Estimation for Software Project

Ashish Kumar Dwivedi

Metrics

- Process, Project and Product Metrics
- Estimation for Software Project: COCOMO
- Software metrics are analyzed and assessed by software managers.
- Measures are often collected by software engineers.
- If you don’t measure, judgment can be based only on subjective evaluation

Software Measurements

Size-oriented metrics

Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
•	•	•	•	•	•		
•	•	•	•	•	•		
•	•	•	•	•	•		

Programming Language	LOC per Function Point			
	Average	Median	Low	High
Access	35	38	15	47
Ada	154	—	104	205
APS	86	83	20	184
ASP 69	62	—	32	127
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
Clipper	38	39	27	70
COBOL	77	77	14	400
Cool:Gen/IEF	38	31	10	180
Culprit	51	—	—	—
DBase IV	52	—	—	—
Easytrieve+	33	34	25	41
Excel47	46	—	31	63
Focus	43	42	32	56
FORTRAN	—	—	—	—
FoxPro	32	35	25	35
Ideal	66	52	34	203
IEF/Cool:Gen	38	31	10	180
Informix	42	31	24	57
Java	63	53	77	—
JavaScript	58	63	42	75
JCL	91	123	26	150
JSP	59	—	—	—
Lotus Notes	21	22	15	25
Mantis	71	27	22	250
Mapper	118	81	16	245
Natural	60	52	22	141
Oracle	30	35	4	217
PeopleSoft	33	32	30	40
Perl	60	—	—	—
PL/1	78	67	22	263
Powerbuilder	32	31	11	105
REXX	67	—	—	—
RPG II/III	61	49	24	155
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
VBScript36	34	27	50	—
Visual Basic	47	42	16	158

Questions:

- Compute the function point value for a project with the following information domain characteristics: Number of user inputs: 32, Number of user outputs: 60, Number of user inquiries: 24, Number of files: 8, Number of external interfaces: 2, Assume that all complexity adjustment values are average. **It is also given that, out of fourteen value adjustment factors that influence the development effort, four factors are not applicable, each of the other four factors has value 3, and each of the remaining factors has value 4. The computed value of the function point metric is….**
- The software used to control a photocopier requires 32,000 lines of C and 4,200 lines of Smalltalk. Estimate the number of function points for the software inside the copier
- Ans:

1 FP per 162 lines of C and 1 FP for 26 lines of Smalltalk.

So,

FP = (32000/162) + (4200/26)

FP = 359.069

Function Points

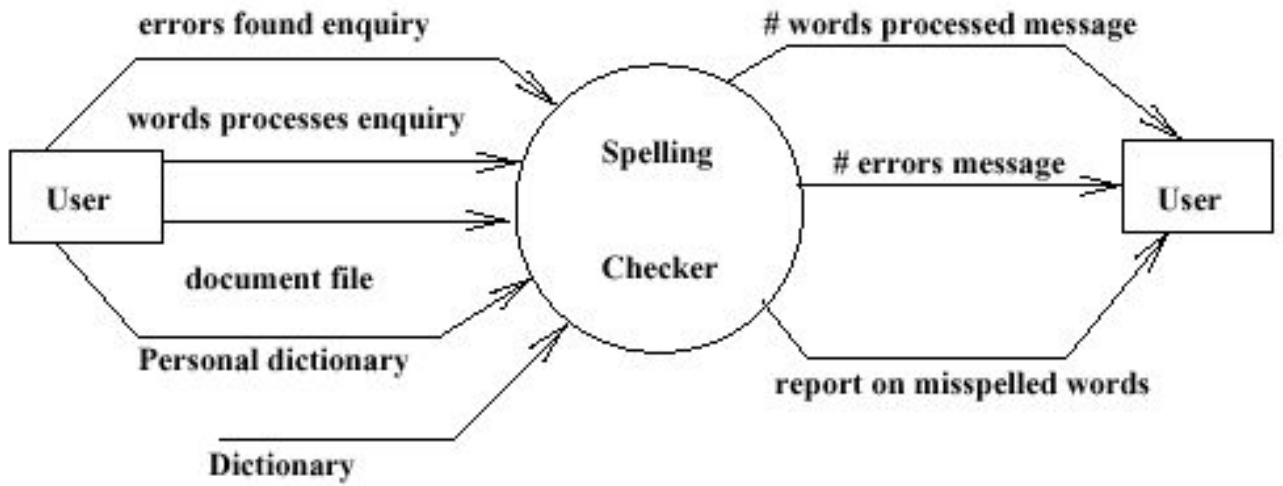
<u>STEP 1</u>: measure size in terms of the amount of functionality in a system. Function points are computed by first calculating an ***unadjusted function point count*** **(UFC).** Counts are made for the following categories

