Software Engineering

SOFTWARE PROJECT PLANNING - I
COCOMO & PUTNAM

Gagan Deep
Founder & Director
Rozy Computech Services
Kurukshetra. rozygag@yahoo.com
www.rozyph.com

PLANNING A SOFTWARE PROJECT

- *Planning is perhaps the most important activity of management.*
- The **basic goal** of planning is to find the activity which are to be performed for completing a project.
- A **good plan** is that which can handle all the uncertain event which can occur during the development of project.
- A good planning helps in good decision making.
- **Lack of planning** is a primary cause of *schedule slippage*, *cost overruns*, *poor quality*, and *high maintenance costs* for software.

Some factors to consider in project planning:

- Cost & Schedule estimation techniques to be used; accuracy required.
- Life-cycle model, control functions, and reviews.
- Organizational structure (team structure)
- Level of formality in specifications, test plans, etc.
- Level of verification and validation (independent group?).
- Level of configuration management required.
- Level of quality assurance required.
- Planning for management of risk.
- Project monitoring & controlling techniques.
- Personnel recruitment and training.

SOFTWARE COST ESTIMATION

- Estimation of resources, cost and schedule for a software development effort requires experience.
- Access to good historical information and the courage to commit to quantitative measures when qualitative data are all that exist.
- The importance of software cost estimation is well documented.
- Good estimation techniques serve as a basis for communication between software personnel and non-software personnel such as managers, sales people or even customers

- Cost estimation is part of the planning stage of any engineering activity.
- The difference in cost estimation between software engineering and other disciplines is that **in software engineering** the primary cost is for people.
- In **other engineering disciplines**, the cost of materials - chips, bricks, or aluminum, depending on the activity - is a major component of the cost that must be estimated.
- In **software engineering**, to estimate the cost, we only have to estimate how many engineers are needed.

Uses of Cost Estimation

- Software cost estimation has two uses in software project management.
- During the planning stage, one needs to decide how many engineers are needed for the project and to develop a schedule.
- In monitoring the project's progress, one needs to assess whether the project is progressing according to schedule and take corrective action, if necessary.

Basili (1980) described four classes of resources models:

- Static single-variable models
- Static multi-variable models
- Dynamic multi-variable models
- Theoretical models

- The static single-variable model takes the form:

$$\text{Resources} = c_1 * (\text{Estimated Characteristics})^{C_2}$$

where the resources could be effort, project duration, staff size or requisite lines of software documentation. The constants c_1 and c_2 are derived from data collected from past projects. The basic version of the Constructive Cost Model or COCOMO is an example of a static single-variable model.

- Static multi-variable model has the following form:

  $$\text{Resources} = c_{21}e_1 + c_{22}e_2 + \ldots$$

  where e_1 is the ith software characteristics and c_{21} , c_{22} are empirically derived constants for the ith characteristics.
- A dynamic multivariate model projects resource requirements as a function of time.
- A theoretical approach to dynamic multivariable modeling hypothesizes a continuous resource expenditure curve and from it, derives equations that model the behavior of the resource. The Putnam Estimation Model is a theoretical dynamic multi-variable model.

COCOMO Model

- As we have already knows that COCOMO model is used for Cost estimation and developed by Boehm.
- This model also estimates the total effort in terms of person-months of the technical project staff.
- The **Effort Estimate** *includes development, management, and support tasks* but does *not include the cost of the secretarial and other staff that might be needed in an organization.*

The basic steps in this model are :

1. Obtain an initial estimate of the development effort from estimate of thousands of delivered lines of source code (KDLOC).
2. Determine a set of 15 multiplying factors from different attributes of the project.
3. Calculated the effort estimate by multiplying the initial estimate with all the multiplying factors i.e. multiply the values in step 1 and step 2.

- The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of size. To determine the initial effort E_i in person-months the equation used is of the type is shown below

$$E_i = a * (KDLOC)^b.$$

- The value of the constants a and b depend on the project type.
- In COCOMO, projects are categorized into three types –
  1. Organic
  2. Semidetached
  3. Embedded.

- ***Organic* projects** are in an area in which the organization has considerable experience and requirements are not difficult to find out. Such systems are usually developed by a small team. Example of this type of project are simple business systems, simple inventory management systems, and data processing systems.
- Projects of the **embedded type** are ambitious and novel; the organization has little experience and it is difficult to find out requirements. These systems have tight constraints from the environment (software, hardware, and people). Examples are embedded avionics systems and real-time command systems.

