Unit-1 & 5
Software Estimation

Chamundeswari Arumugam
Professor
SSN College of Engineering, Chennai

January 2018

Outline

- Line of Code
- Function point
- COCOMO
- Make/Buy decision

Line of Code

- Measurements - direct measure (eg. LOC, memory size, defects) and indirect measure(eg. efficiency, reliability)
- Project - 12,100 LOC were developed with 24 pm of effort at a cost of $\$168,000$.
- Effort and cost include all software engineering activities (analysis, design, code, and test)
- LOC measure is used to derive productivity metrics
- A set of simple size-oriented metrics
  1. Errors per KLOC (thousand lines of code)
  2. Defects per KLOC
  3. $ per KLOC
  4. Pages of documentation per KLOC
  5. Errors per person-month
  6. KLOC per person-month
  7. $ per page of documentation

Line of Code (contd..)

- Software development projects can be easily counted
- Estimation models use LOC or KLOC
- Literature and data predicated on LOC already exists
- Planner estimate the LOC to be produced long before analysis and design
- Disadvantages
  1. Programming language dependent
  2. Can't count nonprocedural language

Line of Code - An example

Case study

- The software is to execute on an engineering workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high-resolution color display, and laser printer.

1. organizational average productivity=620 LOC/pm.
2. labor rate=$8000 per month, cost per LOC=$13
3. Total effort required to develop the software =(33200 LOC) /(620 LOC/person month) = 54 person month
4. Total project cost to develop the software =54*$8000=$431, 000

Function	**Estimated LOC**
User interface and control facilities (UICF)	2,300
Two-dimensional geometric analysis (2DGA)	5,300
Three-dimensional geometric analysis (3DGA)	6,800
Database management (DBM)	3,350
Computer graphics display facilities (CGDF)	4,950
Peripheral control function (PCF)	2,100
Design analysis modules (DAM)	8,400
Estimated lines of code	*33,200*

Function Point

- FP measures is used to derive productivity metrics
- Programming language independent
- Ideal for conventional and nonprocedural languages
- Based on data that are more likely to be known early in the evolution of a project
- Rough estimates of the average number of LOC to build one function point in various programming languages is available.
- Avg LOC/FP, for C++=66, java=63, perl=60
- Disadvantages
  1. Computation is subjective
  2. Collection of data

Function Point

- Information domain values are defined are as follows.
  1. Number of external inputs (EIs) - Originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs).
  2. Number of external outputs (EOs)- Derived data within the application that provides information to the user. In this context EO refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.
  3. Number of external inquiries (EQs) - An online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF).
  4. Number of internal logical files (ILFs). Logical grouping of data that resides within the applications boundary and is maintained via external inputs.
  5. Number of external interface files (EIFs) - logical grouping of data that resides external to the application but provides information that may be of use to the application.

- FP=count total x $[0.65 + 0.01 \times \sum_{i=1}^{n}(F_i)]$

Figure: Function point

Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)		×	3	4	6	=	
External Outputs (EOs)		×	4	5	7	=	
External Inquiries (EQs)		×	3	4	6	=	
Internal Logical Files (ILFs)		×	7	10	15	=	
External Interface Files (EIFs)		×	5	7	10	=	
Count total →							

Function Point(contd..)

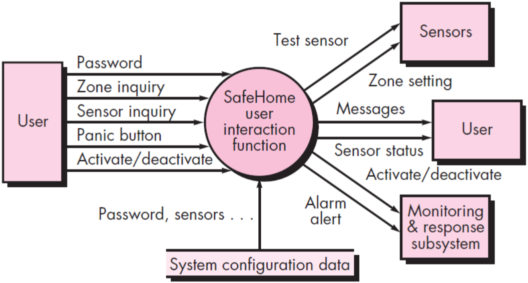


Figure: FP case study -I

- EI=password, panic button, and activate/deactivate
- EO=messages, sensor status
- EQ=zone inquiry, sensor inquiry
- ILF=system configuration file
- EIF=test sensor,zone setting,activate/deactivate,alarm alert

Function Point(contd..)

Figure: FP case study -I

Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)	3	×	(3)	4	6	=	9
External Outputs (EOs)	2	×	(4)	5	7	=	8
External Inquiries (EQs)	2	×	(3)	4	6	=	6
Internal Logical Files (ILFs)	1	×	(7)	10	15	=	7
External Interface Files (EIFs)	4	×	(5)	7	10	=	20
Count total →							50

Function Point(contd..)

