Software Project Estimation

Eunseok Lee, Prof.
School of Software
Sungkyunkwan University

Objectives

- To introduce the concepts of software cost components
- To describe the techniques of software measurement
- To explain how to estimate the cost and effort
- To explain the basic concepts of COCOMO model

Topics covered

1. Software measurement
2. Options for reliable cost/effort estimation
3. Decomposition techniques
4. Empirical estimation models
5. COCOMO models

Cost estimation

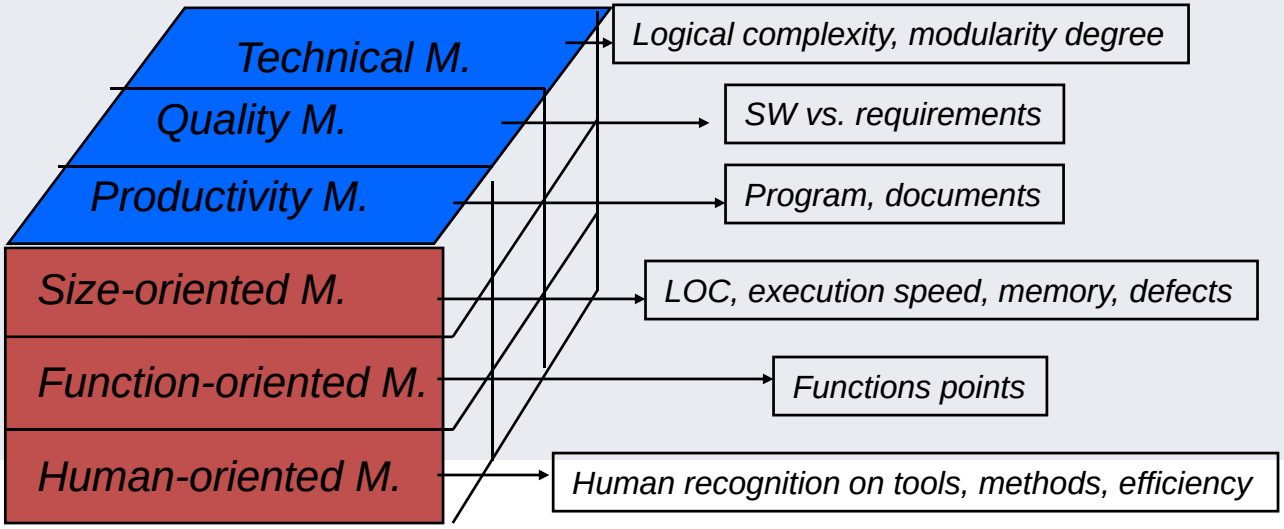
- Not exact science due to a lot of variables, human, technical, environment, political and so on..
- A large cost estimation error can make the difference between profit and loss
- Cost overrun can be disastrous for the developer
- Particularly important for today, as software is the most expensive element in most computer-based system

Project cost components

- Hardware and software costs.
- Travel and training costs.
- Effort costs (the dominant factor in most projects)
  - The 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.).

6-Key SW Technologies for Survival

- Structured methods (1st Gen.)
- CASE tools (2nd Gen.)
- Object-oriented methods (3rd Gen.)
- Software Quality Assurance
  : Structural, Functional, Validational
- Software Metrics
- Re-Engineering



1. Software Measurement

- **Direct measures (e.g., the line of code)**
- **Indirect measures (e.g., the productivity of a project)**

1. Size-oriented metrics

Project	LOC	Effort $(000)	pp. doc.	Errors	Defects	People
A	12,100	24	168	365	134	29	3
B	27,200	62	440	1224	321	86	5
C	20,200	43	314	1050	256	64	6
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-

- Normalization : Errors/KLOC, Errors/MM , etc.

