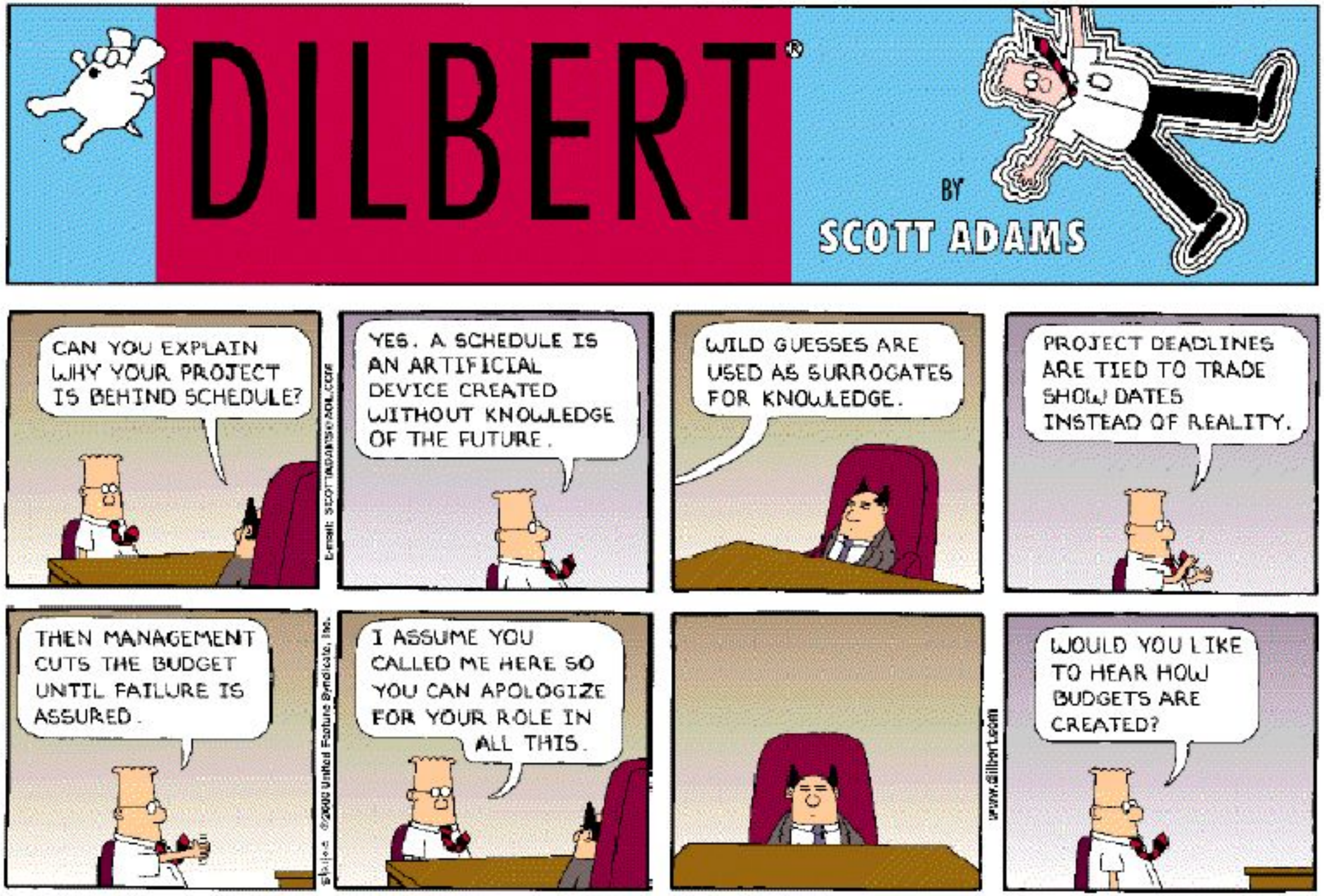
COCOMO Models

Project Management and Mr. Murphy

1. Logic is a systematic method of coming to the wrong conclusion with confidence.
2. Technology is dominated by those who manage what they do not understand.
3. Nothing ever gets done on schedule or within budget.
4. If mathematically you end up with the incorrect answer, try multiplying by the page number.

Mr. Murphy was an optimist


DILBERT
BY
SCOTT ADAMS
CAN YOU EXPLAIN WHY YOUR PROJECT IS BEHIND SCHEDULE?
YES. A SCHEDULE IS AN ARTIFICIAL DEVICE CREATED WITHOUT KNOWLEDGE OF THE FUTURE.
WILD GUESSES ARE USED AS SURROGATES FOR KNOWLEDGE.
PROJECT DEADLINES ARE TIED TO TRADE SHOW DATES INSTEAD OF REALITY.
THEN MANAGEMENT CUTS THE BUDGET UNTIL FAILURE IS ASSURED.
I ASSUME YOU CALLED ME HERE SO YOU CAN APOLOGIZE FOR YOUR ROLE IN ALL THIS.
WOULD YOU LIKE TO HEAR HOW BUDGETS ARE CREATED?

