100만원이 필요하다고 말했는데
200이필요하면 〉〉 끝내지를 못하고
50이 필요하면 〉〉 프로젝트를 못딸수있음

Unit 10.
Estimation Techniques

PM이 해야하는 일 중 하나!

추정해야한다
〉돈
〉기간
〉인원
을 정확하게 추정해야함!

Contents

- Project Planning Process
- Software Project Estimation
- Decomposition Technique
- Empirical Estimation Model

Project Planning Process

- Establish project scope
- Determine feasibility
- Analyze risks
- Define required resources
  - Determine human resources required
  - Define reusable software resources
  - Identify environmental resources

Project Planning Process

- **Estimate cost and effort**
  - Decompose the problem
  - Develop two or more estimates using size, function points, process tasks, or use cases
  - Reconcile the estimates

man-month
person-month
〉〉한사람이 한달동안 할 수 있는 일의양
(여기서는 effort의 기준(유닛)으로 사용됨)

- **Develop a project schedule**
  - Establish a meaningful task set
  - Define a task network
  - Use scheduling tools to develop a timeline chart
  - Define schedule tracking mechanisms

Project Estimation

- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent in the process

Software Project Estimation

- **Decomposition Technique**
  - Decompose a project into major functions and related software engineering activities.
  - Perform cost and effort estimation on each.
  - Use historical effort for the project

경험적인 = = ~ 통계적인

- **Empirical Estimation Model**
  - Can be used to complement decomposition techniques.
  - Based on experience (historical data).

이미 다른사람이 만들어 놓은 공식, 행렬등을 이용해 간단하게 구할 수 있는 경우!

Process-Based Estimation Example

Activity →	CC	Planning	Risk Analysis	Engineering		Construction Release		CE	Totals
Task →				analysis	design	code	test		
Function									
▼									
UICF				0.50	2.50	0.40	5.00	n/a	8.40
2DGA				0.75	4.00	0.60	2.00	n/a	7.35
3DGA				0.50	4.00	1.00	3.00	n/a	8.50
CGDF				0.50	3.00	1.00	1.50	n/a	6.00
DSM				0.50	3.00	0.75	1.50	n/a	5.75
PCF				0.25	2.00	0.50	1.50	n/a	4.25
DAM				0.50	2.00	0.50	2.00	n/a	5.00
Totals	0.25	0.25	0.25	3.50	20.50	4.50	16.50		46.00
% effort	1%	1%	1%	8%	45%	10%	36%		

CC = customer communication CE = customer evaluation

Based on an average burdened labor rate of $8,000 per month, the total estimated project cost is $368,000 and the estimated effort is 46 person-months.

cost & effort를 추정하기 위해 사용하는 기법

Decomposition Techniques

Lines of code

function point
= = 기능함수

언어마다 같은기능을 수행하도록 만들어도 나오는 라인수가 다르니까 (자바의 경우 c의 3분의1)
》요즘은 fp를 유닛으로 더 많이 사용한다.

- **LOC and FP data are used**

이 두가지를 유닛으로 사용한다

  - as an estimation variable of 'size', and
  - as baseline metrics collected from past projects.

- **Steps**

1. Decompose the software scope into problem functions that can each be estimated individually.

EV를 쓰는이유 》most likely를 쓰는 것 보다 정확도가 높음

2. Estimate LOC or FP for each function.
   - Compute a *three-point* or *expected value(EV)*, A weighted average of the *optimistic, most likely,* and *pessimistic*.
   - $EV = (S_{opt} + 4S_m + S_{pess}) / 6$

expected value(기대치)

3가지 경우의 평균을 구하게 됨
most likely 》 거의 일반적인경우
optimistic 》 이론적 베스트케이스
pessimistic 》 이론적 워스트케이스
so 시간적으로 봤을때
opt 〈 most likely 〈 pess

3. Apply baseline productivity metrics to derive cost or effort each function.

   LOC / Person-Month, FP / Person-Month

LOC-Based Estimation

- **Step 1. Decompose.**
  - CAD Application

Function	
User Interface and Control Facilities	
Two-dimensional Geometric Analysis	
Three-dimensional Geometric Analysis	
Database Management	
Computer Graphics Display Facilities	
Peripheral Control	
Design and Analysis Modules	
Estimated Lines of Code	

LOC-Based Estimation

- **Step 2. Estimate LOC or FP for each function.**
  - $EV = (S_{opt} + 4S_m + S_{pess}) / 6$
    $= (4600 + 4*6900 + 8600) / 6 = 6,800$

Function	Estimated LOC
User Interface and Control Facilities	2,300
Two-dimensional Geometric Analysis	5,300
Three-dimensional Geometric Analysis	6,800
Database Management	3,350
Computer Graphics Display Facilities	4,950
Peripheral Control	2,100
Design and Analysis Modules	8,400
Estimated Lines of Code	33,200

각각의 값들이 모두
이러한 방식으로 구해진것임

LOC-Based Estimation

- **Estimation of LOC**
  - 33,200 Lines of Codes

하루에 20줄씩 짠다고 가정하는 것이다.
이는 전체 개발기간동안 작성한 코드로 보기때문에
디자인, 분석등을 할때는 코드를쓰지 않아서
값이 적어보이는 것이다.

