Software Project Estimation

Software Project Estimation Techniques

- Experience (historical data)
- The Decomposition Techniques
- The Empirical Estimation Model

Software Project Estimation Techniques Cont'd

1. Experience (historical data) based Project Estimation

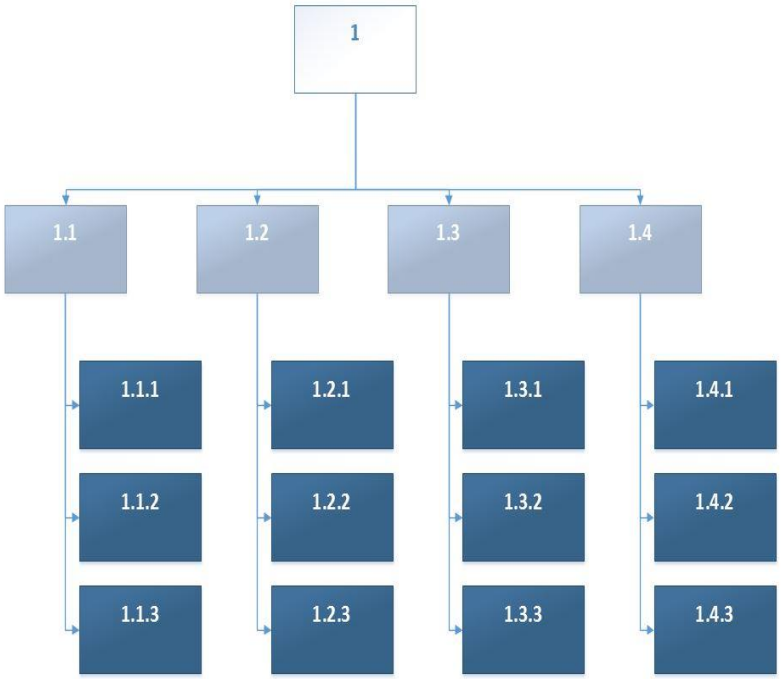
- A model is based on experience (historical data) and takes the form

$$d = f(v_i)$$

- where d is one of a number of estimated values (e.g., effort, cost, project duration) and vi are selected independent parameters (e.g., estimated LOC or FP).

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach



- The Project Estimation Approach that is widely used is Decomposition Technique.

- Decomposition techniques take a divide-and-conquer approach. Size, Effort and Cost estimation are performed in a stepwise manner by breaking down a Project into major Functions or related Software Engineering Activities.

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

2.1 Software Sizing

- A project estimate is only as good as the estimate of the size of the work to be accomplished.
- In project planning, size refers to a quantifiable outcome of the software project.
    - Direct approach - lines of code (LOC)
    - An indirect approach - Function points (FP)

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

2.1 Software Sizing (cont'd)

- Four different approaches to the sizing problem proposed by Putnam and Myers (1992)
  - Fuzzy logic sizing - the approximate reasoning techniques that are the cornerstone of fuzzy logic.
  - Function point sizing - The planner develops estimates of the information domain characteristics.
  - Standard component sizing - the standard components for an information system are subsystems, modules, screens, reports, interactive programs, batch programs, files, LOC, and object-level instructions.
  - Change sizing - the use of existing software that must be modified in some way as part of a project.

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

2.2 Problem-Based Estimation

LOC and FP data are used in two ways during software project estimation:

(1) As estimation variables to "size" each element of the software and

(2) As baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections.

- LOC and FP estimation are distinct estimation techniques. Yet both have a number of characteristics in common.
- Other components for sizing, such as classes or objects, changes, or business processes affected

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

2.2 Problem-Based Estimation Cont'd

- A three-point or expected value can then be computed.
- The *expected value* for the estimation variable (size) *S* can be computed as a weighted average of the optimistic (s_{opt}), most likely (s_m), and pessimistic (s_{pess}) estimates.

$$S = \frac{S_{opt} + 4S_m + S_{pess}}{6}$$

- gives the heaviest credence to the "most likely" estimate and follows a beta probability distribution.

- Once the expected value for the estimation variable has been determined, historical LOC or FP productivity data are applied.

- Productivity = KLOC / Person-month
- Quality = Defects / KLOC
- Cost = $ / LOC
- Documentation = pages of documentation / KLOC

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

Lines of Code (LOC)

- It is one of the earliest and simpler metrics for calculating the size of the computer program.
- It is generally used in calculating and comparing the productivity of programmers.
- These metrics are derived by normalizing the quality and productivity measures by considering the size of the product as a metric.
- As Lines of Code (LOC) only counts the volume of code, you can only use it to compare or estimate projects that use the same language and are coded using the same coding standards

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

An example of LOC based Estimation

- Take the Library management system case. Software developed for the library will accept data from the operator for issuing and returning books. Issuing and returning will require some validity checks. For issue, it is required to check if the member has already issued the maximum number of books allowed. In case of a return, if the member is returning the book after the due date then a fine has to be calculated. All the interactions will be through the user interface. Other operations include maintaining the database and generating reports at regular intervals.