- The **semidetached systems** fall between these two types. Semidetached project in which analyst know not all but much of requirement and have little experience of developing. Examples of semidetached system include developing a new operating system (OS), a database management system (DBMS), and complex inventory management system.
- The constants a and b for different systems are given in table.
- There are 15 different attributes, called cost driver attributes, that determine the multiplying factors.

Table A : Constants for different project types

Project Type	Nominal effort Constant		Schedule Constant	
	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Table B : Effort multipliers for different cost drivers

Cost Drivers	Rating				
	Very Low	Low	Nominal	High	Very High
Product Attributes RELY, DATA, CPLX	Different Estimated Effort Multipliers for Different Attributes				
Computer attribute TIME, STOR VITR, TURN					
Personnel attribute ACAP, AEXP PCAP, VEXP, LAXP					
Project attribute MODP TOOL SCHED					

The Cost driver are divided into four categories: As specified in the last slide

- The value of each type of attribute varies from very low , low, nominal, high, very high as shown in table B.
- The multiplying factors for all 15 cost drivers are multiplied to get the **Effort Adjustment Factor** (EAF). The final effort estimate, E, is obtaining by multiplying the initial estimate by EAF :

$$E = EAF * E_i$$

- COCOMO provides three levels of models of increasing complexity :

  basic,

  intermediate, and

  detailed.

Average Duration Estimation

- Single variable models can be used to determine the overall duration of the project. Generally, schedule is modeled as depending on the Total Effort (which, in turn, depends on size). Again, the constants for the model are determined from historical data.
- **The IBM Federal System Division** found that the total duration, 'D 'in calendar months can be estimated by

$$D = 4.1 \times E^{.36}$$

- In COCOMO, the schedule is determined in a similar manner. The general equation for calculation of schedule is

$$D = c \times E^{d}$$

The value of c and d are shown in table A.

- The equation for an organic type of software is

$$D = 2.5 \times E^{.38}$$

- For other project types, the constants vary only slightly.

- Software cost estimation models such as COCOMO are required for an engineering approach to software management.
- Without such models, one has only judgment to rely on, which makes decisions hard to trust and justify.
- Worse, one can never be sure whether improvements are being made to the software.
- While current models still lack a full scientific justification, they can be used and validated against an organization's project data base.
- With the current state of the art of cost estimation modeling, it is not wise to have complete and blind trust in the results of the models, but a project manager would be equally unwise to ignore the value of such tools for a software development organization as a whole.

COCOMO II

Main objectives of COCOMO II:

- To develop a software cost and schedule estimation model tuned to the life cycle practices of the 1990's and 2000's
- To develop software cost database and tool support capabilities for continuous model improvement

Has three different models

- The Application Composition Model

  Good for projects built using rapid application development tools (GUI-builders etc)
- The Early Design Model

  This model can get rough estimates before the entire architecture has been decided
- The Post-Architecture Model

  Most detailed model, used after overall architecture has been decided on

COCOMO II Differences

- The exponent value b in the effort equation is replaced with a variable value based on five scale factors rather then constants
- Size of project can be listed as object points, function points or source lines of code (SLOC).
- EAF is calculated from seventeen cost drivers better suited for today's methods, COCOMO81 has fifteen
- A breakage rating has been added to address volatility of system

COCOMO II Calibration

- For COCOMO II results to be accurate the model must be calibrated
- Calibration requires that all cost driver parameters be adjusted
- Requires lots of data, usually more then one company has
- The plan was to release calibrations each year but so far only two calibrations have been done (COCOMO II.1997, COCOMO II. 1998)
- Users can submit data from their own projects to be used in future calibrations

Importance of Calibration

- Proper calibration is very important
- The original COCOMO II.1997 could estimate within 20% of the actual values 46% of the time. This was based on 83 data points.
- The recalibration for COCOMO II.1998 could estimate within 30% of the actual values 75% of the time. This was based on 161 data points.

Is COCOMO the Best?

- COCOMO is the most popular method however for any software cost estimation you should really use more then one method
- Best to use another method that differs significantly from COCOMO so your project is examined from more then one angle
- Even companies that sell COCOMO based products recommend using more then one method.

