COCOMO MODEL

RAILWAY TIME TRACKING AND PREDICTION SYSTEM

**Aim:** To estimate the cost of the Railway Time Tracking and Prediction System by using COCOMO Model.

**Description**: COCOMO (Constructive Cost Model) is a procedural cost estimate model based on LOC, i.e., the number of Lines of Code, used for software projects to predict various project parameters such as size, effort, cost, time, and quality reliably. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, making it one of the best-documented models. The key parameters that define the quality of any software product, which are also an outcome of the COCOMO, are primarily Effort Schedule.

**Effort**: Amount of labor that will be required to complete a task. It is measured in person months units.

Schedule: The amount of time required for the completion of the job, which is proportional to the effort put in. It is measured in units of time such as weeks or months.

The Railway Time Tracking and Prediction System can be classified as a semidetached or embedded system, depending on the complexity and experience required. As it involves high levels of complexity and creativity, a larger team size and experienced developers will be necessary.

Types of Models: COCOMO consists of three increasingly detailed and accurate forms that can be adopted according to requirements. The three types of COCOMO models are:

1. Basic COCOMO Model
2. Intermediate COCOMO Model
3. Detailed COCOMO Model

The first level, Basic COCOMO, can be used for quick and slightly rough calculations of software costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations. Intermediate COCOMO takes these Cost Drivers into account, and Detailed COCOMO additionally accounts for the influence of individual project phases, i.e., in the case of Detailed COCOMO, it accounts for both these cost drivers, and calculations are performed phase-wise, producing a more accurate result.

**Estimation of Effort**:

Calculations – Basic Model – E= a(KLOC)b

time = c(Effort)d

Person required = Effort/time

The above formula is used for the cost estimation of the Basic COCOMO model and is used in subsequent calculations. The effort is measured in Person-Months and is dependent on Kilo-Lines of code. The development time is measured in months. These formulas are used as such in the Basic Model calculations as not much consideration of different factors such as reliability, expertise is taken into account; hence, the estimate is rough.

| Project | $a_i$ | $b_i$ | $c_i$ | $d_i$ |
|---|---|---|---|---|
| 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 |

**Intermediate Model** – The Basic COCOMO model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, capability, etc., are used as Cost Drivers. The Intermediate Model utilizes 15 such drivers for cost estimation.

**Detailed Model** - The Detailed COCOMO incorporates all characteristics of the Intermediate Version with an assessment of the cost driver's impact on each step of the software engineering process. The Detailed Model uses different effort multipliers for each cost driver attribute. In Detailed COCOMO, the whole software is divided into different modules, and then we apply COCOMO in different modules to estimate effort and then sum the effort. The six phases of Detailed COCOMO are:

1. Planning and requirements

2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Maintenance

Each phase of the software development process is assigned effort multipliers based on different cost driver attributes, which are then used to calculate the overall effort required for the project

**Estimation of Effort**: Estimation of effort involves the calculation of the amount of time and resources required to develop the Railway Time Tracking and Prediction System. This process takes into account various factors such as the size and complexity of the project, the skills and experience of the development team, and the tools and technologies that will be used.

**Cost Drivers and Their Attributes**: Cost drivers are the factors that influence the overall cost of developing the Railway Time Tracking and Prediction System. These drivers can be classified into various categories such as product attributes, hardware attributes, personnel attributes, and project attributes. Some of the attributes of these cost drivers include the size and complexity of the system, the quality of the hardware and software components, the experience and skills of the development team, and the level of project management.

**Product Attributes**: Product attributes are the characteristics of the Railway Time Tracking and Prediction System that affect the overall development cost. These attributes include the size and complexity of the system, the number of features and functions, the level of reliability and maintainability, and the level of user interaction.

**Hardware Attributes:** Hardware attributes are the characteristics of the hardware that will be used in the development of the Railway Time Tracking and Prediction System. These attributes include the processing power, memory, storage capacity, and network connectivity of the hardware.

**Personnel Attributes**: Personnel attributes are the characteristics of the development team that will be working on the Railway Time Tracking and Prediction System. These attributes include the experience and skills of the team members, their level of education and training, and their familiarity with the tools and technologies that will be used in the project.

**Project Attributes:** Project attributes are the characteristics of the development project that affect the overall development cost. These attributes include the size and complexity of the project, the level of project management, the level of documentation required, and the level of collaboration and communication among team members.

**Detailed COCOMO:** The Detailed COCOMO model is a software cost estimation model that takes into account various factors such as the size and complexity of the system, the level of documentation required, the experience and skills of the development team, and the tools and technologies that will be used. The model provides a detailed breakdown of the cost of developing the Railway Time Tracking and Prediction System.

**YOUR BASIC COCOMO RESULTS!!**

| MODE | "A" variable | "B" variable | "C" variable | "D" variable | KLOC | EFFORT, (in person/months) | DURATION, (in months) | STAFFING, (recommended) |
|---|---|---|---|---|---|---|---|---|
| embedded | 3.6 | 1.2 | 2.5 | 0.32 | 3 | 13.453894147847587 | 5.743430507902744 | 2.342484013576126 |

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

**effort** =a*KLOC$^{b}$, in person/months, with KLOC = lines of code, (in the thousands), and:

**duration** =c*effort$^{d}$, finally:

**staffing** =effort/duration

For further reading, see Boehm, "Software Engineering Econimics",(81)

**WARNING:** If you see "NaN" in any field above, you have entered an **INVALID** value for KLOC!! Hit the "BACK" button on your browser, hit the "RESET" button, and enter a **DECIMAL NUMBER** in the KLOC input text box!

***Thank you***, and happy software engineering!

---

**Phases of Detailed COCOMO:**

The phases of Detailed COCOMO include the following:

1. **Planning and Requirements**: This phase involves gathering and analyzing the requirements for the Railway Time Tracking and Prediction System.
2. **Design**: This phase involves designing the system architecture, modules, and components.
3. **Implementation**: This phase involves coding and unit testing the system components.
4. **Integration and Testing**: This phase involves integrating the system components and testing the system as a whole.
5. **Maintenance:** This phase involves maintaining and supporting the system after it has been deployed.