- *External inputs* – those items provided by the user that describe distinct application-oriented data (such as file names and menu selections)
- *External outputs* – those items provided to the user that generate distinct application-oriented data (such as reports and messages, rather than the individual components of these)
- *External inquiries* – interactive inputs requiring a response
- *External files* – machine-readable interfaces to other systems
- *Internal files* – logical master files in the system

Example

Example

The Spell-Checker accepts as input a document file and an optional personal dictionary file. The checker lists all words not contained in either of these files. The user can query the number of words processed and the number of spelling errors found at any stage during processing.



Example (.)

- 2 users inputs: document file name, personal dictionary name (average)
- 3 users outputs: fault report, word count, misspelled error count (average)
- 2 users requests: #treated words?, #found errors? (average)
- 1 internal file: dictionary (average)
- 2 external files: document file, personal dictionary (av).

$$UFP = 4 \times 2 + 5 \times 3 + 4 \times 2 + 10 \times 1 + 7 \times 2 = 55$$

Relation between LOC and FP

- Relationship:
  - ***LOC = Language Factor * FP***
  - where
    - **LOC** (Lines of Code)
    - **FP** (Function Points)

Relation between LOC and FPs

Language	LOC/FP
assembly	320
C	128
Cobol	105
Fortan	105
Pascal	90
Ada	70
OO languages	30
4GL languages	20

Relation between LOC and FP(.)

Assuming LOC's per FP for:

