Information System Design

IT60105

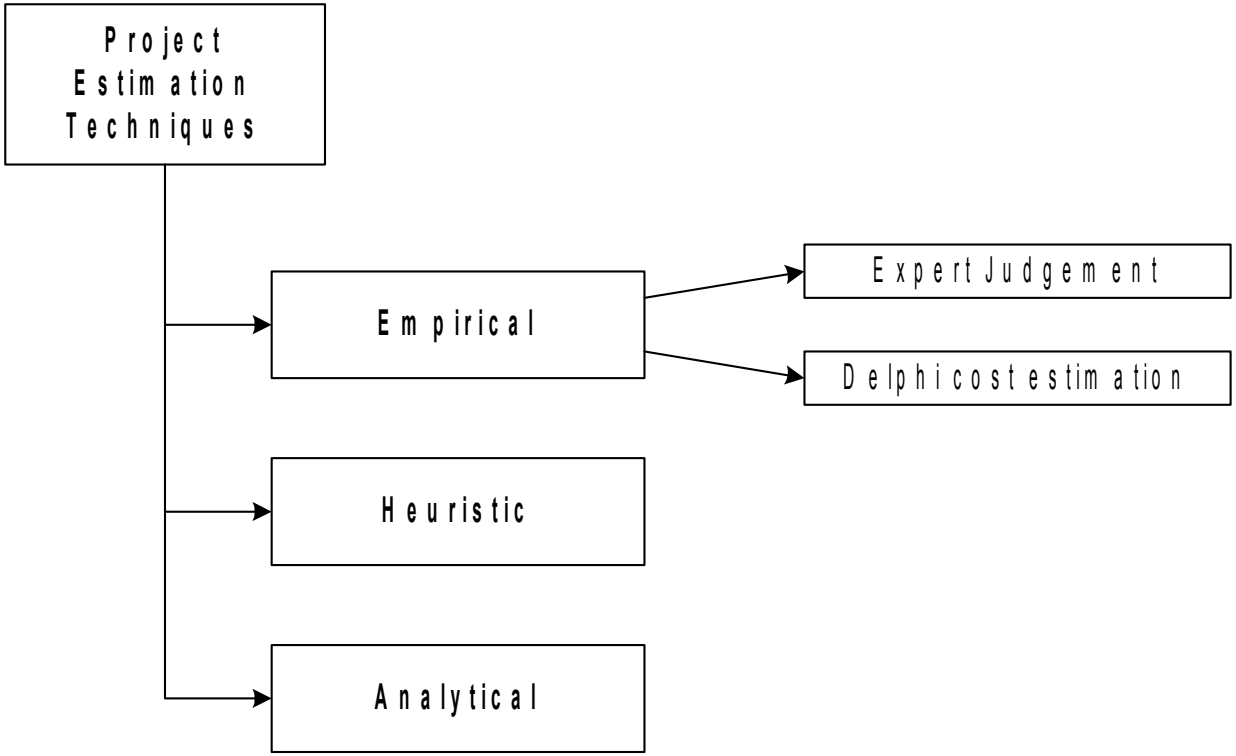
Lecture 20

Project Estimations

Lecture #20

- Project Estimation Techniques
  - Empirical estimation techniques
    - Expert judgment technique
    - Delphi cost estimation technique

Project Estimation Techniques



Empirical Estimation Techniques

Empirical Estimation Techniques

- Making an educated guess of the project parameters
- Prior experience with the development of similar project is helpful
- Two empirical estimation techniques are known:
  - Expert Judgment Technique
  - Delphi Cost Estimation

Expert Judgment Technique

- Expert thoroughly analyze the problems and then guess the problem size
- Estimations are usually based on measuring attributes of software
  - Size related metrics
  - Function related metrics

Empirical Cost Estimation

- **Size related metrics**
  - These are related to the size of some outputs from a project
  - The most commonly used size related metric is LOC (Lines Of delivered Code)
  - Other metrics are
    - The number of delivered object code instruction
    - The number of pages of system documentation etc.

Empirical Cost Estimation

- **Function related metrics**
  - These are related to the overall functionality of the delivered software
  - Measured in terms of number of functionalities produced in some given time
  - Example of function related metrics are
    - **Function points (FPs)**
    - Object points etc.

Basic Approach

- A metric is chosen as an estimation variable
- Project planner begins by estimating a range of values for each information domain
- Using historical data or intuition, the planner estimates an optimistic (S_{opt}), most likely (S_m), and pessimistic (S_{pess}) values or counts for each information domain value
- A three-point or expected-value can then be computed as a weighted of the three values

$$S = (S_{opt}+4S_m+S_{pess})/6$$

An Example of LOC Based Estimation

- Let us consider the SANJOG project
- Following values for LOC metrics have been evaluated

Functionality	S_{opt}	S_m	S_{pess}
Browser interface compatible with Hindi	4600	6900	8600
Browser interface compatible with Bengali	4800	7300	9200
Voice enabled interface and information retrieval	5200	8400	9600
Icon-based man-cyber interaction	2400	4200	6200
Dialogue-based information access	3200	5000	7400
E-mail based off-line Internet access	5600	8200	9800

An Example of LOC Based Estimation

- After the calculation of beta probabilistic values, the expected values can be calculated as shown

Functionality	**$S=(S_{opt}+4S_m+S_{pess})$**
Browser interface compatible with Hindi	6800
Browser interface compatible with Bengali	7200
Voice enabled interface and information retrieval	8067
Icon-based man-cyber interaction	4234
Dialogue-based information access	5100
E-mail based off-line Internet access	8034
Total LOC	**39434**

An Example of LOC Based Estimation

- From the historical data assume the following
  - A review of historical data indicates that the organizational average productivity for system of the project type is 420 LOC/PM
  - Cost for a person-month, α = Rs. 20,000/
  - Cost per line of code, β = Rs.50/
- We can estimate

  Effort = LOC/α = 94 Person-months

  Cost = LOC *β = Rs. 19,71,700 $\approx$ Rs. 21 lakhs

LOC: Size Related Metric

- **Salient features**
  - Simple yet useful
  - At the beginning of a project
  - Based on previous experience of the similar type of project
  - Shortcomings in LOC calculation
    - Can not address coding style
    - Coding activity only, not measure analysis, design, testing, documentation etc.
    - Does not correlates with quality and efficiency
    - Does not proper for 4GL, Library based, or HLL
    - Does not address structural, logical complexities (only lexical)

Function Point Based Metric

- The FP-based metric is first proposed by A. J. Albercht (1979)
- FP can be used to
  - Estimate the cost or effort required to design, code and test the software
  - Predict the number of errors that will be encountered during testing
  - Forecast the number of components and/or the number of projected source lines in the system under implementation

Project Size Estimation: FP-Based Metric

- Information domain values for the FP-metric are
  - Number of external inputs (**EI**s)
    - Each external inputs from 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*
  - Number of external outputs (**EO**s)
    - Each external output is derived within the application and provides information to the user. External outputs refers to report, screen, error messages etc.