Major software functions identified.

- User interface
- Database management
- Report generation

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

An example of LOC based Estimation

For user interface(Assumptions of LOC).

- S_{opt} : 1800
- S_m : 2000
- S_{pess} : 4000

For database management

- S_{opt}: 4600
- S_m: 6900
- S_{pess} : 8600

For report generation

- S_{opt}: 1200
- S_m: 1600
- S_{pess} : 3200

Expected Value(EV) for user interface

EV = (1800 + 4*2000 + 4000) / 6

EV = 2300

Expected Value(EV) for database management

EV = (4600 + 4*6900 + 8600) / 6

EV = 6800

Expected Value(EV) for user interface

EV = (1200 + 4*1600 + 3200) / 6

EV = 1800

Estimated lines of code

=2300+6800+1800

=10900

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

An example of LOC based Estimation

- For example, A review of historical data indicates that the organizational average productivity for systems of this type is 500 LOC/pm. Based on a burdened labor rate of $6000 per month.
  - The cost per line of code is approximate.
    - Cost/LOC=(6000/500)=$12
  - Based on the LOC estimate and the historical productivity data and By Considering the total estimated LOC as 10900,

    The total Estimated project cost =(10900*12)=$130800.

  - The estimated effort is

    The total estimated project effort = (10900/500) =21.8 Persons- Months

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

An example of LOC based Estimation

The problems of lines of code (LOC)

- Different languages lead to different lengths of code
- It is not clear how to count lines of code
- A report, screen, or GUI generator can generate thousands of lines of code in minutes
- Depending on the application, the complexity of the code is different.

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

FP-Based Estimation

- Decomposition for FP-based estimation focuses on information domain values rather than software functions.

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	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						*320*

Value adjustment factor 1.17

$$FP_{estimated} = \text{count total} \times [0.65 + 0.01 \times \Sigma(F_i)]$$

FP=320*1.17=375

- The organizational average productivity for systems of this type is 6.5 FP/pm.
- Based on a burdened labour rate of $8000 per month, the cost per FP is approximately $1230.
- Based on the FP estimate and the historical productivity data, the total estimated project cost is $461,000 and the estimated effort is 58 person-months.

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

2.3 Process-Based Estimation

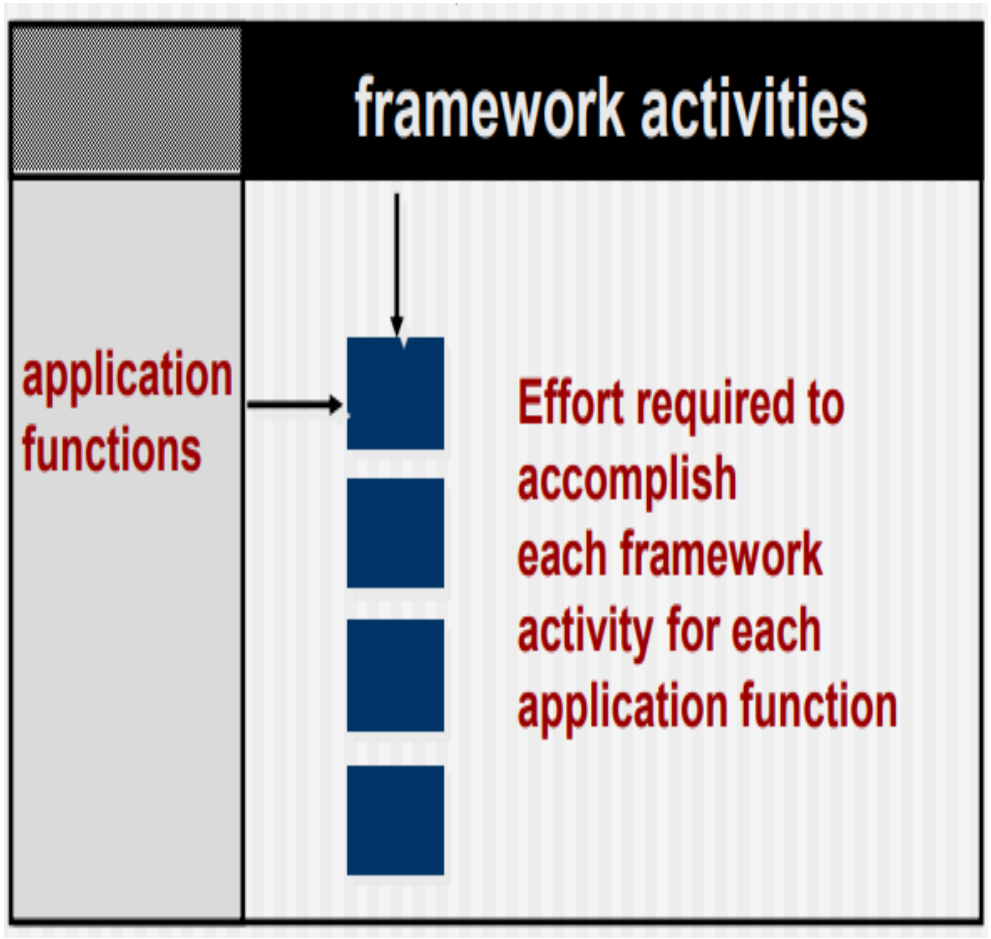
- The most common technique for estimating a project is to base the estimate on the process that will be used.
- The process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated.

