Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Academic Year : 2024-25

Experiment No. 7
Apply COCOMO Model for cost estimation
Name: Krisha Chikka
Std/Div: TE/1 Roll no. : 30
Date of Performance: 02/09/2024
Date of Submission: 23/09/2024

Aim: To apply COCOMO Model for cost estimation.

Objective: To evaluate and apply COCOMO Model for Currency Detector for the Visually Impaired for cost estimation.

Theory:

Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981.COCOMO is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

The necessary steps in this model are:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
2. Determine a set of 15 multiplying factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size. To determine the initial effort E_i in person-months the equation used is of the type is shown below

E_i=a*(KDLOC)b.

In Software Engineering, there are three stages of the COCOMO Model estimation technique are:
i. Basic COCOMO Model
ii. Intermediate COCOMO Model
iii. Complete COCOMO Model

i. **Basic COCOMO Model:**

Boehm postulated that any software development project. It classifies into one of the following three categories based on the development complexity –
i. Organic
ii. Semi-detached
iii. Embedded

Organic:

We can classify a development project to be of the organic type if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.

Semidetached:

A development project classifies the semidetached type if the development team. It consists of a mixture of experienced and inexperienced staff. Team members may have limited experience in related systems but may be unfamiliar with some aspects of the system being developed.

Embedded:

A development project considers the embedded type, if the software being developed is strongly coupled to hardware, or if stringent regulations on the operational procedures exist. Team members may have limited experience in related systems but they may be unfamiliar with some aspects of the system being developed.

ii. Intermediate COCOMO Model:

The basic COCOMO model assumes that effort and development time are functions of the product size alone. However, a host of other project parameters besides the product size affect the effort as well as the time required to develop the product.

It uses a set of 15 cost drivers (multipliers) that determines based on various attributes of software development. These cost drivers multiply with the initial cost and effort estimates to appropriately scale those up or down.

In general, Boehm identifies the cost drivers. He also classifies as being attributes of the following items:

Product: The characteristics of the product that considers include the inherent complexity of the product, reliability requirements of the product, etc.

Computer: The Characteristics of the computer that considers include the execution speed required, storage space required, etc.
Personnel: The attributes of development personnel that considers include the experience level of personnel, their programming capability, analysis capability, etc.
Development environment: These attributes capture the development facilities available to the developers. An important parameter that considers the sophistication of the automation (CASE) tools are uses for software development.

iii. Complete COCOMO model:

A major shortcoming of both the basic and the intermediate COCOMO models is that they consider a software product as a single homogeneous entity. However, most large systems are made up of several smaller sub-systems. These sub-systems often have widely different characteristics.

The **Complete COCOMO model** considers these differences in characteristics of the subsystems and estimates the effort and development time as the sum of the estimates for the individual sub-systems.
In other words, the cost to develop each sub-system estimates separately. The complete system cost determines as the subsystem costs. This approach reduces the margin of error in the final estimate.