Project Size Estimation: FP-Based Metric

- Information domain values for the FP-metric are
  - Number of external inquiries (**EQ**s)
    - An external enquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output
  - Number of internal logical files (**ILF**s)
    - Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs
  - Number of external interface files (**EIF**s)
    - Each external interface file is a logical grouping of data resides external to the the application but provides data that may be of use to the application

Project Size Estimation: FP-Based Metric

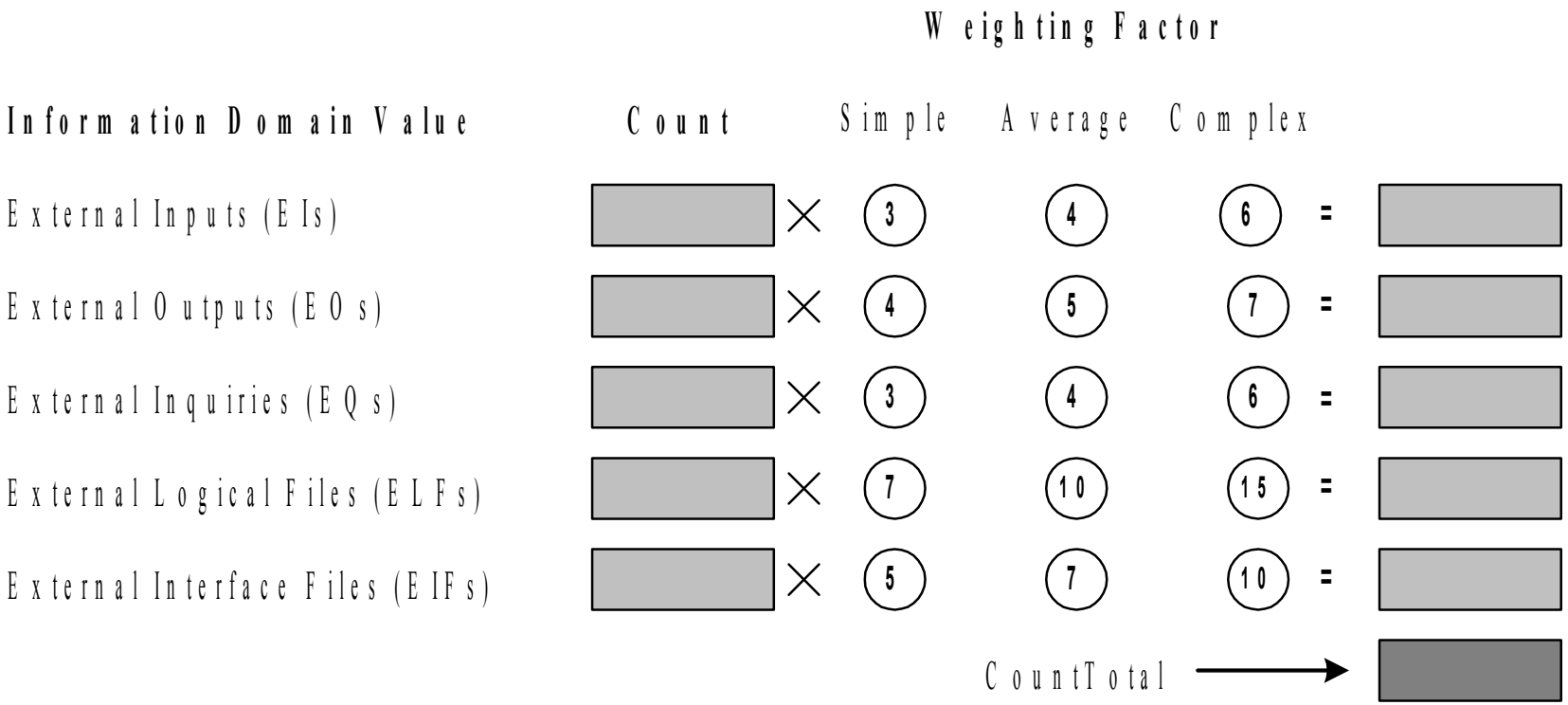
- To compute function points (FP), the following empirical relationship is used

$$FP = CountTotal \times \left[0.65 + 0.01 \sum \left(F_i\right)\right]$$

Here *CountTotal* is the sum of all FP entries

The F_i (i = 1 to 14) are *value adjustment factor* (***VAF***)

CountTotal in FP-Based Metric



VAF in FP-Based Metric

	VAF	Implication
01.	**Backup and Recovery**	Does the system require reliable backup and recovery?
02.	**Data Communication**	Are specialized data communications required to transfer information to or from the application?
03.	**Distributed Processing**	Are there distributed processing functions?
04.	**Performance Critical**	Is performance critical?
05.	**Existing Operating Environment**	Will the system run in an existing, heavily utilized operational environment?
06.	**Online Data Entry**	Does the system require online data entry?
07.	**Input Transaction over Multiple Screen**	Does the online data entry require the input transaction to be built over multiple screens or operations?

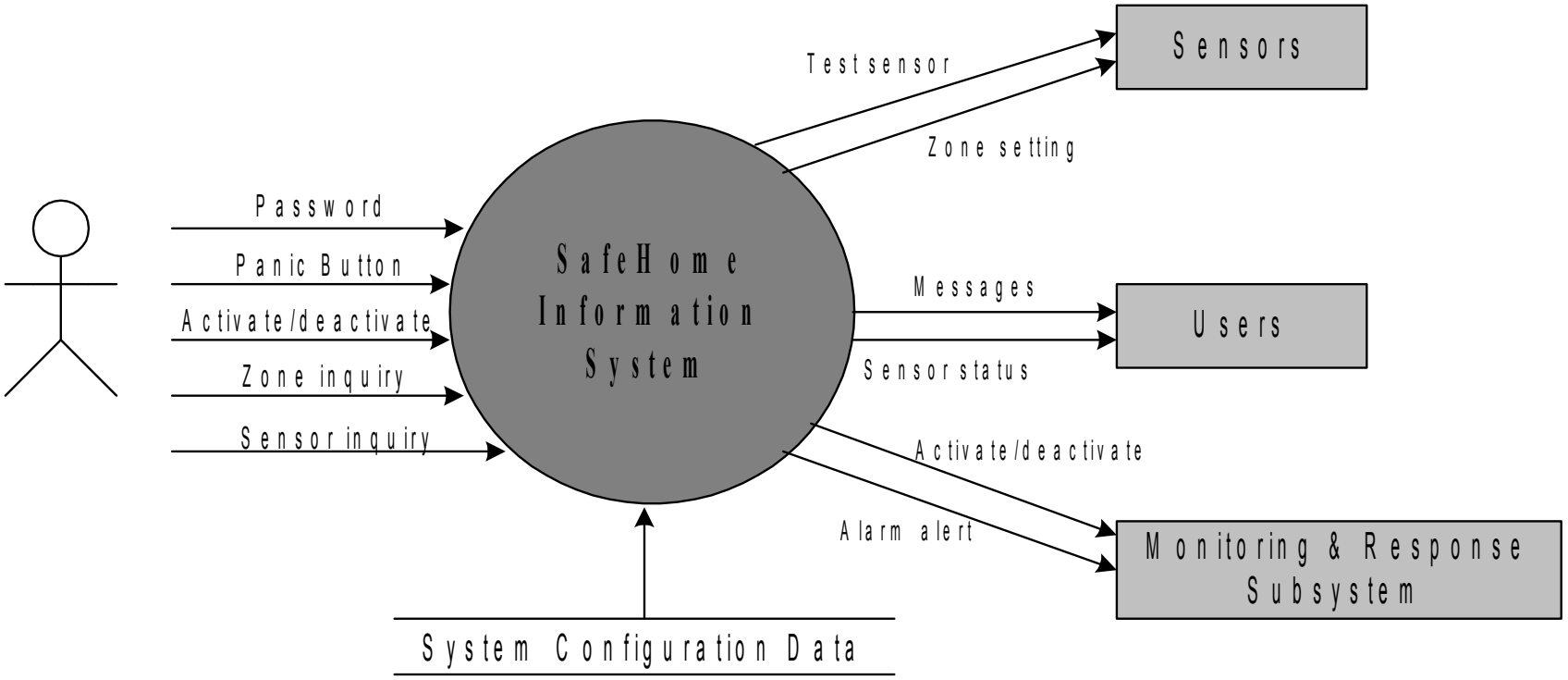
VAF in FP-Based Metric (Cont'd)