- 14 Value adjustment factors= $\sum_{i=1}^{n=14}(F_i)$=46
- FP = 50 x [0.65 + (0.01 x 46)]= 56 FP
- One FP (conversion based on organization) = 60 Lines Of Code(LOC) per FP
- Total LOC of the software = 56 FP x 60 LOC per FP =3360 LOC
- Organizational effort = 12 FP person-month
- Total effort required to develop software = 56 FP /12 FP person-month = 5 person-month

Function Point(contd..) - case study II

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	110
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						342

Value adjustment factor	Value
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
Application designed for change	5
	52

1. Total FP = 342 x (0.65 + 0.01 x 52) = 342 x 1.17 = 400 FP
2. Org avg productivity = 6.5 FP/person month & labor rate =$8000 per month
3. Org avg cost per FP = $1230
4. Effort= 400 FP / 6.5 FP per person-month = 62 person-month
5. cost=62 * $8000 = $496,000

COCOMO II steps

- Original COCOMO - Barry Boehm, Evolve - COCOMO II

COCOMO hierarchy of estimation models

- Application composition model - Used during the early stages of software engineering,
- Early design stage model - when requirements have been stabilized and basic software architecture has been established.
- Post-architecture-stage model - Used during the construction of the software

- Sizing options - LOC, FP, Object point

COCOMO II steps (contd..)

- Component-based development or reuse is applied for New object point, then %reuse is estimated.
- PROD depends productivity rate of developer experience and development environment maturity

COCOMO hierarchy of estimation models

- NOP= (object points) x [(100 - % reuse) / 100]
- Estimate of project effort = NOP / PROD

COCOMO II steps (contd..)

Figure: Object point estimation

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

Figure: Productivity rate

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity/capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

COCOMO II steps - case study (contd..)

- IIST Airline sales system - A booking screen to record a new advertising sale booking,a pricing screen showing the advertising rate for each day and each flight, an availability screen showing which flights are available, a sales report showing total sales for the month and year, and comparing them with previous months and years.

Case study - Given data

- Screens = 3(simple, simple, medium), report =1(medium)
- Developer experience is very low (4) and the CASE tool is low (7).

Case study - COCOMO II Solution

- Object point=3x1+3x1+3x2+1x5 = 17
- NOP=17x[(100-0)] /100=17
- PROD =(4+7)/2=5.5
- Effort = NOP/ PROD = 17/5.5 = 3 pm

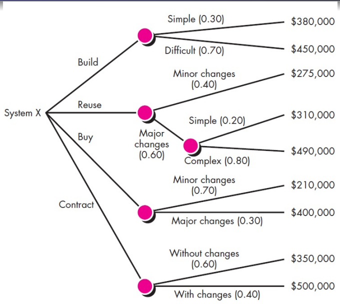
Make/Buy decision

Introduction

- Softwares are cost effective to acquire rather than develop computer software.
- For acquisition of software, make/buy decision is applied.

Conditions to be taken care in acquistion:

1. Will the delivery date of the software product be sooner than that for internally developed software?
2. Will the cost of acquisition plus the cost of customization be less than the cost of developing the software internally?
3. Will the cost of outside support (e.g., a maintenance contract) be less than the cost of internal support?



Software acquistion: based on make/buy decision

1. Expected cost= $\sum$ (path probability)$_i$ x (estimated path cost)$_i$
2. Expected cost$_{build}$ = 0.30 ($380K) + 0.70 ($450K)= $429K
3. Expected cost$_{reuse}$ =0.40 ($275K)+0.60 [0.20 ($310K) x 0.80 ($490K)] = $382K
4. Expected cost$_{buy}$ =0.70 ($210K) x 0.30 ($400K) = $267K——**less amount**
5. Expected cost$_{contract}$ =0.60 ($350K) x 0.40 ($500K)= $410K

Make/buy decision(Contd..)

- Outsourcing-Another way to develop a software.
- Software engineering activities are contracted to a third party who does the work at lower cost.
- The decision to outsource can be either strategic or tactical
  - Strategic level - a significant portion of all software work can be contracted to others
  - Tactical level - part or all of a project can be best accomplished by subcontracting the software work
- Pros and cons of the decision in organization perspective
  - Positive side - reducing the number of software people and the facilities
  - Negative side - company loses some control over the software that it needs.
- Outsourcing will undoubtedly continue
- To survive is to become as competitive as the outsourcing vendors themselves.

Summary

- Importance of project size metrics
- Importance of LOC in determining the software effort.
- Usage of function point to evaluate the project size.
- The project cost estimation for a given case study
  - COCOMO II estimation
- Make/buy decision

Assessment

- Count the LOC in a code
- Estimate the FP for a given case study
- A project estimation technique based on making an educated guess of the project parameters (such as project size, effort required to develop the software, project duration, cost etc.) is
  1. analytical estimation technique
  2. heuristic estimation technique
  3. empirical estimation technique
  4. none of the above
- An example of single variable heuristic cost estimation model is
  1. Halsteads software science
  2. basic COCOMO model
  3. intermediate COCOMO model
  4. complete COCOMO model
- Operating systems and real-time system programs can be considered as
  1. application programs
  2. utility programs
  3. system programs
  4. none of the above