- **Historical Data**
  - Productivity: 620 LOC / Person-Month
  - Cost per LOC: $13 / LOC
    - Based on the Burdened Labor Rate: $8,000 / Person-Month
    - $8000 / 620 = $13 /LOC

한사람이 받는 돈이아니라,
임대료,전기세,보험료,식대비등등
모든 비용이 포함 된 금액

여기에 있는
cost, effort는 모두
추정일뿐 실제가 아니라는 것을
기억해야한다!
그래도 최대한 근사치를 추정 해야함

- **Estimation of Project Cost**
  - 33,200 LOC * $13 = $431,600

실제로 54명을 고용한다고 해서
한달만에 일을 끝낼 순 없다
>>사람이 많아지면 의사소통오버헤드가
많아지기 때문에

- **Estimation of Effort**
  - 33,200 / 620 = 54 Person-Months

= =기능복잡도를 나타냄(complexcity of the functionallity)

FP-Based Estimation

- Estimate inputs, outputs, inquiries(질의), files and external interfaces.
  - 가지고 있는 DB수 (멤버, 상품, 장바구니, 등등)
  - 카드결제시스템, 무통장입금시스템 등등
  - Use Expected Value (EV) formula.

Function	Opt.	Most	Pess.	Est.Count
# of Inputs	20	24	30	24
# of Outputs	12	15	22	16
# of Inquires	16	22	28	22
# of Files	4	4	5	4
# of External Interfaces	2	2	3	2
Count_Total				~~318~~

잘못나옴

FP-Based Estimation

- **Assume that the complexity weighting factor is average.**

복잡도 가중치

Parameter	Count	Simple	Average	Complex	
# of User Inputs	[24] x	3	4	6	= [96]
# of User Outputs	[16] x	4	5	7	= [80]
# of User Inquiries	[22] x	3	4	6	= [88]
# of Files	[4] x	7	10	15	= [40]
# of Ext. Interfaces	[2] x	5	7	10	= [14]

= = number

가중치는 고정된 값이지만
인풋들이 어떻게 분포 할 것인지는 추정으로 정할 수 있다
〉〉위에서는 모두average로 넣었지만
실제경우에서는
# of user inputs에시를 들면
simple 5개, average 10개, complex 9개 등으로하여
5*3 + 10 * 4 + 9 * 6 = 109로 계산 할 수도 있다

Count_Total 318

FP-Based Estimation

복잡도 조정인자

점수가 높으면 높을수록 복잡한 프로그램이다.
테스트를 해보고
점수를 보고 그 프로그램이 얼마나 복잡한지 확인가능

- **Complexity Adjustment Factors**
  - Backup and Recovery 4
  - Data Communications 2
  - Distributed Processing 0
  - Performance Critical 4
  - Existing Operating Environment 3
  - Online Data Entry 4
  - Input Transaction over Multiple Screens 5
  - Master Files updated online 3
  - Information Domain Values complex 5
  - Internal Processing Complex 5
  - Code Designed for Reuse 4
  - Conversion/Installation in Design 3
  - Multiple Installations 5
  - Applications Designed for Change 5

FP-Based Estimation

- **Estimation of FP**
  - FP = Count_Total x [0.65 + 0.01 x sum(F_i)], i = 1 to 14
  - FP = 318 x [0.65 + 0.01 x 52] = 318 x 1.17 = **372**

위에 복잡도 조정인자의 합

기존 값 보다 증가 했다
〉〉 내 프로젝트가 복잡하다

- **Historical Data**
  - Productivity: 6.5 FP / Person-Month
  - Cost per FP: $1,230 / FP
    - Based on the Burdened Labor Rate: $8,000 / Person-Month
    - $8000 / 6.5 = $1,230 / FP

기존의 count total을 기준으로 했을때
위의 과정을 거쳐서
값이 증가한다면 프로젝트가 어느정도 복잡한것
값이 감소한다면 프로젝트가 어느정도 심플한것

- **Estimation of Project Cost**
  - 372 FP * $1,230 = $ 457,560
- **Estimation of Effort**
  - 372 FP / 6.5 = 57.2 Person-Months

경험적인 Empirical Estimation Models

- **Uses empirically derived formulas to estimate.**
  - A typical estimation model is derived using regression analysis on data collected from past software projects.
  - $E = A + B \times (ev)^C$
  - where A, B, and C are empirically derived constants, *E* is effort in person months, and *ev* is the estimation variable (either LOC or FP).
  - Also uses project adjustment components.
  - Estimation models must be calibrated for local needs.