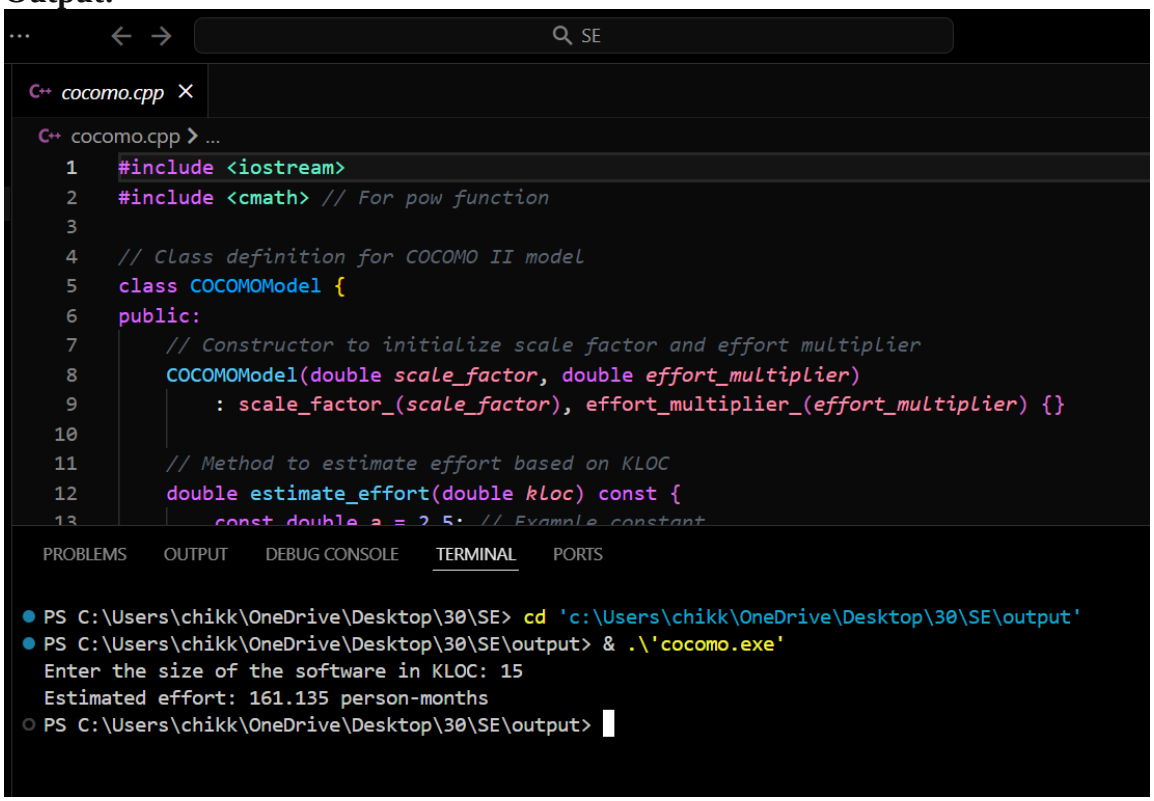
Solution:

Code:

```
#include <iostream>
#include <cmath> // For pow function

// Class definition for COCOMO II model
class COCOMOModel {
public:
    // Constructor to initialize scale factor and effort multiplier
    COCOMOModel(double scale_factor, double effort_multiplier)
        : scale_factor_(scale_factor),
effort_multiplier_(effort_multiplier) {}

    // Method to estimate effort based on KLOC
    double estimate_effort(double kloc) const {
        const double a = 2.5; // Example constant
        const double b = 1.2; // Example exponent

        // Effort Adjustment Factor (EAF) = scale factor * effort
multiplier
        double eaf = scale_factor_ * effort_multiplier_;

        // Calculate the effort
        return (a * std::pow(kloc, b)) * eaf;
```

```
    }

private:
    double scale_factor_;
    double effort_multiplier_;
};

int main() {
    double kloc;                // Size of the software in KLOC
(thousands of lines of code)
    double scale_factor;       // Scale factor derived from cost drivers
    double effort_multiplier; // Effort multiplier derived from cost
drivers

    // Get inputs from user
    std::cout << "Enter the size of the software in KLOC: ";
    std::cin >> kloc;

    // Here we use example values for scale factor and effort
multiplier.
    // In a real application, these values should be calculated based
on cost drivers.
    scale_factor = 1.0; // This is a placeholder value
    effort_multiplier = 2.5; // This is a placeholder value

    // Create an instance of COCOMOModel
    COCOMOModel cocomo(scale_factor, effort_multiplier);

    // Estimate effort
    double effort = cocomo.estimate_effort(kloc);

    // Output the estimated effort
    std::cout << "Estimated effort: " << effort << " person-months\n";

    return 0;
}
```

Output:



Conclusion: We have successfully applied COCOMO Model for Currency Detector for the Visually Impaired.In this experiment, we successfully applied the COCOMO Model to estimate the development effort for the Currency Detector for the Visually Impaired. By analyzing key parameters such as the size of the software in KLOC and using appropriate scale factors and effort multipliers, we were able to predict the required person-months for development. This process demonstrated the effectiveness of the COCOMO Model in providing a reliable cost estimation for software projects, ensuring accurate planning and resource allocation.