Software Measurement (*Cont'*)

Controversy in size-oriented metrics

Proponents

1) LOC is an "artifact" of all software development project
2) Easy to count
3) Many existing software *estimation models* use (K)LOC as a key input
4) Large body of *literature* and *data* predicated on LOC already exists

Opponents

1) Dependent on programming languages and individual programming styles
2) Penalize well-designed but shorter programs
3) Hard to accommodate non-procedural languages
4) Difficult to estimate the cost and budget by LOC at an early stage of project due to the difficulty of estimation of LOC

Software Measurement (*Cont'*)

2. Function-oriented metrics (*or* FP metrics)

$$FP = count\_total \times [\, 0.65 + 0.01 \, \Sigma \, F_i \,]$$

Measurement parameter	count		Weighting Factor: simple		average		complex		
number of user inputs		×	3		4		6	=	
number of user outputs		×	4		5		7	=	
number of user inquires		×	3		4		6	=	
number of files		×	7		10		15	=	
number of external interfaces		×	5		7		10	=	
count_total	→								

Software Measurement (*Cont'*)

Fi : Complexity Adjustment Value

1. Does the system require reliable backup and recovery?
2. Are data communication required?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require on-line data entry?
7. Does the on-line data entry require the input transaction to be built over multiple screens or operational?
8. Are the master files updated on-line?
9. Are the inputs, outputs, files, or inquires complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and ease of use by the user?

No influence(0) Incidental(1) Moderate(2) Average(3) Significant(4) Essential(5)

Software Measurement (*Cont'*)

• Example of Function point based estimation : 인사정보시스템

어플리케이션명	단위업무 기능명	SW 기능					내부 파일명	외부 파일명
		추가	수정	삭제	출력	조회		
인사	직원정보관리	1	1	1	1	1	직원정보	
인사	직무정보관리	1	1	1	1	1	직무정보	
인사	보직관리	1	1	1	1	2	직원보직	
인사	근무지조회 및 출력				1	1		근무지정보
인사	직원면담기록	1	1	1		1	직원면담	
인사	직원기록대장관리	1			1		출력옵션	
인사	부양가족변동자료작성					1		
인사	고과대상자자동통보				1			
인사	화면필드헬프메세지	1				1		필드헬프
인사	직원자료변환(구->신)	1						
인사	진급대상자발췌						진급대상자	
인사	정현원통계				1		부서파일	
인사	부서별직원통계				1			
합계		7	4	4	8	8	7	2

Software Measurement (*Cont'*)

- **A Guideline for Complexity Identification**

구분		처리의 복잡도		
		쉽다	보통	어렵다
항목수	적다	단순	단순	보통
	보통	단순	보통	복잡
	많다	보통	복잡	복잡

(정통부 사업대가기준 개정안, 2002. 05)

Software Measurement (*Cont'*)

Controversy in function-oriented metrics

Proponents)

1) Programming language independent
2) Easy to estimate

Opponents)

1) Still requires some "sleight of hand" to estimate
2) Difficult to collect the data deliberately after the project completes
3) FP has no direct physical meaning

Software Measurement (*Cont'*)

Reconciling different metrics approaches

PL	LOC/FP(average)
Assembly language	320
C	128
Cobol	105
Fortran	105
Pascal	90
Ada	70
OOL	30
4GLs	20
Code generators	15
Spreadsheets	6
Graphical languages	4

2. Options for reliable cost / effort estimation

1. Delay estimation until in the late project

   ⇒ Estimation should be conducted in the early stage

2. Perform the estimation based on similar projects that have already been completed

   ⇒ Other project influences, customer, business condition, constraints

3. Use relatively simple "*decomposition techniques*" to estimate project cost and effort

   ⇒ By decomposing a project into major functions and related SE activities, cost/effort estimation can be performed in a stepwise fashion

4. Use one or more empirical models for software cost and effort estimation

   ⇒ used to complement decomposition techniques

   ⇒ $d = f(v_i)$ ex) cost = f (LOC)

3. Decomposition Techniques

Accuracy factors for S/W project estimation

1) the degree to which a developer has appropriately estimated the size of the product to be built
2) the ability to translate the estimated size into human effort, duration and cost
3) the degree of reflecting the abilities of the S/W team
4) the stability of product requirements and the support environment

Problem-based estimation

1) Preliminary statement of software scope
2) Decompose software into problem functions
3) Estimation of LOC or FP for each function
4) Combine the results to produce an overall estimate for the entire project
5) Estimate the effort, cost, time

(1) Application example of PBE

i) **Preliminary statement of software scope**
ii) Decompose software into problem functions
iii) Estimation of LOC or FP for each function
iv) Combine the results to produce an overall estimate for the entire project
v) Estimate the effort, cost, time