	VAF	Implication
08.	**ILFs Updated Online**	Are the ILFs updated online?
09.	**Information Domain Value Complex**	Are the inputs, outputs, files or inquiries complex?
10.	**Internal Processing Complex**	Is the internal processing complex?
11.	**Code Designed for Reuse**	Is the code designed to be reusable?
12.	**Conversation/Installation in Design**	Are the conversion and installation included in the design?
13.	**Multiple Installation**	Is the system designed for multiple installations in different organizations?
14.	**Application Designed for Change**	Is the application designed to facilitate change and for ease of use by the uesr?

Values in FP-Based Metric

- Each of the ***VAF*** is evaluated in scale of range from 0 (not important or applicable) to 5 (absolute essential)
- The constant values in the equation for FP is decided empirically
- The values for weighting factors that are applied to information domain counts are also determined empirically

An Example of FP-Based Estimation



An Example of FP-Based Estimation

External inputs

- Password
- Panic Button
- Activate/deactivate

External outputs

- Messages
- Sensor status

External inquiries

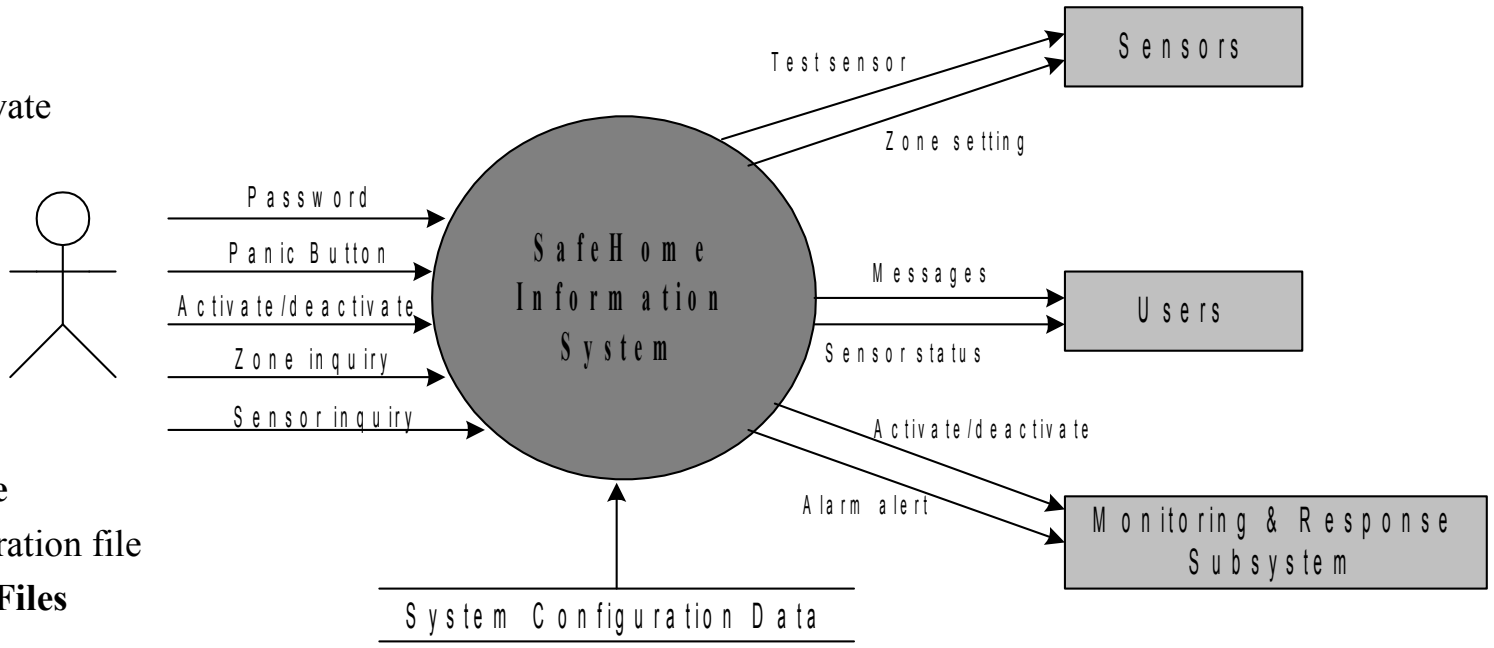
- Zone inquiry
- Sensor inquiry

Internal Logical file

- System configuration file

External Interface Files

- Test sensors
- Zone setting
- Activate/deactivate
- Alarm alert



An Example of FP-Based Estimation

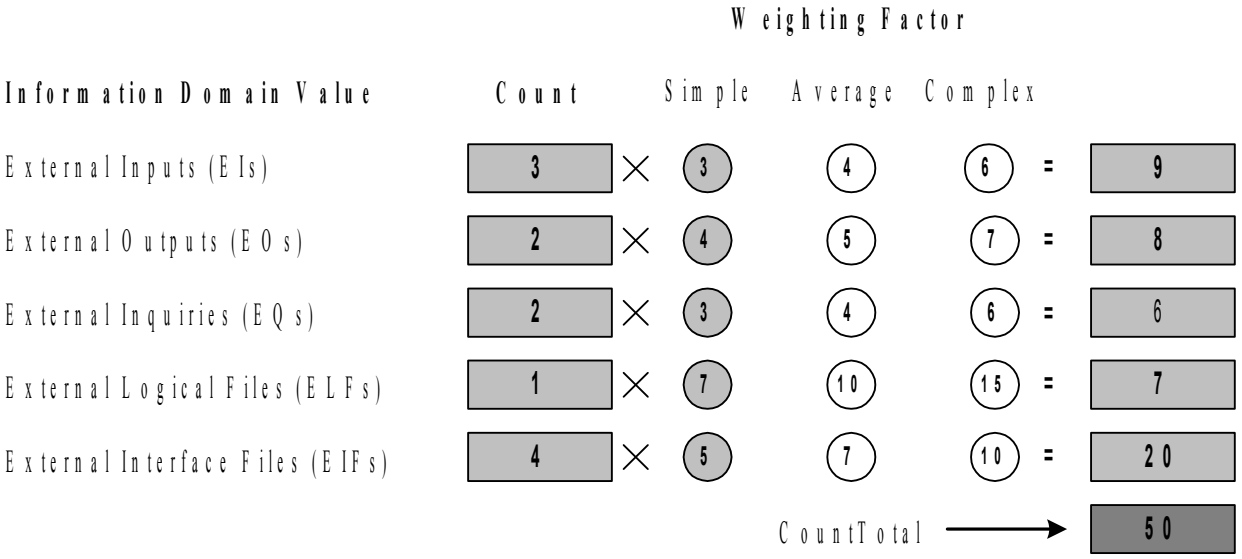
External inputs
- Password
- Panic Button
- Activate/deactivate