Java = 53,

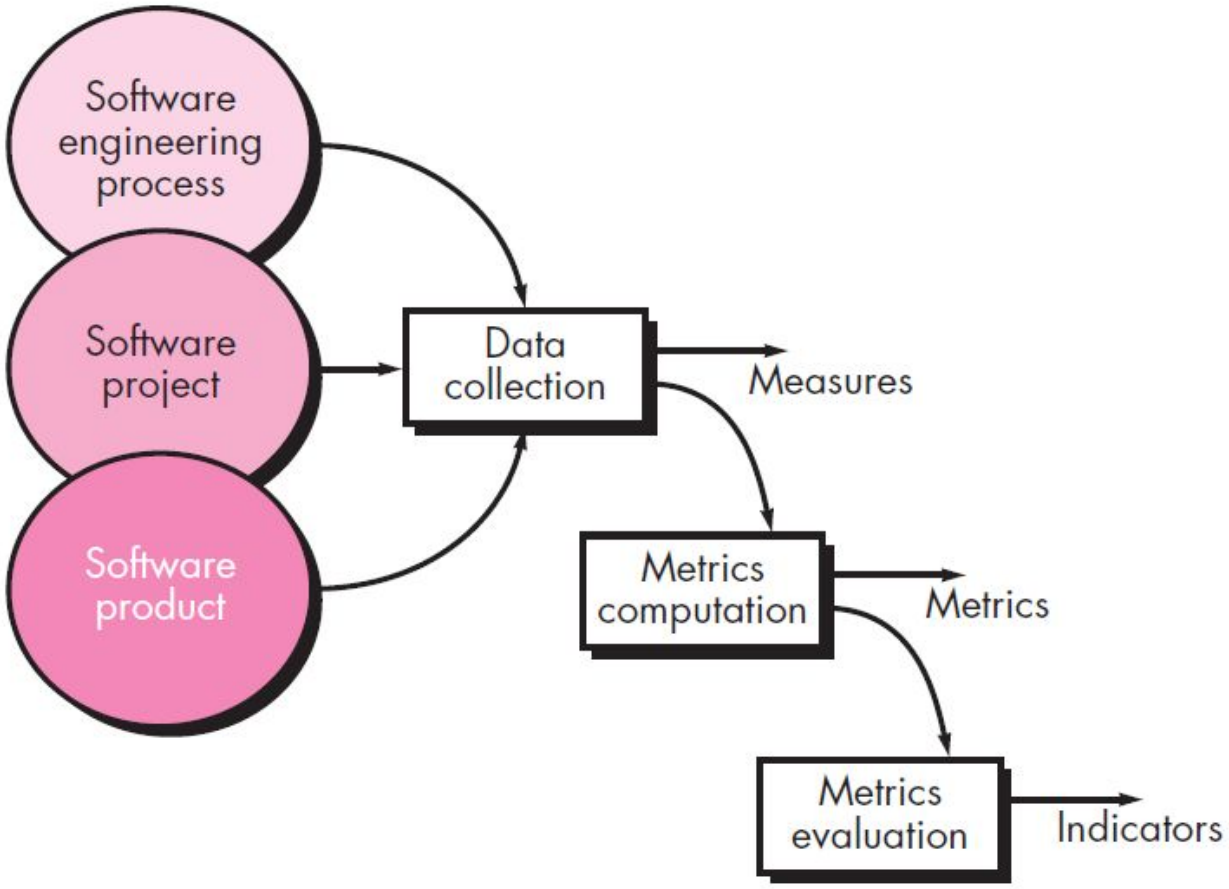
C++ = 64

aKLOC = FP * LOC_per_FP / 1000

It means for the SpellChekcer Example: (Java)

LOC=52.25*53=2769.25 LOC or 2.76 KLOC

Software Metrics Collection Process



Project and Process Metrics

Objective: To assist in the definition, collection, evaluation, and reporting of software measures and metrics.

Mechanics: Each tool varies in its application, but all provide mechanisms for collecting and evaluating data that lead to the computation of software metrics.

Representative Tools: [7]

Function Point WORKBENCH, developed by Charismatek (**www.charismatek.com.au**), offers a wide array of FP-oriented metrics.

MetricCenter, developed by Distributive Software (**www.distributive.com**), supports automating data collection, analysis, chart formatting, report generation, and other measurement tasks.

PSM Insight, developed by Practical Software and Systems Measurement (**www.psmsc.com**), assists in the creation and subsequent analysis of a project measurement database.

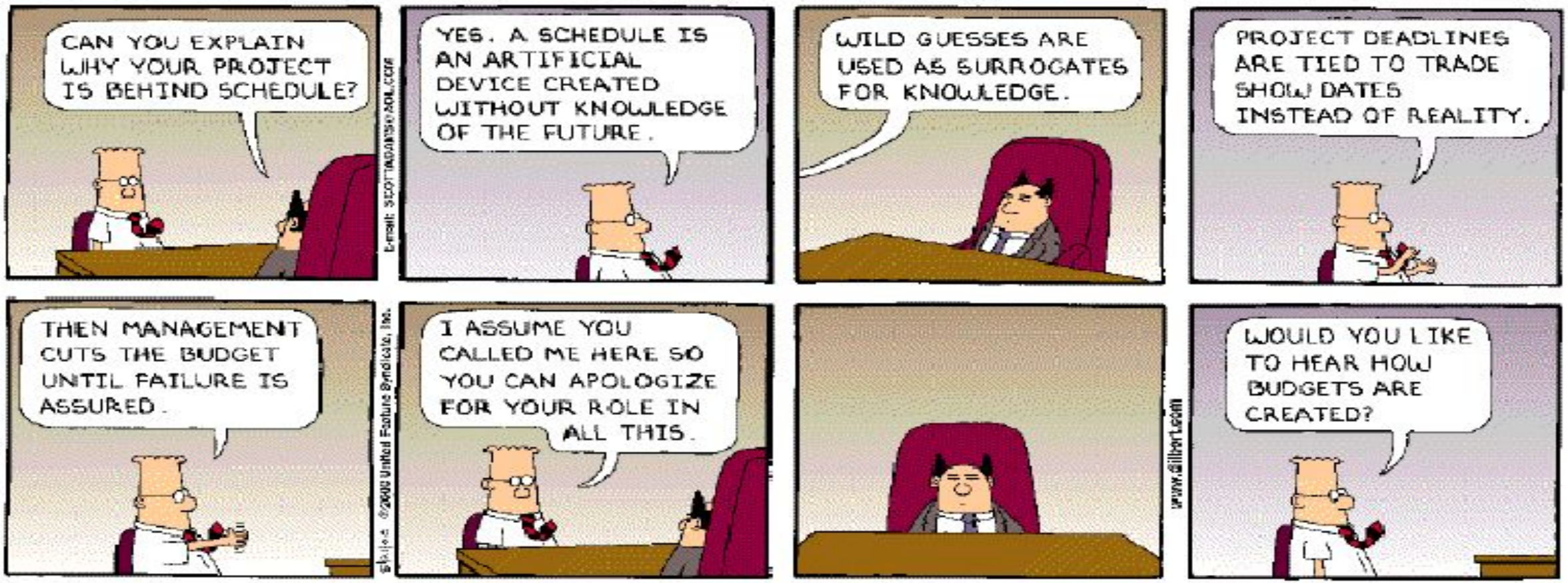
SLIM tool set, developed by QSM (**www.qsm.com**), provides a comprehensive set of metrics and estimation tools.