The CAD software will accept two-and three- dimensional geometric data from an engineer. The engineer will interact and control the CAD system through a user interface that will exhibit characteristics of good human-machine interface design. All geometric data and other supporting information will be maintained in a CAD database. Design analysis modules will be developed to produce required output which will be displayed on a variety of graphics devices. The software will be designed to control and interact with peripheral devices that include a mouse, digitizer, and laser printer

(2) Application example of PBE

i) Preliminary statement of software scope
ii) Decompose software into problem functions
iii) Estimation of LOC or FP for each function
iv) Combine the results to produce an overall estimate for the entire project
v) Estimate the effort, cost, time

- Decompose (refine) problem (major) functions
  - User interface and control facilities (UICF)
  - 2-dimensional geometric analysis (2DGA)
  - 3-dimensional geometric analysis (3DGA)
  - Database management (DBM)
  - Computer graphics display facilities (CGDF)
  - Peripheral control (PC)
  - Design analysis modules (DAM)

(3) Application example of PBE

Case of 3-dimensional geometric analysis(3DGA)

i) Establishes the range of LOC estimates for 3DGA
- optimistic: 4600
- most likely : 6900
- pessimistic : 8600

ii) Calculates the expected value for the 3DGA
- $EV=(S_{opt} + 4S_m + S_{pess}) / 6$
  = 6,800 LOC

iii) Combine the results : 33,200 LOC

iv) Estimate the effort, cost from historical data,
- productivity of systems of this type : 620LOC/pm
- burdened labor rate : $8000/M -> cost/LOC = $13/LOC

Total estimated cost = $431,000

Total estimated effort = 54 PMs

i) Preliminary statement of software scope
ii) Decompose software into problem functions
iii) Estimation of LOC or FP for each function
iv) Combine the results to produce an overall estimate for the entire project
v) Estimate the effort, cost, time

in a same way →

UICF	2,300
2DGA	5,300
3DGA	6,800
DBM	3,350
CGDF	4,950
PC	2,100
DAM	8,400

Productivity estimates

- Real-time embedded systems, 40-160 LOC/P-month.
- Systems programs , 150-400 LOC/P-month.
- Commercial applications, 200-900 LOC/P-month.
- In object points, productivity has been measured between 4 and 50 object points/month depending on tool support and developer capability.

(4) Application example of PBE

- $FP_{estimated} = count\_total \times [0.65 + 0.01 \times \Sigma F_i]$

i) Preliminary statement of software scope
ii) Decompose software into problem functions
iii) Estimation of LOC or FP for each function
iv) Combine the results to produce an overall estimate for the entire project
v) Estimate the effort, cost, time

Information dom. value	Opt	likely	Pess	Est. count	weight	FP. count
number of user inputs	20	24	30	24	4	96
number of user outputs	12	15	22	16	5	80
number of user inquires	16	22	28	22	4	88
number of files	4	4	5	4	10	40
number of external interfaces	2	2	3	2	7	14
count_total						318

$\Sigma F_i = 52$: complexity adjustment factors

$FP_{estimated} = 372$

Complexity Adjustment Factors

- F_i

1. Does the system require reliable backup and recovery? 4
2. Are data communication required? 2
3. Are there distributed processing functions? 0
4. Is performance critical? 4
5. Will the system run in an existing, heavily utilized operational environment? 3
6. Does the system require on-line data entry? 4
7. Does the on-line data entry require the input transaction to be built over multiple screens or operational? 5
8. Are the master files updated on-line? 3
9. Are the inputs, outputs, files,or inquires complex? 5
10. Is the internal processing complex? 5
11. Is the code designed to be reusable? 4
12. Are conversion and installation included in the design? 3
13. Is the system designed for multiple installations in different organizations? 5
14. Is the application designed to facilitate change and ease of use by the user? 5

No influence(0) Incidental(1) Moderate(2) Average(3) Significant(4) Essential(5)

4. Empirical Estimation Models

- The empirical data that supports most estimation models is derived from a limited sample of project
- No estimation model is appropriate for all classes of software and in all development environments

- The structure of estimation models

  $E = A + B \times (ev)^C$ A,B,C : empirically derived constants

  E : effort (person-months)

  ev : estimation value(LOC, FP)