External outputs
- Messages
- Sensor status

External inquiries
- Zone inquiry
- Sensor inquiry

Internal Logical file
- System configuration file

External Interface Files
- Test sensors
- Zone setting
- Activate/deactivate
- Alarm alert



An Example of FP-Based Estimation

VAF		Value
01.	**Backup and Recovery**	4
02.	**Data Communication**	2
03.	**Distributed Processing**	0
04.	**Performance Critical**	4
05.	**Existing Operating Environment**	3
06.	**Online Data Entry**	4
07.	**Input Transaction over Multiple Screen**	5
08.	**ILFs Updated Online**	3
09.	**Information Domain Value Complex**	5
10.	**Internal Processing Complex**	5
11.	**Code Designed for Reuse**	4
12.	**Conversation/Installation in Design**	3
13.	**Multiple Installation**	5
14.	**Application Designed for Change**	5
	Total **VAF**	**62**

An Example of FP-Based Estimation

- The estimated number of FP can be derived as

$$FP = CountTotal \times \left[0.65 + 0.01 \sum \left(F_i\right)\right]$$

$$= 50 \text{ X } [0.65 + 0.01 \text{ X } 62]$$

$$= 63.5$$

An Example of FP-Based Estimation

Suppose the organizational average productivity for system of this type = 1.2 FP/PM

Cost of a PM = Rs. 20,000/

Effort = FP/Productivity

= 63.5/1.2 = 53 PM

Cost = 53 X 20,000 = Rs. 10,60,000 ≈ 11 lakhs

Pros and Cons of FP-Based Estimation

- This metric is language independent and can be easily computed from the SRS document during project planning
- This metric is subjective and require a sleight of hand
- Different engineer can arrive at different FP for the same project

Expert Judgment Technique: Summary

- Expert thoroughly analyze the problems and then guess the problem size: LOC or FP
- Drawback:
  - Technique is subject to human errors and biased with individual
  - Expert may overlook some factors inadvertently
  - Expert may not have experience and knowledge of all aspects of projects
- Remedy
  - Estimation made by a group of experts

Delphi Cost Estimation

- Estimated by a team (composed with a group of experts) and a coordinator
- Individual team member estimates based on SRS supplied by the coordinator
- Estimator points out typical characteristic (s) by which s/he has been influenced while estimating
- Based on the input from all estimators, coordinator prepared a summary sheet and distributes the same to all estimators emphasizing the important things noted by others
- Based on the summary, estimators re-estimate and the process may be iterated depending on the satisfaction of the coordinator. Coordinator is responsible for compiling final results and preparing the final estimates

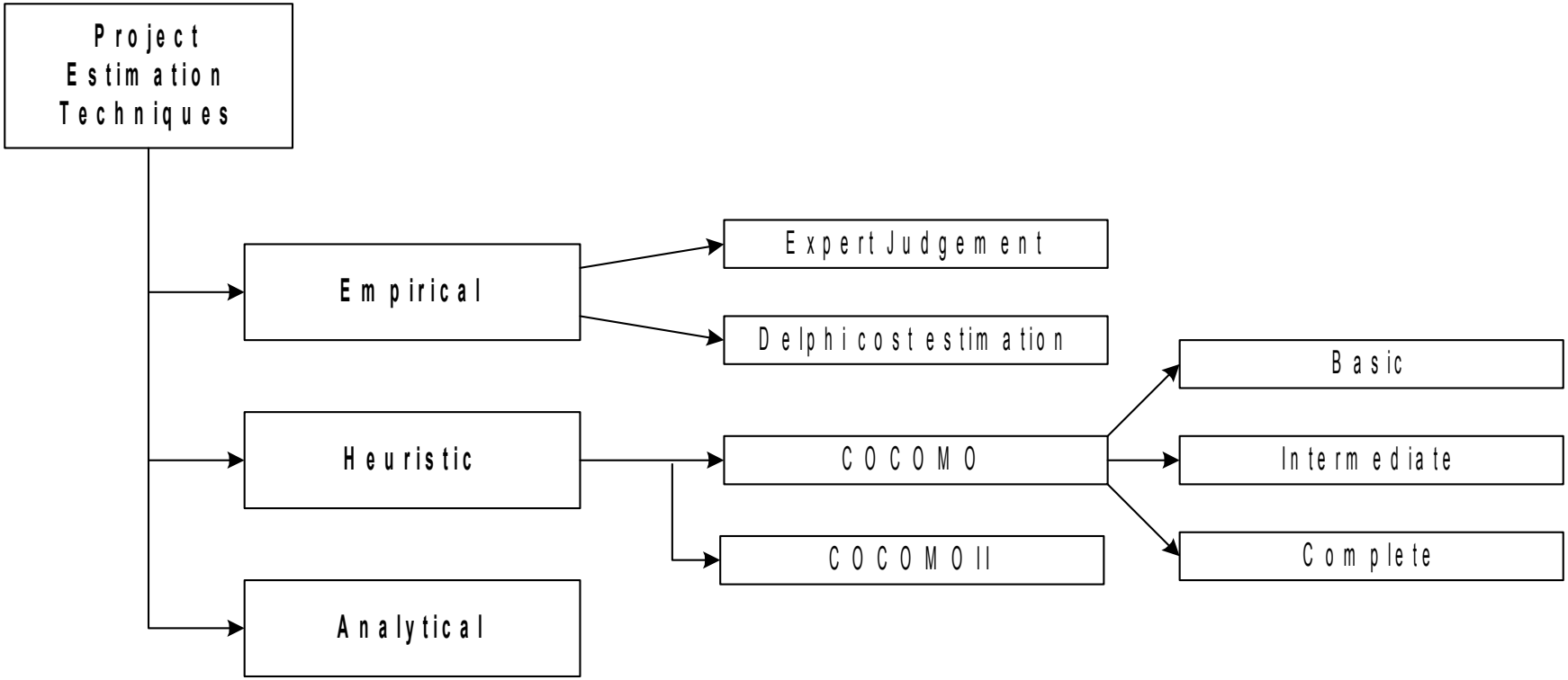
Note: An estimator is opaque to any other estimators

Heuristic Estimation Techniques

Heuristic Estimation Techniques

- Project Estimation Techniques
  - Heuristic estimation techniques
    - COCOMO (1981)
    - COCOMO II (2000)

