Project Estimation

Project Planning Objectives

- Providing a framework that allows managers to make responsible estimates of the resources and time required to build a software product.
- Determining the scope of a software project is the first project planning activity.
- Until the developer and customer agree on the scope of the project, it is impossible to determine what the project will cost and when the project will end.

Resources

- For software project, the resources used involve people, reusable software components, the development environment (hardware and software).
- Number of people required for software projects can only be determined after an estimate of development (e.g. man day) effort is computed.
- In modern software development, people and hardware may be shared among several projects.

Software Project Estimation

- Software is now the most costly element of virtually every computer system.
- Cost and effort estimates may determine whether an organization can realistically undertake the development of software product or not.
- It is important to get used to the idea or using two or more methods for making estimate and then using the results to cross check one another.

The Estimating Process

1. Establish the reason for the estimates
2. Obtain all the existing information
3. Establish the requirements
4. Calculate size, effort an timescales using appropriate method
5. Discuss and refine the estimate
6. Agree and document the estimate

The Estimate Report

- Background to the Estimates
- Statement of Effort, Costs and Times
- Proposed Schedule / Assumed Resources
- Final Products
- Assumption Made
- Risks to Meeting Estimates
- Recommendations

Estimation Techniques

- Past (similar) project experience
- Conventional estimation techniques
  - Task breakdown and effort estimates
  - Size (e.g. FP) estimates
- Tools (e.g. Checkpoint)

Cost Estimation

- Project scope must be explicitly defined
- Task and/or functional decomposition is necessary
- Historical measures (metrics) are very helpful
- At least two different techniques should be used
- Remember that uncertainty is inherent

Cost Estimation Elements

- **Effort**: Main cost, proportional to its size, usually measured as “lines of source code”
- **Plant**: Computer Equipment, Software Utilities
  - E.g. Editor, compilers, Word Processor Computer Aided Software Engineering tools (CASE).
- **Raw Materials**: Paper, Stationery, Diskettes etc.
- **Overheads**: Rent, Electrical Power etc.
- **External Costs**: Subcontracted outside.

Cost Estimation Consideration

- Life cycle model must be defined (include maintenance phase) and expenditure can then be estimated for each phase.
- It is important to cash flow calculation.
- Must make sure that Price must cover total cost plus profit.
- For “off the shelf” system, Price should multiplied by estimate volume of sales.

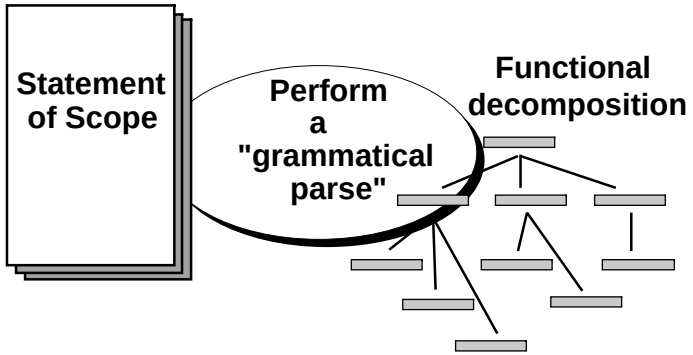
Schedule Estimation

- To determine that the deadline is feasible.
- The end of each phase will be marked by a ***Milestone*** which consist of the delivery of some ***Intermediate Product*** such as specification, design document, source code, etc.
- The total time of each phase is added to see whether the deadline is feasible.
- Usually more people is need in coding & testing than requirement specification and maintenance.

Schedule Estimation

- ***Gutness Estimation*** occurs when those making the estimates lack the courage to tell management they really believe the project will take.
- ***90% Syndrome*** refers to programmers when making informal estimates completion of work, say “It is 90% complete!”.
- Should include contingency time in schedule.

Functional Decomposition



Decomposition Techniques

- It will be easier to develop meaningful LOC or FP estimates to decompose the projects along the functional lines and then estimate the size of each sub function individually.
- This approach is called **Problem-based Estimation**.

