during the development process)



Project Scheduling

- Software project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.
- Split project into tasks and estimate time and resources required to complete each task
- Organize tasks concurrently to make optimal use of workforce
- Minimize task dependencies to avoid delays caused by one task waiting for another to complete

Things done before project scheduling.....

- Selection of an appropriate process model
- Identification of the software engineering tasks
- Estimation of Work size, people
- Calculation of deadline
- Risk analysis

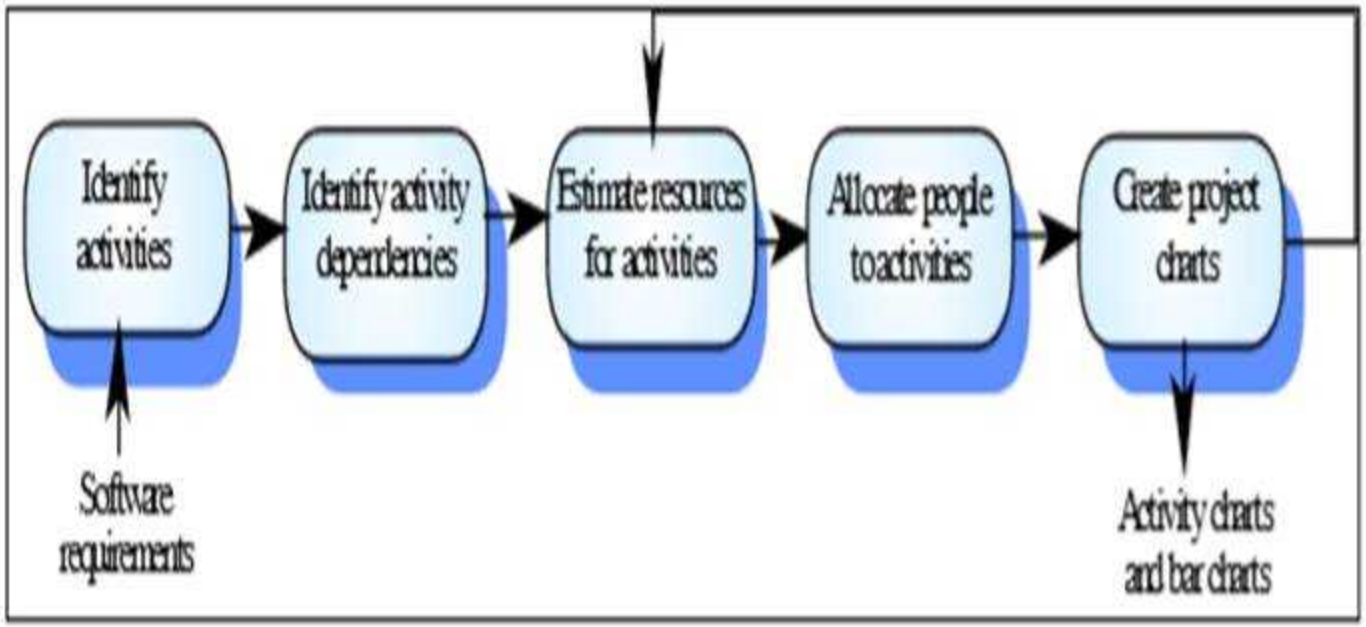
Things to do during scheduling....

- Creation of activity network (Effort and duration allocated to each task and a task network)
- Assign responsibility for each task

Software Project Scheduling Principles

- Compartmentalization - the product and process must be decomposed into a manageable number of activities and tasks
- Interdependency - tasks that can be completed in parallel must be separated from those that must be completed serially
- Time allocation - every task has start and completion dates
- Effort validation - project manager must ensure that on any given day there are enough staff members assigned to complete the tasks within the time estimated in the project plan
- Defined Responsibilities - every scheduled task needs to be assigned to a specific team member
- Defined outcomes - every task in the schedule needs to have a defined outcome (usually a work product or deliverable)
- Defined milestones - a milestone is accomplished when one or more work products from an engineering task have passed quality review

Project Scheduling Process



Risk Management

If you don't attack risks, they will attack you.

Risk Management [Ctd...]

- Risk is an uncertainty.
- We don't know whether a particular event will occur or no but if it does has a negative impact on a project.
- An example would be that team is working on a project and the developer walks out of project and other person is recruited n his place and he doesn't work on the same platform and converts it into the platform he is comfortable with. Now the project has to yield the same result in the same time span. Whether they will be able to complete the project on time. That is the risk of schedule .