- Like problem-based techniques, process-based estimation begins with a delineation of software functions obtained from the project scope.
- A series of framework activities must be performed for each function.
- Once problem functions and process activities are melded, you estimate the effort (e.g., person-months) that will be required to accomplish each software process activity for each software function.

Software Project Estimation Techniques Cont'd

2. The Decomposition Approach Cont'd

2.3 Process-Based Estimation Cont'd



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

Software Project Estimation Techniques Cont'd

3.Estimating With Use Case Points

- A Use-Case is a series of related interactions between a user and a system that enables the user to achieve a goal.
- Use-Case Points (UCP) is a software estimation technique used to measure the software size with use cases. The concept of UCP is similar to FPs.
- The Use-Case Point estimation method was introduced by Gustav Karner in 1993. The work was later licensed by Rational Software which merged into IBM.

The number of UCPs in a project is based on the following

- The number and complexity of the use cases in the system.
- The number and complexity of the actor in the system.
    - Various non-functional requirements such as portability, performance, and maintainability that are not written as use cases.
    - The environment in which the project will be developed such as the language, the team's motivation, etc.

Software Project Estimation Techniques Cont'd

3.Estimating With Use Case Points cont'd

The Use-Case Points counting process has the following steps –

- Unadjusted Use Case Weight (UUCW)
- Unadjusted Actor Weight (UAW)
- Technical Complexity Factor (TCF)
- Environmental Complexity Factor (ECF)

Use Case Points (UCP)

UCP = (UUCW + UAW) x TCF x ECF

Unadjusted Use Case Weight (UUCW)

Use Case Classification	No. of Transactions	Weight
Simple	1 to 3 transactions	5
Average	4 to 7 transactions	10
Complex	8 or more transactions	15

UUCW = (Total No. of Simple Use Cases x 5) + (Total No. Average Use Cases x 10) + (Total No. Complex Use Cases x 15)

Software Project Estimation Techniques Cont'd

3. Estimating With Use Case Points cont'd

Unadjusted Actor Weight (UAW)

Actor Classification	Type of Actor	Weight
Simple	External system that must interact with the system using a well-defined API	1
Average	External system that must interact with the system using standard communication protocols (e.g. TCP/IP, FTP, HTTP, database)	2
Complex	Human actor using a GUI application interface	3

UAW = (Total No. of Simple actors x 1) + (Total No. Average actors x 2) + (Total No. Complex actors x 3)

Technical Complexity Factor (TCF)

- The TCF is one of the factors applied to the estimated size of the software in order to account for technical considerations of the system

TCF = 0.6 + (TF/100)

Factor	Description	Weight
T1	Distributed system	2.0
T2	Response time/performance objectives	1.0
T3	End-user efficiency	1.0
T4	Internal processing complexity	1.0
T5	Code reusability	1.0
T6	Easy to install	0.5
T7	Easy to use	0.5

Factor	Description	Weight
T8	Portability to other platforms	2.0
T9	System maintenance	1.0
T10	Concurrent/parallel processing	1.0
T11	Security features	1.0
T12	Access for third parties	1.0
T13	End user training	1.0

Software Project Estimation Techniques Cont'd

3. Estimating With Use Case Points cont'd

Environmental Complexity Factor (ECF)

- The ECF is another factor applied to the estimated size of the software in order to account for the environmental considerations of the system.