Decomposition Techniques

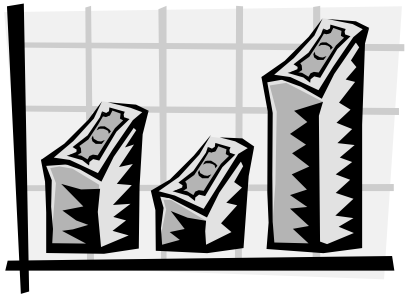
- Most people prefer process-based estimation since they are estimating the amount of time they plan spend on the tasks that make up each phase of their process model after they have determined the work products for each phase.
- It is important to point out that without historical data to give these estimates a context LOC and FP values may not be very useful for estimating cost or effort.

Aspects of Cost

- Cost of development
- Cost of marketing
- Cost of maintenance
- Cost of ownership (purchase, training, software unreliability)

Cost Classification (Size-based Estimation)

- Expert Judgement
- Analogy
- Parametric Cost Models



Cost Classification – Expert Judgment

- Experience + Past projects database
- Can use more than 1 expert

Cost Classification – Analogy

- Noting the similarities (and differences) between the current project and previous projects.
- Record of size, cost, development time are kept on paper or database – Corporate Memory
- When estimate the cost (effort & size) of a new project, manager should access to records of past projects, with all their essential parameters recorded in a quantitative way.

Cost Classification – Parametric Cost Models

- They yield estimates of total effort (in man-months) based on an estimate of the "size" (number of line of code) and relate the schedule to this by some formula.
- **Cost = Salaries + Overhead**

PERT (Program Evaluation and Review Technique)

- Three estimates
  - **s** – smallest
  - **m** – most likely
  - **l** – largest
- Estimation **E = (s + 4m + l) /6**
- Variance **V = (l – s) / 6**

Empirical Estimation Models

- A typical estimation model is derived using regression analysis on data collected from past software projects.
- The overall structure of such models is
  - **E = A + B * (ev)C**
- where ***A***, ***B*** & ***C*** are derived constants; ***E*** is effort in person-months and ***ev*** is the estimation variable (in KLOC or FP).
- In addition, most of them have some form of project adjustment component that enables E to be adjusted by other project characteristics such as complexity, staff experience and development environment.

Some Estimation Models

Model	Formula
Walston-Felix Model	$E = 5.2 * (KLOC)^{0.91}$
Bailey-Basili Model	$E = 5.5 + 0.73 * (KLOC)^{1.16}$
Boehm Simple Model	$E = 3.2 * (KLOC)^{1.05}$
Doty Model for KLOC > 9	$E = 5.288 * (KLOC)^{1.047}$
Albrecht & Gaffney Model	E = -13.39 + 0.0545 * FP
Kemerer Model	$E = 60.62 * 7.728 * 10^{-8} * FP^{3}$
Matson, Barnett & Mellichamp Model	E = 585.7 + 15.12 * FP

COCOMO

- The **CO**nstructive **CO**st **MO**del (COCOMO) is due to Barry Boehm. It has two fundamental equations to estimate effort and schedule.
- Boehm classifies software as
  - **Organic Software** is more-or-less freestanding, e.g. applications or system software.
  - **Embedded Software** is intimately bound up with a larger hardware / software system, e.g. missile control systems software.
  - **Semi-detached** is between the two.

COCOMO

- The model has three levels,
  - **Basic** – quick, easy, first approximation
  - **Intermediate** – 'effort multipliers', which are the product of 15 cost drivers
  - **Advanced** – includes still more cost drivers

Model Level	Parameters Use
Basic	Program Size
Intermediate	Program Size + Cost Driver (Subjective assessment of product, Hardware, Personnel, Project's attribute. There are totally 4 cost drivers with 15 attributes.)
Advanced	Program Size + Cost Driver + Cost Driver Impact on analysis, design etc.