Ctd...

- The Risks we encounter in a project should be resolved so that we are able to deliver the desired project to the customer.
- The project should be managed in such a way that the risks don't affect the project in a big way.
- The art of managing of the risks effectively so that the WIN-WIN situation and friendly relationship is established between the team and the customer is called **Risk Management.**
- By using various paradigms, principles we can manage the risks

Risk Management

- In the field of software engineering, risk management is a methodology or a mechanism, carried out throughout the development process to identify, manage and control risks evolved before and during the development process.
- Basically, three types of activities are covered under the risk management process.
  - Risk Identification.
  - Risk Analysis.
  - Risk Control.

Risk Categories

- performance risks
  - does not meet the requirements
- cost risks
  - over budget
- schedule risks
  - not delivered on time
- support risks
  - maintenance problems

Boehm's Top 10 Software Risks

1. Schedules, budgets, process
2. Requirements Changes
3. Personnel Shortfalls
4. Requirements Mismatch
5. Rapid change
6. Architecture, performance, quality, distribution/mobility
7. External components
7. Legacy Software
9. Externally-performed tasks
10. User interface mismatch

SEI Risk Management Principles

1. Global perspective
2. Forward-looking view
3. Open communications
4. Integrated management
5. Continuous process

Risk Management Principles

1.Global Perspective:

In this we look at the larger system definitions, design and implementation. We look at the opportunity and the impact the risk is going to have .

2.Forward Looking View:

Looking at the possible uncertainties that might creep up. We also think for the possible solutions for those risks that might occur in the future.

3.Open Communication:

This is to enable the free flow of communication between in the customers and the team members so that they have clarity about the risks.

4.Integrated management:

In this phase risk management is made an integral part of project management.

5.Continous process :

In this phase the risks are tracked continuously throughout the risk management paradigm.

Risk management paradigm

1. **Identify:** Search for the risks before they create a major problem

2. **Analyze:** understand the nature , kind of risk and gather information about the risk.

3. **Plan:** convert them into actions and implement them.

4. **Track:** we need to monitor the necessary actions.

5. **Control:** Correct the deviation and make any necessary amendments.

6. **Communicate:** Discuss about the emerging risks and the current risks and the plans to be undertaken.

Risk Management in Project management:

Basically project management deals with following :

1. **Planning:** Looking for the desired results, the strategies to be applied.

2. **Organizing:** Getting all the things together so that the desired results are obtained. By organizing the efficiency is increased and lot of time is saved.

3. **Directing:** Communication takes place and exchange of ideas is formatted in this phase.

4. **Controlling:** In the last phase feedback and evaluation is done.

Software Cost Estimation

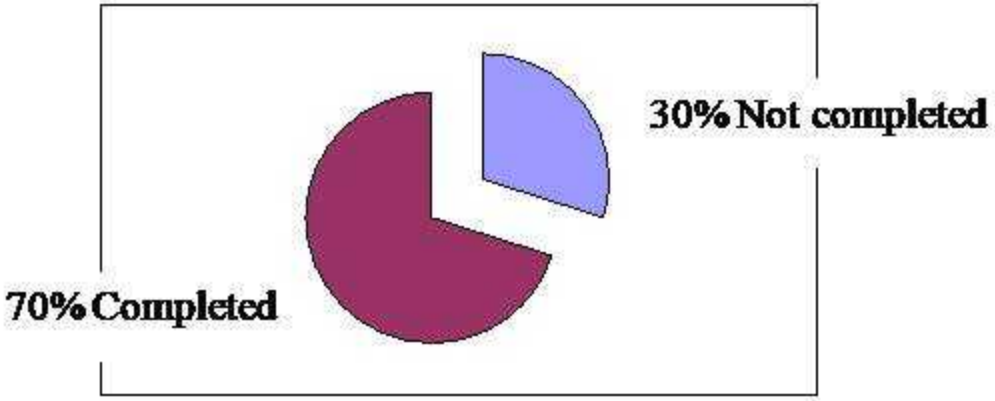
Objectives

- To introduce the fundamentals of software costing and pricing
- To explain software productivity metric
- To explain why different techniques for software estimation:
  - LOC model
  - Function points model
  - Object point model
  - COCOMO (COnstructive COst MOdel)
  - UCP: Use Case Points

What is Software Cost Estimation ?