COCOMO Conclusions

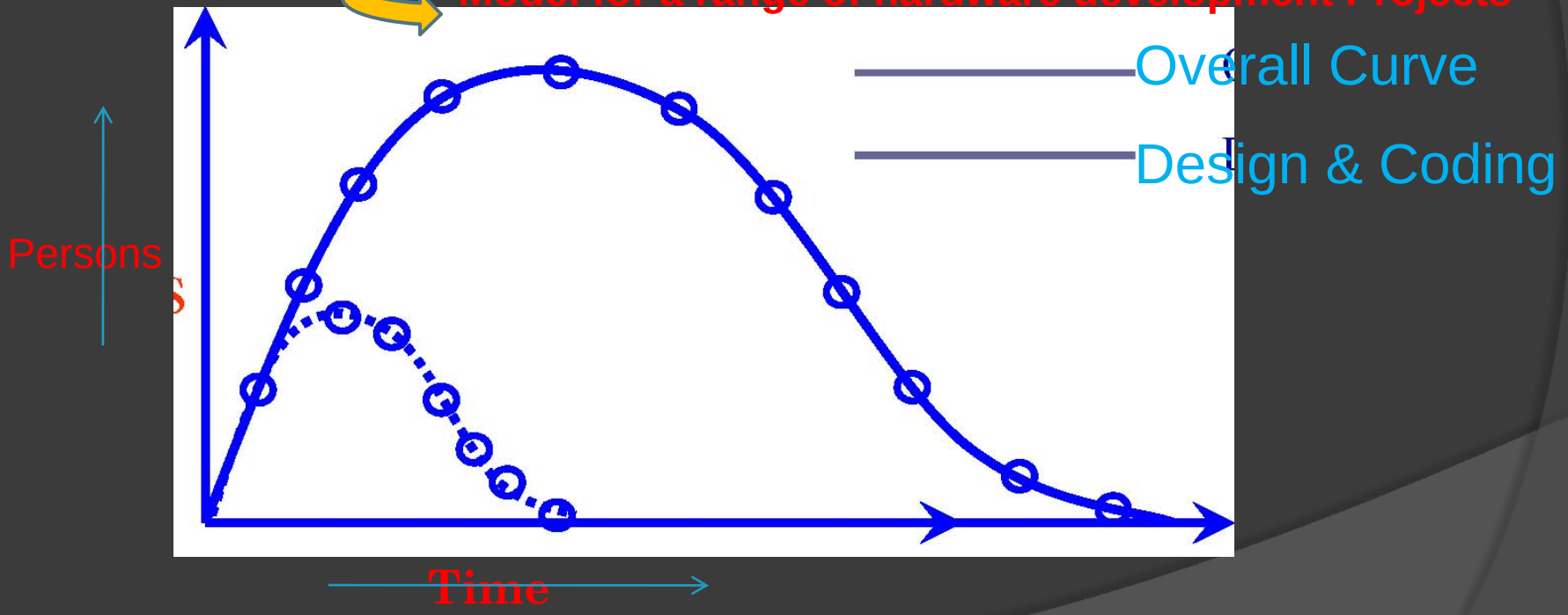
- COCOMO is the most popular software cost estimation method
- Easy to do, small estimates can be done by hand
- A free graphical version available for download
- Many different commercial version based on COCOMO – they supply support and more data, but at a price

Putnam Resource Allocation Model

Norden of IBM

Rayleigh Curve

Model for a range of hardware development Projects



- **Fig.6: The Rayleigh manpower loading curve**

Putnam Model Details

- Volume of work
- Difficulty gradient for measuring complexity
- Project technology factor measuring staff experience
- Delivery time constraints
- Staffing model based on
  - Total cumulative staff
  - How quickly new staff can be absorbed
  - # days in project month

Putnam Equation

$E = y(T) = 0.3945 * K$

K = area under curve [0 , 1)

measured in programmer year

T = optimal development time in years

$D = K / T^2$ *difficulty*

$P = c_i * D^{-2/3}$ *productivity*

$S = c * K^{-1/3} * T^{4/3}$ *lines of code*

Thanks!

Gagan Deep
Rozy Computech Services
3rd Gate, K.U., Kurukshetra
rozygag@yahoo.com