COCOMO

- Parameters ***a*** and ***b*** have different values depending on the software class and model level.
- Parameters ***c*** and ***d*** depend on class only.
- Typical parameter values are:

	Basic		Intermediate			
	a	b	a	b	c	d
Organic	2.4	1.05	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.8	1.20	2.5	0.32

Basic COCOMO

- Effort in person-month **E = a x (KLOC)b**
- Development time in months **Dt = c x (E)d**

Intermediate COCOMO

- **E = a x (KLOC)b x EAF**
- where EAF (Effort Adjustment Factor) is based on the 4 Cost Drivers (15 attributes) Each attributes is rated on a 6-point scale from very-low to extra-high.
- These ratings allow an effort multiplier to be retrieved from tables.
- The product of all multipliers is the EAF with values ranging from 0.7 to 1.66, usually has a value from 0.9 to 1.4.

COCOMO

- The model also considers the distribution of effort, schedule and product activity over project phases, which requires additional variables to be estimated.
- The fundamental equations reduce the whole estimation problem to one of sizing, which is still very difficult.
- Boehm stresses that there is no royal road to sizing, just good corporate memory and PERT formula.

Criticism to COCOMO Models

- Basic COCOMO gives very crude estimates, and intermediate model does not provide any substantial improvement.
- COCOMO does not work well for small projects because small projects are more sensitive to variations and errors in parameters and cost factors.
- Too many parameters are required for Intermediate COCOMO. Also too subjective.

4 Cost Drivers (15 attributes)

- Product
  - Reliability
  - Database Size
  - Product Complexity
- Hardware
  - Execution Time Constraints
  - Storage Constraints
  - Volatility of the Development Machine
  - Turnaround Time of the Development Machine

4 Cost Drivers (15 attributes)

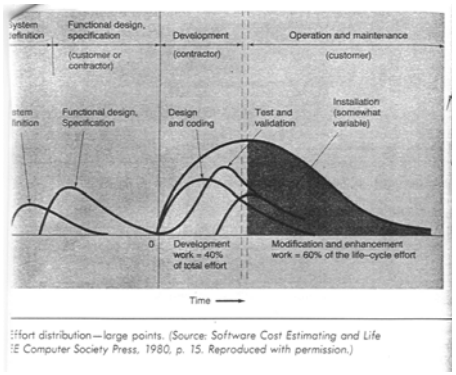
- Personnel
  - Analyst Capability
  - Software Engineer Capability
  - Application Experience
  - Development Machine Experience
  - Programming Language Experience
- Project
  - Use of Tools
  - Use of Software Engineering Methods
  - Development Schedule Constraint

Summary for 4 Cost Drivers

Cost Driver	Description	Rating					
		Very Low	Low	Nominal	High	Very High	Extra High
Product							
RELY	Required software reliability	0.75	0.88	1.00	1.15	1.40	-
DATA	Database size	-	0.94	1.00	1.08	1.16	-
CPLX	Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer							
TIME	Execution time constraint	-	-	1.00	1.11	1.30	1.66
STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.56
VIRT	Virtual machine volatility	-	0.87	1.00	1.15	1.30	-
TURN	Computer turnaround time	-	0.87	1.00	1.07	1.15	-
Personnel							
ACAP	Analyst capability	1.46	1.19	1.00	0.86	0.71	-
AEXP	Applications experience	1.29	1.13	1.00	0.91	0.82	-
PCAP	Programmer capability	1.42	1.17	1.00	0.86	0.70	-
VEXP	Virtual machine experience	1.21	1.10	1.00	0.90	-	-
LEXP	Language experience	1.14	1.07	1.00	0.95	-	-
Project							
MODP	Modern programming practices	1.24	1.10	1.00	0.91	0.82	-
TOOL	Software Tools	1.24	1.10	1.00	0.91	0.83	-
SCED	Development Schedule	1.23	1.08	1.00	1.04	1.10	-

Putnam Estimation Model

- A dynamic multivariable model that assumes a specific distribution of effort over the life of software development project.
- It uses Rayleigh curve to model distribution of effort over life cycle.



Effort distribution—large points. (Source: Software Cost Estimating and Life IEEE Computer Society Press, 1980, p. 15. Reproduced with permission.)