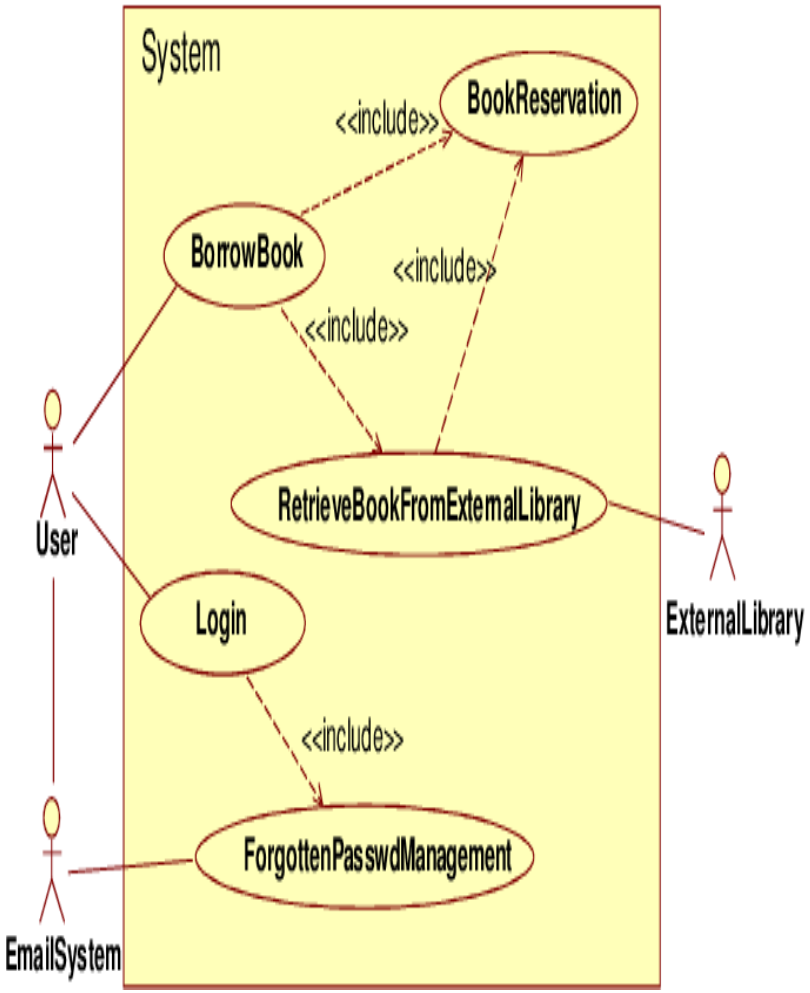
$$ECF = 1.4 + (-0.03 \times EF)$$

Factor	Description	Weight
E1	Familiarity with development process used	1.5
E2	Application experience	0.5
E3	Object-oriented experience of team	1.0
E4	Lead analyst capability	0.5
E5	Motivation of the team	1.0
E6	Stability of requirements	2.0
E7	Part-time staff	-1.0
E8	Difficult programming language	-1.0

Software Project Estimation Techniques Cont'd

3. Estimating With Use Case Points cont'd

Example



Unadjusted Use Case Weight (UUCW)

UUCW = (Total No. of Simple Use Cases x 5) + (Total No. Average Use Cases x 10) + (Total No. Complex Use Cases x 15).

UUCW = (1x 5) + (1 x 10) + (3 x 15) = 60

Unadjusted Actor Weight (UAW)

UAW = (Total No. of Simple Actors x 1) + (Total No. Average Actors x 2) + (Total No. Complex Actors x 3)

UAW = (1 x 1) + (0 x 2) + (2 x 3) = 7

Technical Complexity Factor (TCF)

TCF = 0.6 + (TF/100)

TCF = 0.6 + (49/100) = 1.09

Environmental Complexity Factor (ECF)

- ECF = 1.4 + (-0.03 x EF)
- ECF = 1.4 + (-0.03 * 10.0) = 1.1

Use Case Points (UCP)

- UCP = (UUCW + UAW) x TCF x ECF = (60+7)*1.09*1.1= 80.33

Software Project Estimation Techniques Cont'd

3. Estimating With Use Case Points cont'd

Use cases are problematic for the following reasons

- Use cases are described using many different formats and styles—there is no standard form.
- Use cases represent an external view (the user's view) of the software and can therefore be written at many different levels of abstraction.
- Use cases do not address the complexity of the functions and features that are described.
- Use cases can describe complex behavior (e.g., interactions) that involve many functions and features.

Software Project Estimation Techniques Cont'd

3. Estimating With Use Case Points cont'd

Use-Case-Based LOC Calculation

$$\text{LOC estimate} = N \times \text{LOC}_{\text{avg}} + [(S_a/S_h - 1) + (P_a/P_h - 1)] \times \text{LOC}_{\text{adjust}}$$

- N – the actual number of use cases
- LOC_{avg} -historical average LOC per use case for this type of subsystem
- $\text{LOC}_{\text{adjust}}$ - represents an adjustment based on n percent of LOC_{avg} where n is defined locally and represents the difference between this project and "average" projects.
- S_a - actual scenarios per use case
- S_h - average scenarios per use case for this type of subsystem
- P_a - actual pages per use case
- P_h - average pages per use case for this type of subsystem