Predicting the cost of resources required for a software development process.

Software is a Risky Business



- 53% of projects cost almost 200% of original estimate.
- Estimated $81 billion spent on failed U.S. projects in 1995.
  - All surveyed projects used waterfall lifecycle.

Software is a Risky Business

- British Computer Society (BCS) survey:
  - 1027 projects
  - Only 130 were successful !

- Success was defined as:
  - deliver all system **requirements**
  - within **budget**
  - within **time**
  - to the **quality** agreed on

Why early Cost Estimation?

- Cost estimation is needed early for s/w pricing
- S/W price = cost + profit

Fundamental estimation questions

- **Effort**
  - How much effort is required to complete an activity?
  - Units: man-day (person-day), man-week, man-month,,..
- **Duration**
  - How much calendar time is needed to complete an activity? Resources assigned
  - Units: hour, day, week, month, year,..
- **Cost of an activity**
  - What is the total cost of an activity?

Factor affecting software pricing

Factor	Description
Market opportunity	A development organisation may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the opportunity of more profit later. The experience gained may allow new products to be developed.
Cost estimate uncertainty	If an organisation is unsure of its cost estimate, it may increase its price by some contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organisation may lower its price to win a contract. After the contract is awarded, high prices may be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a small profit or break even than to go out of business.

Productivity measures

S/W productivity measures are based on:

- Size related measures:
  - Based on some output from the software process
  - Number lines of delivered source code (LOC)
- Function-related measures
  - based on an estimate of the functionality of the delivered software:
    - **Function-points (are the best known of this type of measure)**
    - **Object-points**
    - **UCP**

Software Cost Components

- Hardware and software costs
- Travel and training costs
- Effort costs (the dominant factor in most projects)
  - salaries of engineers involved in the project
  - Social and insurance costs
- Effort costs must take **overheads** into account
  - costs of building, heating, lighting
  - costs of networking and communications
  - costs of shared facilities (e.g library, staff restaurant, etc.)

Productivity Measures

- **Size-related measures** based on some output from the software process. This may be lines of delivered source code, object code instructions, etc.
- **Function-related measures** based on an estimate of the functionality of the delivered software. Function-points are the best known of this type of measure

Function Points

- Based on a combination of program characteristics
  - external inputs and outputs
  - user interactions
  - external interfaces
  - files used by the system

Object Points

- Object points are an alternative function-related measure to function points
- Object points are NOT the same as object classes
- The number of object points in a program is a weighted estimate of
  - The number of separate **screens** that are displayed
  - The number of **reports** that are produced by the system
  - The number of **modules** that must be developed

Estimation Techniques

- There is no simple way to make an accurate estimate of the effort required to develop a software system
  - Initial estimates are based on inadequate information in a user requirements definition
  - The software may run on unfamiliar computers or use new technology
  - The people in the project may be unknown
- Project cost estimates may be self-fulfilling
  - The estimate defines the budget and the product is adjusted to meet the budget

Estimation techniques

- Algorithmic cost modelling
- Expert judgement
- Estimation by analogy
- Parkinson's Law
- Pricing to win

Algorithmic code modelling

- A formula – empirical relation:
  - based on historical cost information and which is generally based on the size of the software
- The formulae used in a formal model arise from the analysis of historical data.

Expert Judgement

- One or more experts in both software development and the application domain use their experience to predict software costs. Process iterates until some consensus is reached.
- Advantages: Relatively cheap estimation method. Can be accurate if experts have direct experience of similar systems
- Disadvantages: Very inaccurate if there are no experts!

Estimation by Analogy

- Experience-based Estimates
- The cost of a project is computed by comparing
  the project to a similar project in the same application domain
- Advantages: Accurate if project data available
- Disadvantages: Impossible if no comparable project has been tackled. Needs systematically maintained cost database

Parkinson's Law

- "The project costs whatever resources are available"

  (Resources are defined by the software house)

- Disadvantages: System is usually unfinished
  - The work is contracted to fit the budget available: by reducing functionality, quality
  - **Parkinson's Law states that work expands to fill the time available. The cost is determined by available resources rather than by objective statement.**
  - **Example: Project should be delivered in 12 months and 5 people are available. Effort = 60 p/m**

Pricing to Win

- The project costs whatever the customer budget is.
- Advantages: You get the contract
- Disadvantages:
  - The probability that the customer gets the system he/she wants is small.
  - Costs do not accurately reflect the work required