Motivation

The software cost estimation provides:

- The vital link between the general concepts and techniques of **economic analysis** and the particular world of **software engineering.**
- Software cost estimation techniques also provides an essential part of the foundation for **good software management**.

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)***

Cost of a project (.)

- the cost of the project depends on the nature and characteristics of the project,
- at any point, the accuracy of the estimate will depend on the amount of reliable information we have about the final product.

Software Cost Estimation

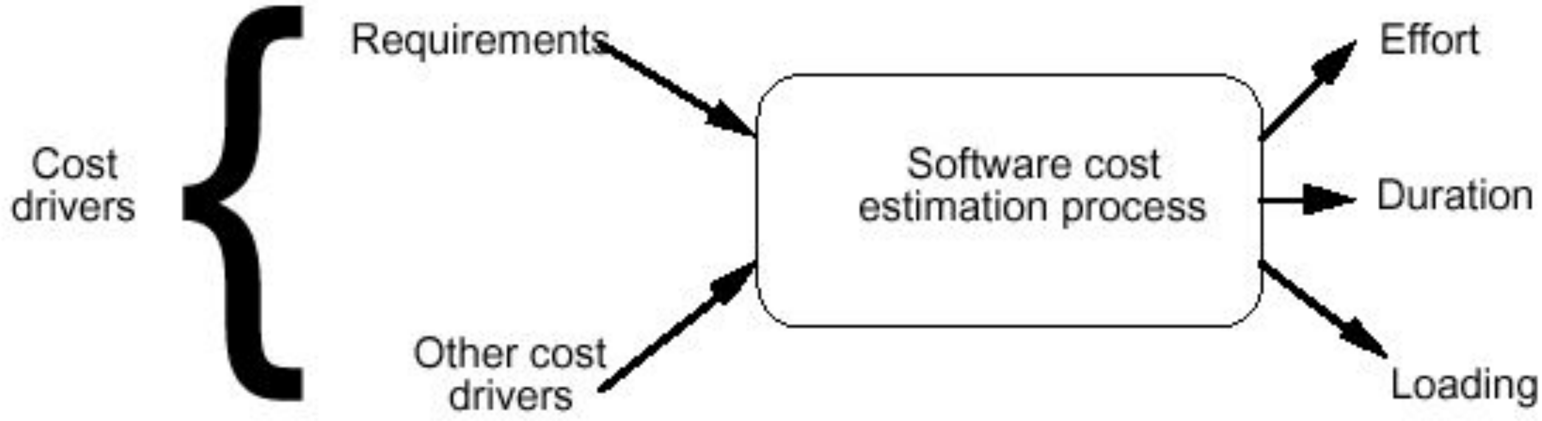


Figure 1. Classical view of software estimation process.

Effort

- Effort Equation
  - **PM = C * (KDSI)$^{\mathbf{n}}$ (**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.

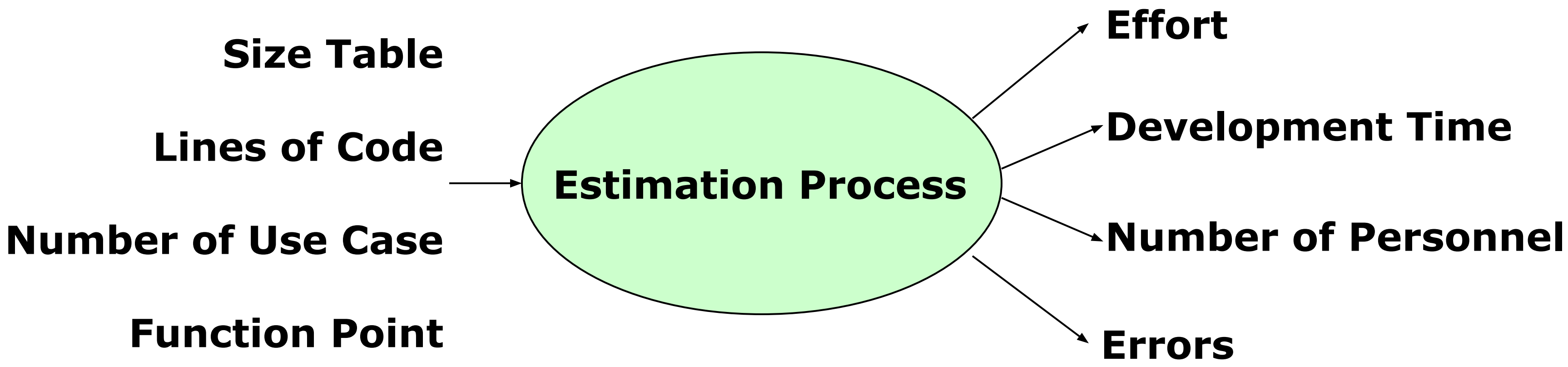
Average Staffing

- 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**

Project Size – Metrics(.)

<u>Availability of Size Estimation Metrics:</u>

	Development Phase	Available Metrics
a	Requirements Gathering	1, 2, 3
b	Requirements Analysis	4, 5
d	High Level Design	6, 7, 8, 9
e	Detailed Design	10, 11, 12
f	Implementation	12, 13

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

Function Points(..)

