TOPIC – 4
Empirical Estimation Techniques

Empirical Estimation Models

- An estimation model for computer software uses **empirically derived formulas** to **predict effort as a function of LOC** or **FP**.
- The empirical data that support most estimation models are derived from a limited sample of projects.
- The model should be **tested by applying data collected from completed projects**.
- **Plugging the data into the model**.
- Then **comparing actual to predicted results.**
- If agreement is poor, the model must be tuned and retested before it can be used.

The Structure of Estimation Models

- A typical estimation model is **derived using regression analysis** on **data collected** from past software projects.
- Overall : $E = A + B \times (e_v)^C$).
- where,
  - A,B and C are empirically derived constants.
  - E is effort in person-months
  - e_v is the estimation variable either **LOC (or) FP**

- LOC-Oriented estimation models in the literature as follows :
  - many LOC-oriented estimation models proposed in the literature are

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

- FP-Oriented estimation models in the literature as follows :

FP-oriented models have also been proposed. These include

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

COCOMO-II Model

COnstructive **CO**st **MO**del (**COCOMO**) II is actually a hierarchy of estimation models that address the following areas.

- **Application Composition Model.**
  - Used **during the early stages of software engineering**
  - When **prototyping of user interfaces**
  - Consideration of **Software and System Interaction**
  - Assessment of **Performance**.
- **Early Design Stage Model.**
  - Used **once requirements have been stabilized** and
  - Basic **software architecture has been established**.
- **Post-Architecture-Stage Model.**
  - Used **during the construction of the software**.

- Like all estimation models for software, **COCOMO II** model require sizing information.
- **Three different sizing options** are available.
  - **Object Point**
  - **Function Point and**
  - **Lines of Source Code**
-

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

- **Object Point**
  - Indirect software measure that is computed using counts of the number of,
    - Screens, Reports and Components likely to be required to build the application

Object point count is then determined by **multiplying the original number of object instances by the weighting factor**.

$$NOP = (\text{object points}) \times [(100 - \%\text{reuse})/100]$$

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP value, a "productivity rate" must be derived. Figure 26.7 presents the productivity rate

$$PROD = \frac{NOP}{\text{person-month}}$$

for different levels of developer experience and development environment maturity.

Once the productivity rate has been determined, an estimate of project effort is computed using

$$\text{Estimated effort} = \frac{NOP}{PROD}$$

FIGURE 26.7 Productivity rate for object points.

Source: [Boe96].

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

Software Equation

- Software equation is a **dynamic multivariable model** that assumes a specific distribution of effort over the life of **Software Development Project**.

- The model has been derived from **productivity data** collected for over **4000 contemporary software projects**.

$$E = \frac{\text{LOC} \times B^{0.333}}{P^3} \times \frac{1}{t^4} \qquad (26.4)$$

where

E = effort in person-months or person-years

t = project duration in months or years

B = "special skills factor"[13]

P = "productivity parameter" that reflects: overall process maturity and management practices, the extent to which good software engineering practices are used, the level of programming languages used, the state of the software environment, the skills and experience of the software team, and the complexity of the application