Project Estimation Techniques



Heuristic Estimation Models

- Heuristic estimation models are derived using regression analysis on data collected from past software projects

- The overall structure of such models takes the form

$$E = A + B \times (e_v)^C$$

- Where A, B, and C are empirically derived constants, E is effort in person-months, and e_v is the estimation variable (either LOC or FP)

Heuristic Estimation Models

- In addition to the general form, they have some project adjustment component that enables E to be adjusted by other project characteristic (e.g. problem complexity, staff experience, development environment etc.)
- Based on the study of different types of project, a rule of thumbs in the form of mathematical expression
- The heuristic estimation models are also alternatively termed as ***Algorithmic Cost Models***

Some Heuristic Estimation Models

LOC-oriented Heuristic Estimation Models

Walston-Felix model	$E = 5.2 \times (KLOC)^{0.91}$
Baliey-Basili model	$E = 5.5 + 0.73 \times (KLOC)^{1.16}$
Boehm simple model	$E = 3.2 \times (KLOC)^{1.05}$
Doty model for KLOC > 9	$E = 5.288 \times (KLOC)^{1.047}$

FP-oriented Heuristic Estimation Models

Albrecht and Gaffney model	$E = -91.4 + 0.355 \times FP$
Kemerer model	$E = -37 + 0.96 \times FP$
Small project regression model	$E = -12.88 + 0.405 \times FP$

Boehm's COCOMO (1981)

- COCOMO (**CO**nstructive **CO**st estimation **MO**del) – A heuristic estimation technique proposed by Boehm (1981)

- It has been widely used and evaluated in a range of organizations

- It is a comprehensive models with a large number of parameters that can each take a range of values

Boehm's COCOMO 81

- Boehm's classification of projects:
  - **Organic**
    - Size is reasonably small
    - Project deals with developing a well-understood application
    - Team is experienced in developing similar types of projects
  - **Semidetached**
    - Relatively larger size
    - Development team consist of mixed members with experienced and inexperienced staff
    - Team may not familiar with some aspects of system parts
  - **Embedded**
    - Very big systems
    - Team with inexperienced staff
    - Team members are unfamiliar to the most of the system parts

Boehm's COCOMO 81

- Three-level model of estimations:
  - Basic (provides an initial rough estimation)
    - Approximate estimation of the cost

$$Effort = a_1 \times (KLOC)^{a_2}$$
$$Time = b_1 \times (Effort)^{b_2}$$

Project	**Effort** in PM		**Time** in month	
	a_1	a_2	b_1	b_2
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Boehm's COCOMO 81

- Intermediate (modification of the basic estimation)
  - Use the nominal cost as estimated in Basic COCOMO and multiplied by 15 cost drivers on product complexity, computing environment, personnel, development tools etc.
- Complete (detailed estimation)
  - More improved than the previous two models
  - Suitable for heterogeneous projects
  - At the perspective of a system with several subsystems with various complexity (not a single entity rather)

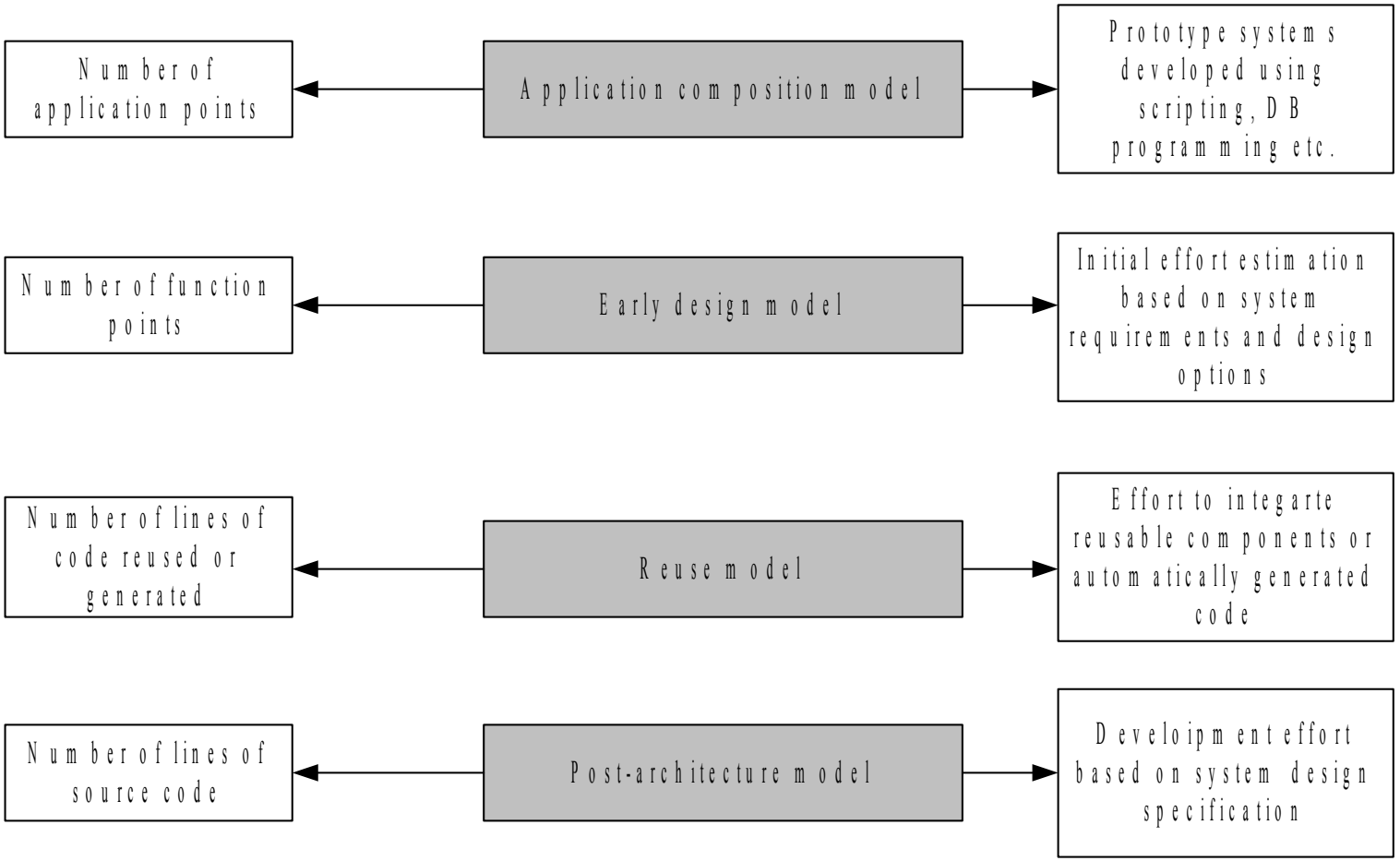
Need to Re-Engineer COCOMO 81

- New software processes
- New sizing phenomena
- New reuse phenomena
- Need to make decisions based on incomplete information

COCOMO II

- COCOMO assumed that the software would be developed according to a Waterfall Process
- Using standard imperative programming language such as C or FORTRAN
- COCOMO II is to take the latest development in software technology into account
- COCOMO II supports a Spiral model of development
- Also embeds several sub-models that produce increasingly detailed estimates

Sub-models in COCOMO II



Application-Composition Model

- It is introduced to estimate
  1. Prototyping of software
  2. Composing software by existing software components
- This model assumes that systems are created from reusable components, scripting or database programming
- Software size estimates are based on application points (same as the object points)

Application-Composition Model

- The formula for computing effort for a system prototype according to this model is

$$PM = \left(NAP \times \left(1 - \%reuse / 100 \right) \right) / PROD$$

where *PM* is the effort estimate in person-month.