COCOMO (Constructive Cost Model)

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry Boehm. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

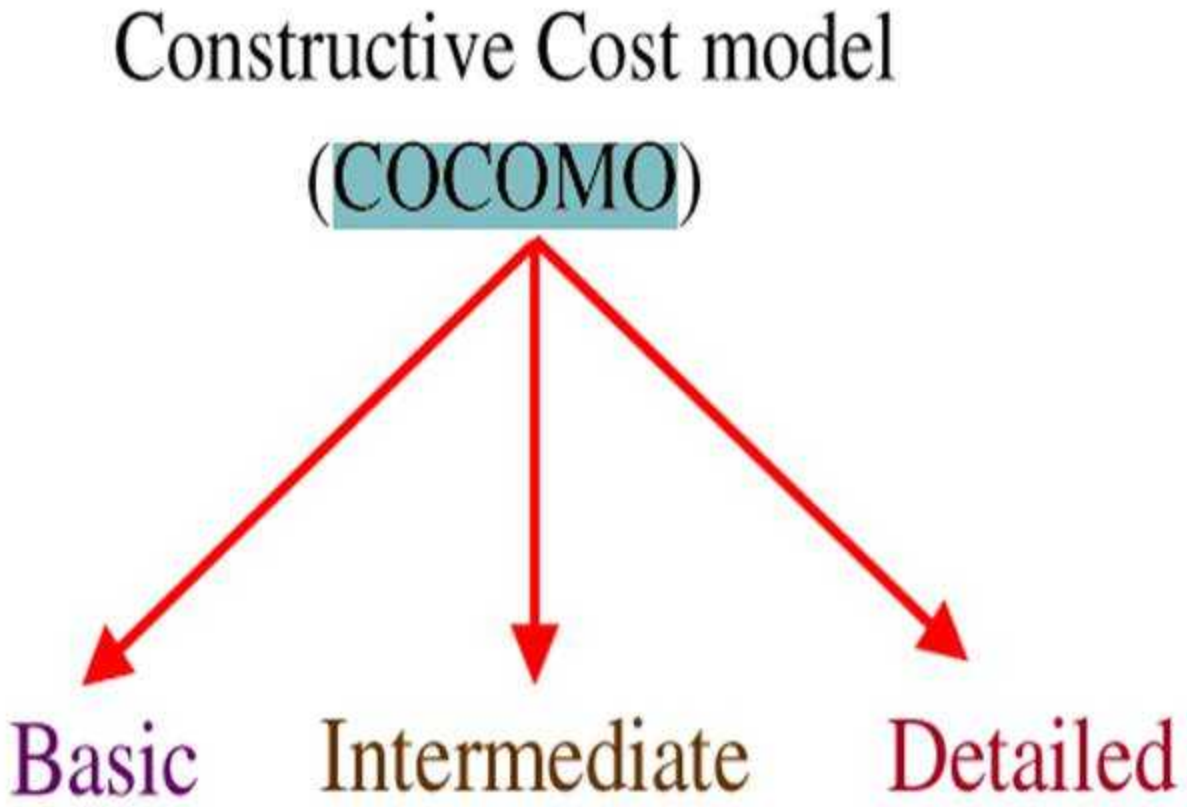
COCOMO applies to three classes of software projects:

* Organic projects - "small" teams with "good" experience working with "less than rigid" requirements