- **<u>STEP 2</u>:** Multiply each number by a weight factor, according to complexity (**simple**, **average** or **complex**) of the parameter, associated with that number. The value is given by a table:

Parameter	simple	average	complex
users inputs	3	4	6
users outputs	4	5	7
users requests	3	4	6
files	7	10	15
external interfaces	5	7	10

Function Points(...)

- **<u>STEP 3</u>:** Calculate the total **UFP** (Unadjusted Function Points)
- **<u>STEP 4</u>:** Calculate the total **TCF** (Technical Complexity Factor) by giving a value between 0 and 5 according to the importance of the following points (next slide):

Function Points(....)

Technical Complexity Factors:

1. **Data Communication**
2. **Distributed Data Processing**
3. **Performance Criteria**
4. **Heavily Utilized Hardware**
5. **High Transaction Rates**
6. **Online Data Entry**
7. **Online Updating**
8. **End-user Efficiency**
9. **Complex Computations**
10. **Reusability**
11. **Ease of Installation**
12. **Ease of Operation**
13. **Portability**
14. **Maintainability**

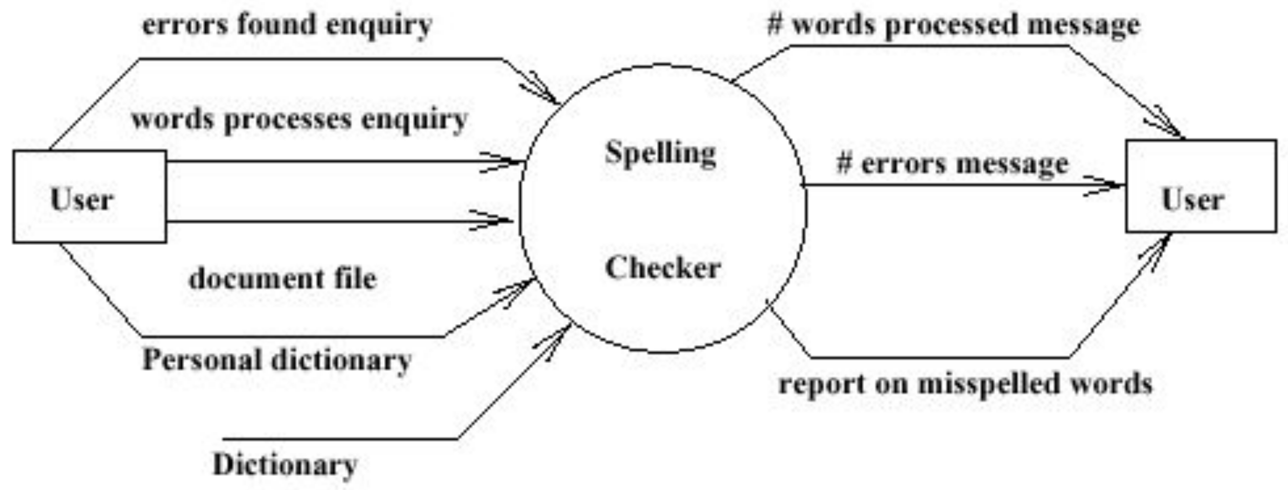
Function Points(.....)

- **<u>STEP 5</u>:** Sum the resulting numbers too obtain **DI** (degree of influence)
- **<u>STEP 6</u>: TCF** (Technical Complexity Factor) by given by the formula
  - ***TCF=0.65+0.01*DI***
- **<u>STEP 6</u>:** Function Points are by given by the formula
  - ***FP=UFP*TCF***

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).

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

Example (..)

- **<u>Technical Complexity Factors:</u>**
  - 1. Data Communication 3
  - 2. Distributed Data Processing 0
  - 3. Performance Criteria 4
  - 4. Heavily Utilized Hardware 0
  - 5. High Transaction Rates 3
  - 6. Online Data Entry 3
  - 7. Online Updating 3
  - 8. End-user Efficiency 3
  - 9. Complex Computations 0
  - 10. Reusability 3
  - 11. Ease of Installation 3
  - 12. Ease of Operation 5
  - 13. Portability 3
  - 14. Maintainability 3
    - » **DI =30(Degree of Influence)**

Example (…)

- Function Points
  - FP=UFP*(0.65+0.01*DI)= 55*(0.65+0.01*30)=52.25

  - That means **FP=52.25**

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

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.

Modes

Feature	**Organic**	**Semidetached**	**Embedded**
Organizational understanding of product and objectives	Thorough	Considerable	General
Experience in working with related software systems	Extensive	Considerable	Moderate
Need for software conformance with pre-established requirements	Basic	Considerable	Full
Need for software conformance with external interface specifications	Basic	Considerable	Full

Modes (.)

Feature	Organic	Semidetached	Embedded
Concurrent development of associated new hardware and operational procedures	Some	Moderate	Extensive
Need for innovative data processing architectures, algorithms	Minimal	Some	Considerable
Premium on early completion	Low	Medium	High
Product size range	<50 KDSI	<300KDSI	All

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:

<u>Percentage Distribution of Effort and Time Table:</u>

	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