NAP is the total number of application points in the delivered system.

%reuse is an estimate of the amount of reused code in the development.

PROD is the object point productivity.

Object Point Productivity

- The PROD can be calculated using the following table

Developer's Experience and Capability	Very Low	Low	Nominal	High	Very High
CASE maturity and capability	Very Low	Low	Nominal	High	Very High
PROD (NOP/month)	4	7	13	25	50

Early Design Model

- It is used at Exploration phase when,
  - User requirement is known
  - Detailed architecture is not developed yet
- Goal is to make an approximate estimate without much effort
- The estimate produced at this stage are based on standard formula for algorithmic models

Early Design Model

- The estimation formula is

  Effort = A X $Size^B$ X M

  where A, B, and M are constants

  The *Size* of the system is expressed in KSLOC, which is the number of thousands of lines of source code

  M is a multiplication of 10 other drivers

Significance of the Constants

- The value of coefficient *A* used by Boehm is 2.94, calculated from a large data set
- *B* depends on novelty of the project, development flexibility, process maturity level of the organization etc. and varies between 1.1 and 1.24
- The multiplier *M* depends on a simplified set of **seven** project and process characteristics e.g.: Product reliability and complexity, Reuse required, Platform difficulty etc.

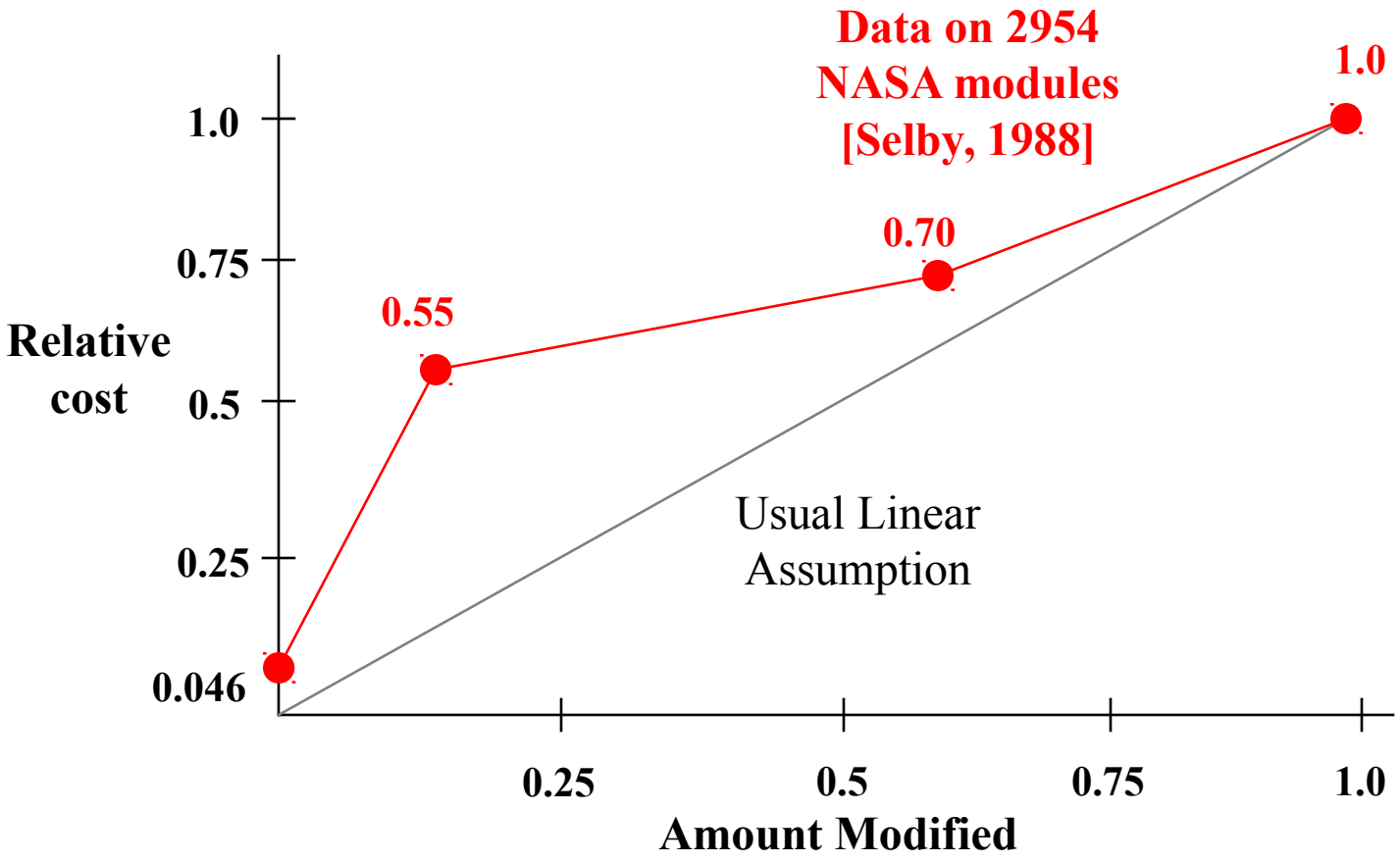
The Reuse Model

- Used to estimate the effort required to integrate reusable or generated code

- Two types of reused codes
  1. **Black box type**: Code can be integrated without understanding the code or making changes to it: development effort is zero e.g. ActiveX used in VB or VC++ projects
  2. **White box Type**: Code has to be adapted before reusing e.g. downloaded programs from internet

- Another type of reusable code is automatically generated code by CASE tools, e.g. code generated by Rational Rose. COCOMO II includes a separate model to tackle this type of code