Putnam Estimation Model

- **L = CK (1/3) t (4/3)**
- where L is line of code, C is constant (2000 for poor software development environment, 8000 for good environment, 11000 for excellent environment), K is effort in person-years and t is development time in year

Putnam Estimation Model

- The area under the curve represents total development effort.
- Various curves
  - Requirement and specification
  - Design and code
  - Total Project

Criticism to Estimation Models

- An estimation model uses empirically derived formulas to predict data values necessary for project planning.
- Since the data from which the model is derived are collected from a limited sample of projects, no single estimation model is appropriate for all classes of software or in all development environments.

Criticism to Estimation Models

- The use of person-month as a unit of effort can be misleading.
- Models are calibrated on past data, parameters obtained for a particular environment will not work under different conditions.
- Use of LOC may not be an accurate measure.

Criticism to Putnam Models

- Rayleigh curve assumes development process is a Poisson process which is for independent events, but tasks in a project are not all independent.

Tool-Based Estimation

- Project characteristics
- Calibration factors
- LOC/FP data

Conventional Methods: LOC/FP Approach

- Compute LOC/FP using estimates of information domain values
- Use historical effort for the project

Example of LOC Approach

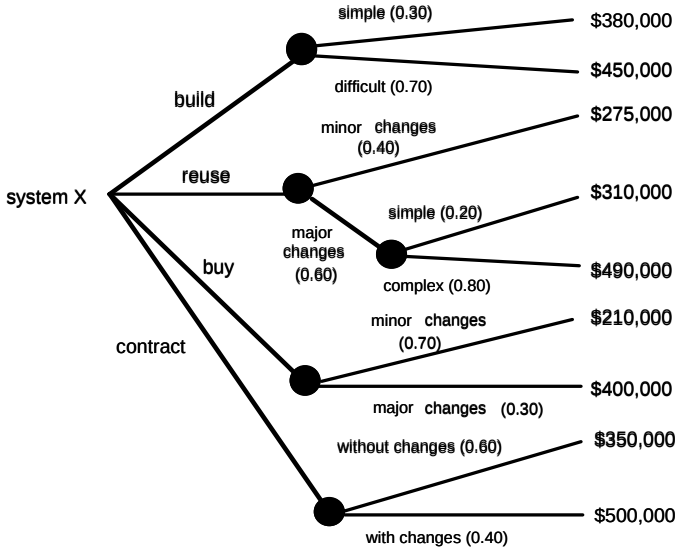
Functions	estimated LOC	LOC/pm	$/LOC	Cost	Effort (months)
UICF	2340	315	14	32,000	7.4
2DGA	5380	220	20	107,000	24.4
3DGA	6800	220	20	136,000	30.9
DSM	3350	240	18	60,000	13.9
CGDF	4950	200	22	109,000	24.7
PCF	2140	140	28	60,000	15.2
DAM	8400	300	18	151,000	28.0
Totals	33,360			655,000	145.0

Example of FP Approach

measurement parameter	count		weight		
number of user inputs	40	x	4	=	160
number of user outputs	25	x	5	=	125
number of user inquiries	12	x	4	=	48
number of files	4	x	7	=	28
number of ext.interfaces	4	x	7	=	28
algorithms	60	x	3	=	180
count-total →					569
complexity multiplier					.84
feature points →					478

× 0.25 p-m / FP = 120 p-m

The Make-Buy Decision



Computing Expected Cost

Expected cost =

$$\sum (\text{path probability})_i \times (\text{estimated path cost})_i$$

For example, the expected cost to build is:

$$\text{expected cost}_{build} = 0.30(\$380K)+0.70(\$450K) = \$429\ K$$

similarly,

$$\text{expected cost}_{reuse} = \$382K$$

$$\text{expected cost}_{buy} = \$267K$$

$$\text{expected cost}_{contr} = \$410K$$

Estimation Guidelines

- Establish a database of historical data on projects completed.
- Use data to calibrate models for the intended environment.
- Use several models (Estimate using at least two techniques)
- Get estimates from independent sources
- Avoid over-optimism, assume difficulties
- Trust none of them! (This means that, in practice results yielded by any model must be taken with care)

Managing Variation