- $E = 5.2 \times (KLOC)^{0.91}$: Walston –Felix model
- $E = 5.5 + 0.73 \times (KLOC)^{1.16}$: Bailey-Basili model
- $E = 3.2 \times (KLOC)^{1.05}$: Boehm simple model
- $E = 5.288 \times (KLOC)^{1.047}$: Doty model
- $E = -13.39 + 0.0545\ FP$: Albrecht and Gaffney model
- $E = 60.62 \times 7.728 \times 10^{-8}\ FP^3$: Kemerer model
- $E = 585.7 + 15.12\ FP$: Matson, Barnett and Mellichamp model

5. The COCOMO Models [Barry Boehm,81]

- COnstructive COst MOdel
- Hierarchical SW estimation model

(1) Basic COCOMO model
: Computes SW development effort(cost) as a function of program size

: $E = a_b (KLOC)^{bb}$ E : effort (person-months)

: $D = c_b (E)^{db}$ D : duration for project

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

3-Classes of SW projects

i) Organic mode
: small, simple software projects
: small teams with good applications experiences
: no rigid requirement

ii) Semi-detached mode
: intermediate projects
: mixed experiences team
: a mix of rigid and less than rigid requirement

iii) embedded mode
: software project that must be developed within a set of tight hardware, software and operational constraints
(e.g., flight control SW for aircraft)

3-Classes of SW projects

Case 1)
- Basic COCOMO,
- Organic mode SW project,
- expected LOC: 32,000

- Effort= $2.4(32)^{1.05}$ = 91 PM
- Duration = $2.5(91)^{0.38}$ = 14 Months
- Productivity = 32,000/91 = 352 LOC
- FSP = 91/14 = 6.5 (fulltime software personnel)

* A person can generate 16 instructions a day.

Can the COCOMO apply to every type of project situation?

Intermediate COCOMO Models

(2) Intermediate COCOMO model
: extend the basic with a set of " cost driver attributes (CDA)"

: $E = a_i (KLOC)^{bi} \times EAF$, EAF : effort adjustment factor (0.9 ~1.4)
: product of all *effort multiplier*

- 4 major category of CDA
  - product attributes(3) - SW신뢰도/ DB크기/ 제품의 복잡도
  - computer attributes(4) – SW 생산성에 영향을 주는 HW의 제한조건 수행시간제한/ 기억장소제한/ VM의 안정성/ T/A시간
  - personnel attributes(5) – 분석가의 능력/ 개발분야의 경험/ VM경험/ 프로그래머의 경험/ 프로그래밍 언어의 경험
  - project attributes(3) – 최신프로그래밍 기법의 이용/ SW도구의 활용 가능성/ 요구되는 개발일정
- 15 attributes for each category
- 6-point scale rate for each attribute (very low ~ extra high)
- *Effort multiplier* is determined from tables by Boehm for each point

Intermediate COCOMO Models

	very low	low	average	high	very high	extra high
RELY	0.75	0.88	1	1.15	1.4	-
DATA	-	0.94	1	1.08	1.16	-
CPLX	0.7	0.85	1	1.15	1.3	1.65
TIME	-	-	1	1.11	1.3	1.65
STOR	-	-	1	1.06	1.21	1.56
VIRT	-	0.87	1	1.15	1.3	-
TURN	-	0.87	1	1.07	1.15	-
ACAP	1.46	1.19	1	0.86	0.71	-
AEXP	1.29	1.13	1	0.91	0.82	-
PCAP	1.42	1.17	1	0.86	0.7	-
VEXP	1.21	1.1	1	0.9	-	-
LEXP	1.14	1.07	1	0.95	-	-
MODP	1.24	1.1	1	0.91	0.82	-
TOOL	1.24	1.1	1	0.91	0.82	-
SCED	1.23	1.08	1	1.04	1.1	-

Object points

- Object points (alternatively named application points) are an alternative function-related measure to function points *when 4GLs or similar languages are used for development*.
- 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;

    (1op/simple, 2op/moderate, 3op/complex)
  - The number of reports that are produced by the system;

    (2op/simple, 5op/moderate, 8op/complex)
  - The number of program modules that must be developed to supplement the database code; (10op/module)

Object point estimation

- Object points are easier to estimate from a specification than function points as they are simply concerned with screens, reports and programming language modules.
- They can therefore be estimated at a fairly early point in the development process.
- At this stage, it is very difficult to estimate the number of lines of code in a system.