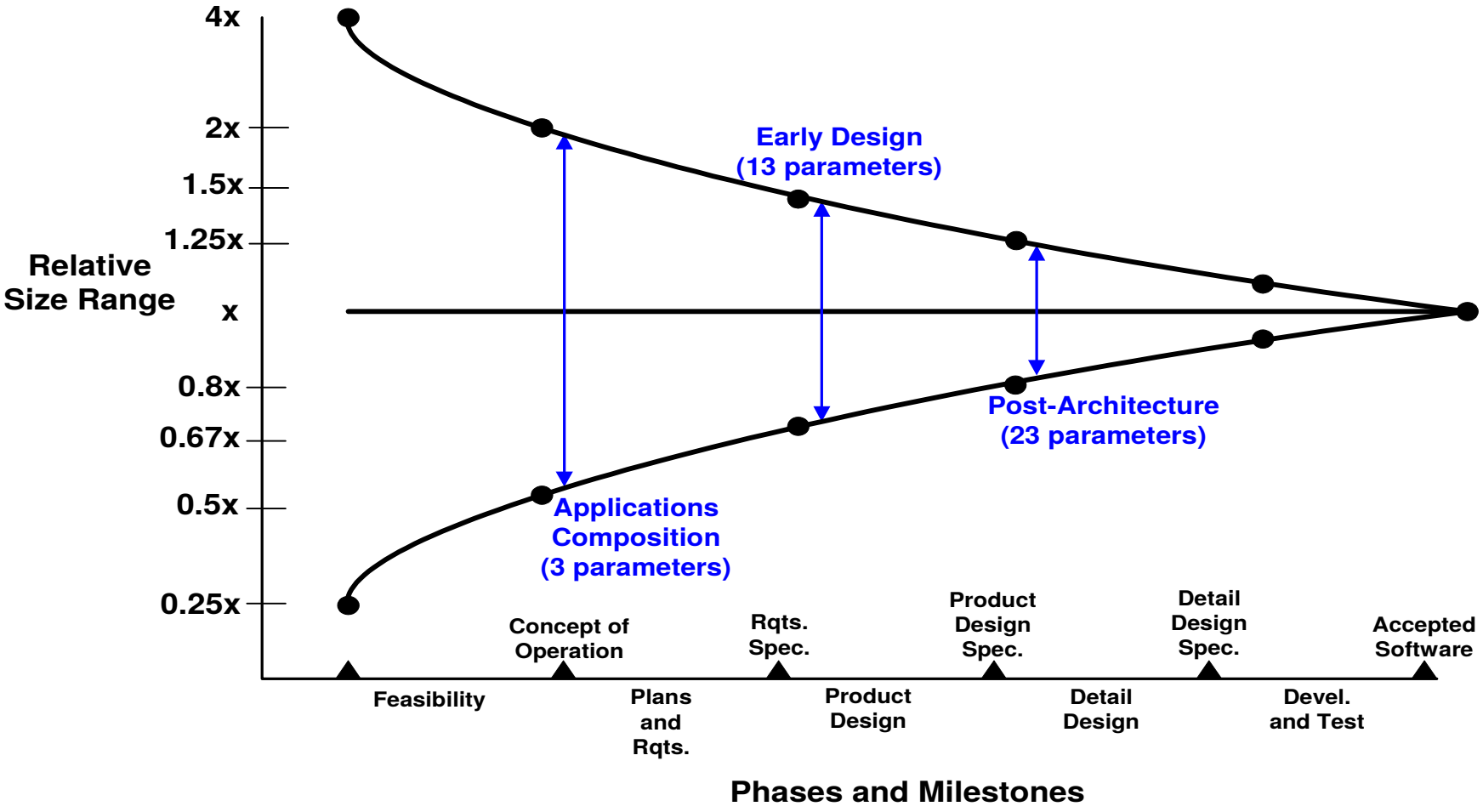
Need of the Reuse Model: Nonlinear Reuse Effects



The Post Architecture Model

- It is the most detailed of the COCOMO II models
- It is used once an initial architecture design for the system is available
- It is based on the same basic formula as in the Reuse model; here only the Size estimate is more accurate
- Further it employs 17 drivers instead of 7 drivers as in Reuse model

COCOMO II Model Stages



Major Decision Situations Helped by COCOMO II

- Software investment decisions
  - When to develop, reuse, or purchase
  - What legacy software to modify or phase out
- Setting project budgets and schedules
- Negotiating cost/schedule/performance tradeoffs
- Making software risk management decisions
- Making software improvement decisions
  - Reuse, tools, process maturity, outsourcing

Extension of COCOMO II

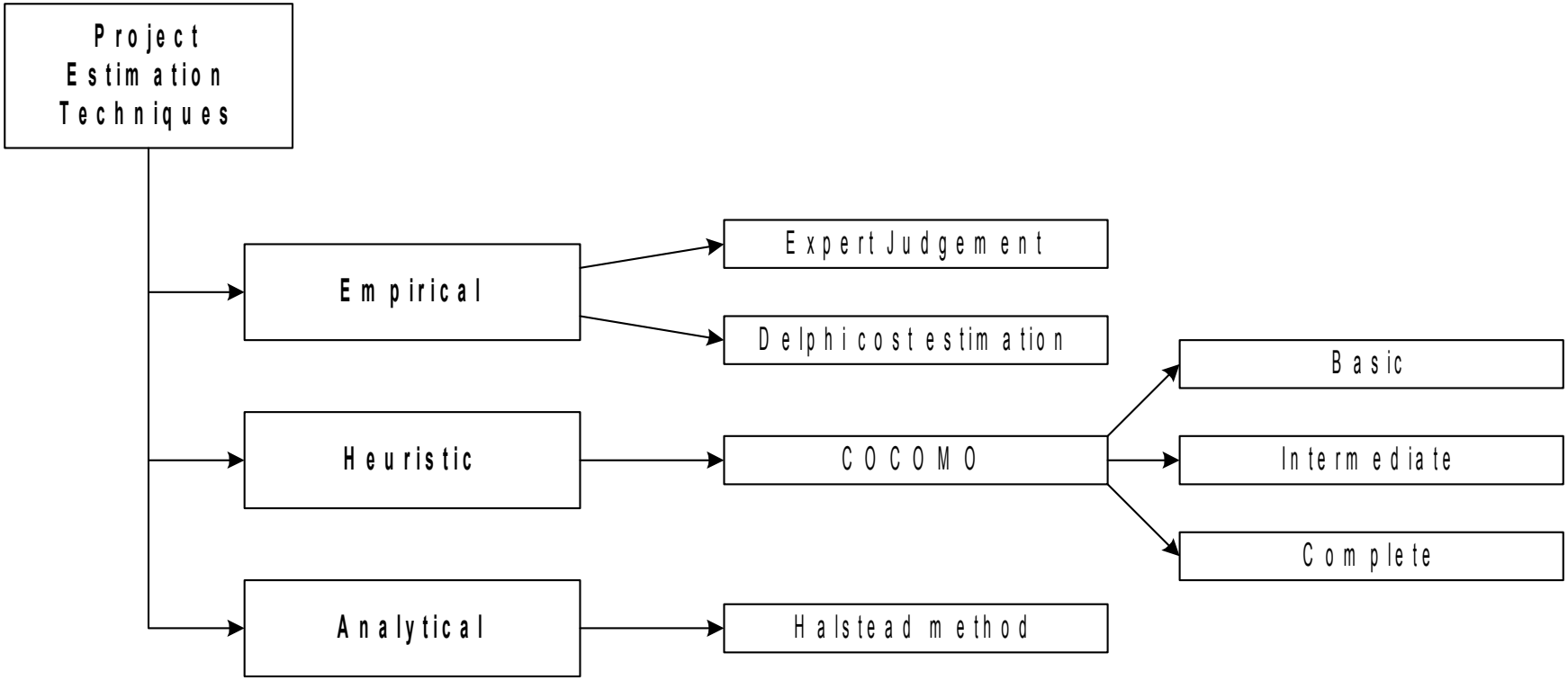
- COTS Integration (COCOTS)
- Quality: Delivered Defect Density (COQUALMO)
- Phase Distributions (COPSEMO)
- Rapid Application Development Schedule (CORADMO)
- Productivity Improvement (COPROMO)
- System Engineering (COSYSMO)
- Tool Effects
- Code Count (COCOTM)