SPR tool set, developed by Software Productivity Research (**www.spr.com**), offers a comprehensive collection of FP-oriented tools.

TychoMetrics, developed by Predicate Logic, Inc. (**www.predicate.com**), is a tool suite for management metrics collection and reporting.

DILBERT®

BY SCOTT ADAMS



Estimation for Software Project

- Estimation begins with a description of the scope of the problem.
- The problem is then decomposed into a set of smaller problems, and each of these is estimated using historical data and experience as guides.
- Problem complexity and risk are considered before a final estimate is made.
- Process and project metrics can provide historical perspective and powerful input for the generation of quantitative estimates.
- Estimation of **resources, cost, and schedule** for a software engineering effort requires experience, access to good historical information (metrics).
- *Project complexity* has a strong effect on the uncertainty inherent in planning.

Cost of a project

- The cost in a project is due to:
  - the requirements for software, hardware and human resources
  - the cost of software development is due to the human resources needed
  - most cost estimates are measured in ***person-months (PM)***

Software Cost Estimation

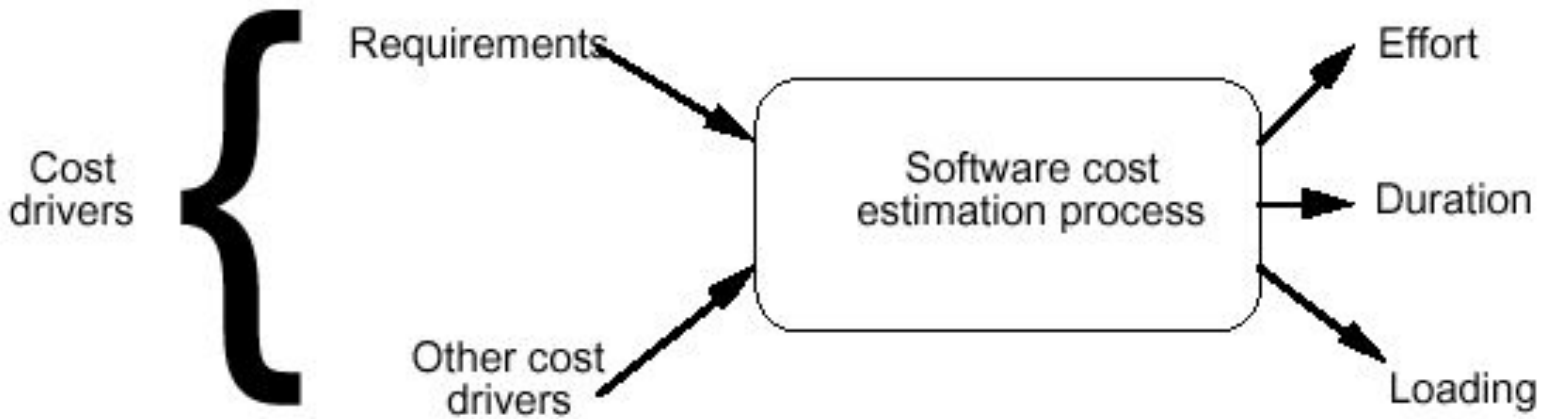


Figure 1. Classical view of software estimation process.

Effort

- Effort Equation
  - **PM = C * (KDSI)^{n} (**person-months)
    - where **PM** = number of person-month (=152 working hours),
    - **C** = a constant,
    - **KDSI** = thousands of "delivered source instructions" (DSI) and
    - **n** = a constant.