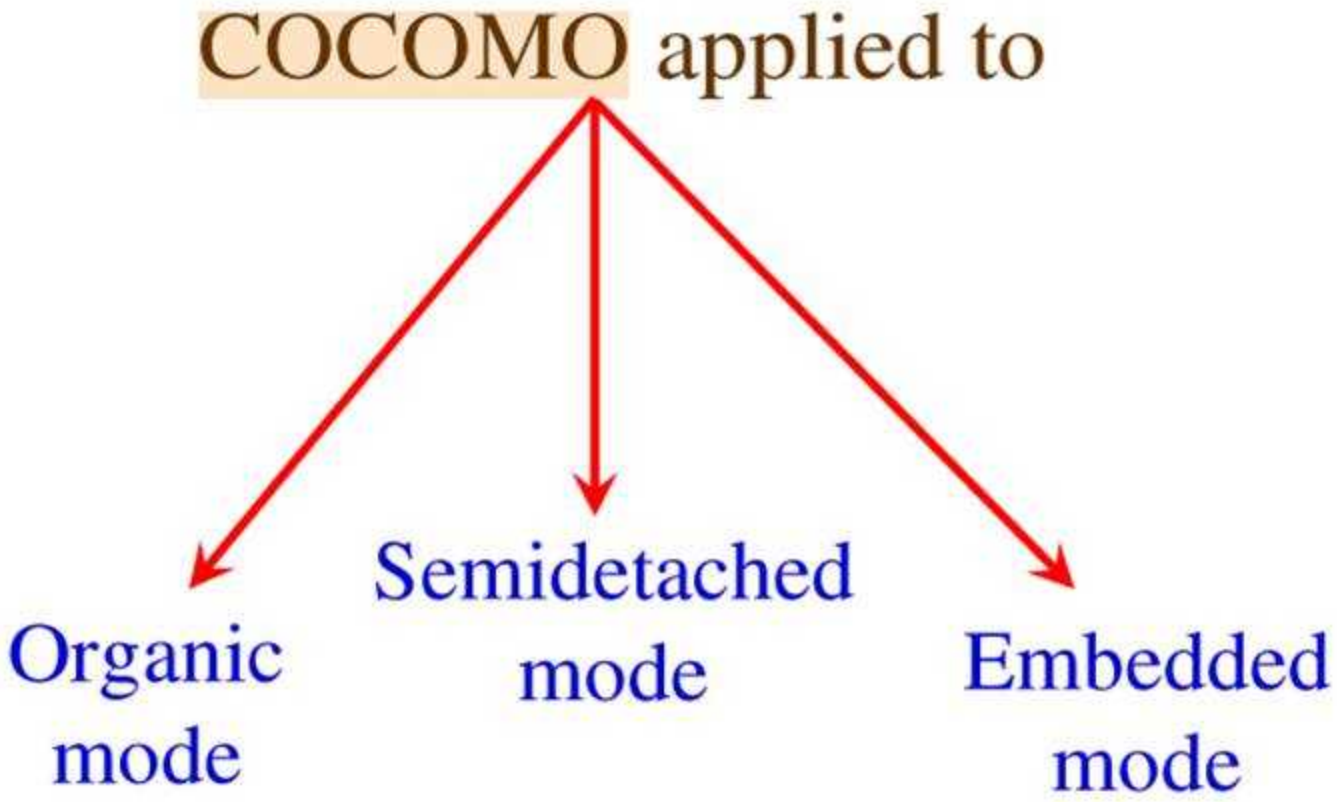
* Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements

* Embedded projects - developed within a set of "tight" constraints (hardware, software, operational

The Constructive Cost Model (COCOMO)



COCOMO [CTD...]


COCOMO applied to
Organic mode
Semidetached mode
Embedded mode

Comparison of Three COCOMO Models

Mode	Project size	Nature of Project	Innovation	Deadline of the project	Development Environment
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & In house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer Interfaces required

COCOMO

Basic COCOMO model takes the form

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (E)^{d_b}$$

where E is effort applied in Person-Months, and D is the development time in months. The coefficients a_b, b_b, c_b and d_b are given in table 4 (a).

COCOMO Coefficients

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

COCOMO…

When effort and development time are known, the average staff size to complete the project may be calculated as:

$$\text{Average staff size } (SS) = \frac{E}{D} \, Persons$$

When project size is known, the productivity level may be calculated as:

$$\text{Productivity } (P) = \frac{KLOC}{E} \, KLOC / PM$$

COCOMO

Suppose that a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three modes i.e., organic, semidetached and embedded.

Solution

The basic COCOMO equation take the form:

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (KLOC)^{d_b}$$

Estimated size of the project = 400 KLOC

(i) Organic mode

$$E = 2.4(400)^{1.05} = 1295.31 \text{ PM}$$

$$D = 2.5(1295.31)^{0.38} = 38.07 \text{ PM}$$

CTD...

(ii) Semidetached mode

$$E = 3.0(400)^{1.12} = 2462.79 \text{ PM}$$

$$D = 2.5(2462.79)^{0.35} = 38.45 \text{ PM}$$

(iii) Embedded mode

$$E = 3.6(400)^{1.20} = 4772.81 \text{ PM}$$

$$D = 2.5(4772.8)^{0.32} = 38 \text{ PM}$$

COCOMO-Assignment

A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the effort, development time, average staff size and productivity of the project.