➢ **Further Information:**

http://sunset.usc.edu/research/COCOMOII/

Analytical Methods

Analytical Models



Halstead's Analytical Method

- **Maurice Halstead** proposed a theory of "Software Science" in 1977
- The first analytical *laws* for computer software
- Use a set of primitive measures that may be generated after code is generated or estimated once design is complete
- His simple models are still considered valid
- The basic approach to consider any program to be a collection of tokens
- He proposed four primitives in his measure

η_1 = Number of **unique operators** that appear in a program
η_2 = Number of **unique operands** that appear in a program
N_1 = Total number of **occurrences of operators**
N_2 = Total number of **occurrences of operands**

Halstead's Analytical Method

- Halstead uses the primitives to estimate the following
  - Overall program length (L)
    - Length of the programs in token
  - Vocabulary (η)
    - Distinct token use in a program
  - Program volume (V)
    - The number of bits required to specify a program
  - Potential minimum volume (V^*)
    - Minimum program volume required to implement for a given algorithm
  - Volume ratio (l)
    - To measure program level
  - Development effort (E)
    - How much effort is needed to develop a program
  - Development time (T)
    - Time required to develop a program

Halstead's Analytical Method

- Estimates
  - Length

    $$N = \#\,Operators + \#Operands$$
  - Vocabulary

    $$\eta = \#\,Unique\ Operators + \#Unique\ Operands$$
  - Program volume

    $$V = N \log_2 \eta$$

[Halstead assumes the volume as a 3D measure, when it is really related to the number of bits it would take to encode the program being measure. Encoding n different items would require at a minimum $log_2 n$ bits for each. To encode a sequence of N such items would require $N*log_2 n$]

Halstead's Analytical Method

- Consider the following code that does multiplication by repeated addition

```
1.      Z = 0;
2.      While X > 0
3.           Z = Z + Y;
4.           X = X – 1;
5.      End-while;
6.      Print(Z);
```

Identify the unique operators and operands in the program and hence program volume.

Halstead's Analytical Method

- Estimates
  - Potential minimum volume

$$V^* = (2+\eta_2)\log_2(2+\eta_2) \quad [\eta_2 = \#\text{Unique operands}]$$

[Halstead assume that in the minimal implementation, there would only be two operators: the name of the function and a grouping operators. η_2 is the number of arguments in the function call.]

  - Volume ratio

$$l = V^*/V$$

[This relates how close the current implementation is to the minimal implementation. This must be always less than 1.]

Halstead's Analytical Method

- Estimates
  - Effort

    $E = V/l = V^2/V^*$

    [The unit is elementary mental discrimination (*emd*), a notation proposed by Halstead.]

  - Time

    Time = E/S, where S = Stroud's number

    [In this estimation, Halstead use some work developed by a psychologist John Stroud (1950). Stroud measured how fast a subject could view items passed rapidly in front of his face. *S*, the Stroud's number (*emd/sec*) implies the speed of mental discrimination. Halstead used 18 as the value of *S*.]

Halstead's Analytical Method: An Example

- Suppose it is required to calculate the *Effort* for a program, which is to search a value stored in an array of finite size
- The "Binary Search" technique can be followed in the implementation
- There are two versions of "Binary Search", namely
  - Iterative
  - Recursive

Example of Halstead's Method

- Iterative "Binary Search"

```
Line#
1.      While     x ≠ NIL and K ≠ Key[x]
2.                Do    If K < Key[x]
3.                      Then x ← Left[x]
4.                      Else  x ← Right[x]
5.      Return x
```

Iterative "Binary Search"

Line No.	Operators	Operands	Unique Operators	Unique Operands
1	while, <>,nil, and,<>,[ ]	x, k, key[x]	while, <>,nil, and, [ ]	x, k, key[x]
2	do, if, <, [ ]	k, key[x]	do, if, <	
3	then, <-, [ ]	x, left[x]	then, <-	left[x]
4	else,<-,[ ]	x, right[x]	else	right[x]
5	return	x	return	
	17	10	12	5

Iterative "Binary Search"

- Estimates

  Length, N = *# Operators + #Operands*

  = **17 + 10 = 27**

  Vocabulary, η = *# Unique Operators(*η_1*) + #Unique Operands(*η_2*)*

  = **12 + 5 = 17**

  Program volume, $V = N \log_2 \eta$

  $27 \log_2 17 = 110.36$

  Potential minimum volume

  $V^* = (2+\eta_2) \log_2 (2+\eta_2)$ [*For a function and its call*]

  $= 14 \log_2 5 = 32.5$

  Effort and Time

  Effort = V^2/V^* = 374.75, Time = Effort/S = 20.82

  Where S = speed of mental discrimination (usually, S = 18)

Example of Holstead's Method

- Recursive "Binary Search"

```
Line#
1.      If      x ≠ NIL or K ≠ Key[x]
2.      Then    Return x
3.      If      K < Key[x]
4.      Then        Return BinarySearch(Left[x], K)
5.      Else        Return BinarySearch(Right[x], K)
```

Recursive "Binary Search"

Line No.	Operators	Operands	Unique Operators	Unique Operands
1	if, =, nil, or, =,[ ]	x, k, key[x]	if, =, nil, or, [ ]	x, k, key[x]
2	then, return	x	then, return	
3	if, <, [ ]	k, key[x]	<	
4	then, return, tree_search( ), ( ), [ ]	k, left[x]	tree_search( ) , ( )	left[x]
5	else, return, tree_search( ), ( ), [ ]	k, right[x]	else	right[x]
	21	10	11	5

Recursive "Binary Search"

- Estimates

  Length, N = *# Operators + #Operands*

  = **21 + 10 = 31**

  Vocabulary, η = *# Unique Operators(*η_1*) + #Unique Operands(*η_2*)*

  = **11 + 5 = 16**

  Program volume, $V = N \log_2 \eta$

  31 $\log_2$16 = 124

  Potential minimum volume

  $V^* = (2+\eta_2) \log_2 (2+\eta_2)$ [*For a function and its call*]

  $= 13\log_2 7 = 36.49$

  Effort and Time

  Effort = V^2/V^* = 421.26, Time = Effort/S = 23.40

  Where S = speed of mental discrimination (usually, S = 18)

Problems to Ponder

- What are the usefulness of LOC and FP metrics in project cost estimation? (Give relative merits and demerits)
- Cost estimations are inherently risky irrespective of technique used. Suggest few ways (at least four) in which the risk in an empirical cost estimation can be reduced.
- Some very large software projects involve writing millions of lines of codes. How useful the empirical estimation technique is for such system?

Problems to Ponder

- Different estimation models predict different results for the same values of LOC or FP. What is the significance of their existence?

- It is argued that ***Algorithmic models*** can be used to support quantitative option analysis. How?

- The time required to complete a project is not simply proportional to the number of person working on the project