Productivity

- Productivity equation
  - **(DSI) / (PM)**
    - where **PM** = number of person-month (=152 working hours),
    - **DSI** = "delivered source instructions"

Schedule

- Schedule equation
  - **TDEV = C * (PM)^{n} (**months)
    - where TDEV = number of months estimated for software development.

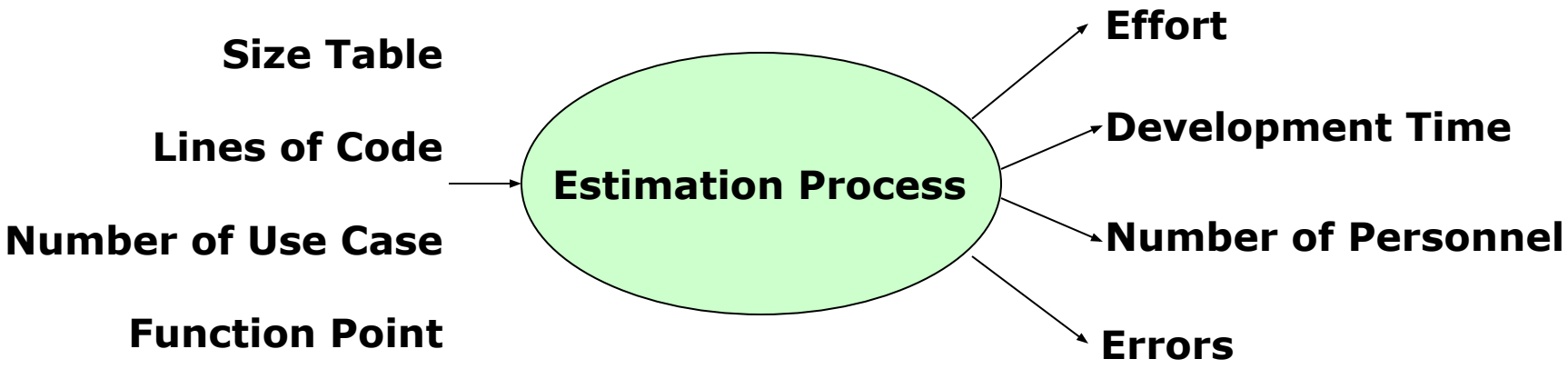
Average Staffing

- Average Staffing Equation
  - **(PM) / (TDEV)** **(**FSP)
    - where FSP means Full-time-equivalent Software Personnel.

Cost Estimation Process

Cost=SizeOfTheProject x Productivity

Cost Estimation Process



Project Size - Metrics

1. **Number of functional requirements**
2. **Cumulative number of functional and non-functional requirements**
3. **Number of Customer Test Cases**
4. **Number of ‘typical sized’ use cases**
5. **Number of inquiries**
6. **Number of files accessed (external, internal, master)**
7. **Total number of components (subsystems, modules, procedures, routines, classes, methods)**
8. **Total number of interfaces**
9. **Number of System Integration Test Cases**
10. **Number of input and output parameters (summed over each interface)**
11. **Number of Designer Unit Test Cases**
12. **Number of decisions (if, case statements) summed over each routine or method**
13. **Lines of Code, summed over each routine or method**

Introduction to COCOMO models

- The **COstructive COst Model** (COCOMO) is the most widely used software estimation model.
- The COCOMO model predicts the **effort** and **duration** of a project based on inputs relating to the size of the resulting systems and a number of "**cost drives**" that affect productivity.

COCOMO Models

- COCOMO is defined in terms of three different models:
  - the **Basic model**,
  - the **Intermediate model**, and
  - the **Detailed model**.
- The more complex models account for more factors that influence software projects, and make more accurate estimates.

The Development mode

- The most important factors contributing to a project's duration and cost is the Development Mode
  - **Organic Mode:** The project is developed in a familiar, stable environment, and the product is similar to previously developed products. The product is relatively small, and requires little innovation.
  - **Semidetached Mode:** The project's characteristics are intermediate between Organic and Embedded.
  - **Embedded Mode:** The project is characterized by tight, inflexible constraints and interface requirements. An embedded mode project will require a great deal of innovation.

Effort Computation

- The **Basic COCOMO model** computes effort as a function of program size. The Basic COCOMO equation is:
  - ***E = aKLOC^b***
- Effort for three modes of Basic COCOMO.