COCOMO Model

- COnstructive COst MOdel by Barry Boehm
- Model 1. Basic COCOMO ——— 우리는 이것만 배움
  - computes the effort as a function of program size (LOC).
- Model 2. Intermediate COCOMO
  - computes the effort as a function of program size and a set of 'cost drivers' that include subjective assessments of product, hardware, personnel, and project attributes.
- Model 3. Advanced COCOMO
  - On top of model 2, it includes an assessment of the cost driver's impact on each step (analysis, design, etc) of software engineering process.

Basic COCOMO

- **Three Classes of Software Projects**
  - **Organic Class**
    - Small product size (less than 50,000 LOC)
    - Small, In-house development team
    - Development team experienced in the application area
    - Relaxed (negotiable, informal) specifications on function and performance requirements, acceptance tests, and interfaces
    - Minimal communication overhead
    - Stable development environment
    - Minimal schedule pressure
    - Existing, proven technology used

내가 lg에서 일하면
>>개발한 프로그램이 lg에
의해서 사용되는것을 의미

반대되는 의미는
contract-base
>내가 만들어서
다른컴퍼니가 사용하는것
이 있다

Basic COCOMO

- **Semi-Detached Class**
  - **Intermediate size (up to 300,000 LOC)**
  - **Team personnel is mix of experienced and inexperienced in the application area and development environment.**
  - **Mix of relaxed and rigid specification**
  - **Moderate schedule pressure**

Basic COCOMO

- **Embedded Class**
  - Size may range fro4m 20,000 to over 1,000,000 LOC.
  - Rigorous specification of function, performance, etc.
  - Product must operate within time constraints on internal and external interface service, processing, and interrupt service.
    ex) Flight Control Software for Aircraft
  - Product must meet rigid, formal quality standards.
  - Close coupling among hardware, software and operators
  - Extensive testing required.
  - Leading edge technology employed.
  - Other system components developed concurrently with software
  - Strong schedule pressure

Basic COCOMO

- **Equations** 1000LOC = = 1KLOC
  - Effort, $E = a * KLOC^{b}$
  - Devel. Time, $D = c * E^{d}$
  - *KLOC* is the estimated LOC for the project.
  - Coefficients Table

Project	a	b	c	d
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

위에 설명한 3가지 클래스중 하나라고 결정하고 나면 〉〉이쪽에 있는 matrix사용

  - Number of People, N = E / D

Basic COCOMO

- **Applied to CAD Software**
  - Project Class: *Organic*
  - Effort, $E = 2.4 * KLOC^{1.05} = 2.4 * (33.2)^{1.05}$
    $= 95$ Person-months
    - Higher than 54/58 Person-Months
    - COCOMO model assumes considerably lower LOC/pm levels.
    - The model can be recalibrated for the local environment.
  - Dev. Time, $D = c * E^{d} = 2.5 * E^{0.38} = 2.5 * 95^{0.38}$
    $= 14.1$ months
    - The *D* value enables the planner to determine a recommended number of people, N.
  - Number of People, N = E / D = 95 / 14.1 == 7 people
    - The planner may decide to extend the project duration with less number of people.

COCOMO의 경우
》전세계 평균을 이용하기때문에
개발자의 평균생산성을 낮게 잡는다
》따라서 간단한 작업의 경우도
사람이 많이 필요한 것으로 나온다

수업은 여기까지 함

COCOMO-II

- **COCOMO II is actually a hierarchy of estimation models that address the following areas:**
  - **Application composition model**
    - **Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.**
  - **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.**

The Software Equation

A dynamic multivariable model

$$E = [LOC \times B^{0.333}/P]^3 \times (1/t^4)$$

where

E = effort in person-months or person-years

t = project duration in months or years

B = "special skills factor"

P = "productivity parameter"

Estimation for OO Projects-I

- Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
- Using object-oriented requirements modeling (Chapter 6), develop use-cases and determine a count.
- From the analysis model, determine the number of key classes (called analysis classes in Chapter 6).
- Categorize the type of interface for the application and develop a multiplier for support classes:

Interface type	Multiplier
No GUI	2.0
Text-based user interface	2.25
GUI	2.5
Complex GUI	3.0

Estimation for OO Projects-II

- Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.
- Multiply the total number of classes (key + support) by the average number of work-units per class. Lorenz and Kidd suggest 15 to 20 person-days per class.
- Cross check the class-based estimate by multiplying the average number of work-units per use-case