Mode	**a**	**b**
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

Example

Mode	Effort Formula
Organic	$E = 2.4 * (S^{1.05})$
Semidetached	$E = 3.0 * (S^{1.12})$
Embedded	$E = 3.6 * (S^{1.20})$

Size = 200 KLOC

Effort = $a * Size^{b}$

Organic — $E = 2.4 * (200^{1.05}) = 626$ staff-months

Semidetached — $E = 3.0 * (200^{1.12}) = 1133$ staff-months

Embedded — $E = 3.6 * (200^{1.20}) = 2077$ staff-months

Effort Computation

- The **intermediate COCOMO model** computes effort as a function of program size and a set of cost drivers. The Intermediate COCOMO equation is:
  - ***E = aKLOC^b*EAF***
- Effort for three modes of intermediate COCOMO.

Mode	**a**	**b**
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Effort computation(.)

- **Effort Adjustment Factor**

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
Required Reliability	.75	.88	1.00	1.15	1.40	1.40
Database Size	.94	.94	1.00	1.08	1.16	1.16
Product Complexity	.70	.85	1.00	1.15	1.30	1.65
Execution Time Constraint	1.00	1.00	1.00	1.11	1.30	1.66
Main Storage Constraint	1.00	1.00	1.00	1.06	1.21	1.56
Virtual Machine Volatility	.87	.87	1.00	1.15	1.30	1.30
Comp Turn Around Time	.87	.87	1.00	1.07	1.15	1.15
Analyst Capability	1.46	1.19	1.00	.86	.71	.71
Application Experience	1.29	1.13	1.00	.91	.82	.82
Programmers Capability	1.42	1.17	1.00	.86	.70	.70
Virtual machine Experience	1.21	1.10	1.00	.90	.90	.90
Language Experience	1.14	1.07	1.00	.95	.95	.95
Modern Prog Practices	1.24	1.10	1.00	.91	.82	.82
SW Tools	1.24	1.10	1.00	.91	.83	.83
Required Dev Schedule	1.23	1.08	1.00	1.04	1.10	1,10

Effort Computation (..)

Total EAF = Product of the selected factors

Adjusted value of Effort: Adjusted Person Months:

APM = (Total EAF) * PM

Example

	Organic	Semidetached	Embedded
a	3.2	3.0	2.8
b	1.05	1.12	1.20

Mode	Effort Formula
Organic	$E = 3.2 * (S^{1.05}) * C$
Semidetached	$E = 3.0 * (S^{1.12}) * C$
Embedded	$E = 2.8 * (S^{1.20}) * C$

---

e.g. Size = 200 KLOC

Effort = a * Size^{b} * C

Cost drivers:

Low reliability .88

High product complexity 1.15

Low application experience 1.13

High programming language experience .95

C = .88 * 1.15 * 1.13 * .95 = 1.086

Organic — $E = 3.2 * (200^{1.05}) * 1.086 = 906$ staff-months

Semidetached — $E = 3.0 * (200^{1.12}) * 1.086 = 1231$ staff-months

Embedded — $E = 2.8 * (200^{1.20}) * 1.086 = 1755$ staff-months

Software Development Time

- **Development Time Equation Parameter Table:**

Parameter	Organic	Semi-detac hed	Embedded
C	2.5	2.5	2.5
D	0.38	0.35	0.32

Development Time, **TDEV = C * (APM **D)**

Number of Personnel, **NP = APM / TDEV**

Distribution of Effort

- A development process typically consists of the following stages:
  - **Requirements Analysis**
  - **Design (High Level + Detailed)**
  - **Implementation & Coding**
  - **Testing (Unit + Integration)**

Distribution of Effort (.)

The following table gives the recommended **percentage distribution of Effort (APM)** and **TDEV** for these stages:

Percentage Distribution of Effort and Time Table:

	Req Analysis	Design, HLD + DD	Implementation	Testing	
Effort	**23%**	**29%**	**22%**	**21%**	**100%**
TDEV	**39%**	**25%**	**15%**	**21%**	**100%**

Error Estimation

- Calculate the estimated number of errors in your design, i.e.total errors found in requirements, specifications, code, user manuals, and bad fixes:
  - Adjust the **Function Point** calculated in step1

$$AFP = FP ** 1.25$$

  - Use the following table for calculating error estimates

Error Type	Error / AFP
Requirements	1
Design	1.25
Implementation	1.75
Documentation	0.6
Due to Bug Fixes	0.4

All Together