COCOMO-1

SPARSH ACHARYA

23FE10CAI00367

Overview

COCOMO stands for the **Constructive Cost Model** and was developed by Barry Boehm in 1981. It is an empirical algorithmic model that estimates software development effort, schedule, and staffing based on the estimated size of the software (typically expressed in KLOC—thousands of lines of code) and other project attributes.

Project Types

COCOMO 1 distinguishes three project types based on size, complexity, environment, and team experience. These types help adjust the estimation coefficients.
They are:

- Organic Projects
- semi-Detached Projects
- Embedded Projects

Organic Project

- **Characteristics:**
  - Small, relatively simple projects.
  - Teams are small and experienced.
  - Problems are well understood.
  - Requirements are stable and less rigid.
- **KLOC Range:** Approximately 2 to 50 KLOC.
- **Examples:**
  - Simple business applications, basic inventory systems, or data processing systems.

Semi-Detached Project

- **Characteristics:**
  - Intermediate complexity and size.
  - Teams are mixed in experience.
  - Requirements may be partially rigid and may require moderate innovation.
- **KLOC Range:** Approximately 50 to 300 KLOC.
- **Examples:**
  - Database management systems, transaction processing systems, or moderately complex inventory applications.

Embedded Projects

- **Characteristics:**
  - High complexity with stringent constraints (hardware, software, real-time requirements).
  - Larger teams often required.
  - Often involve safety-critical or mission-critical systems.
- **KLOC Range:** Typically more than 300 KLOC.
- **Examples:**
  - Air traffic control systems, real-time embedded control systems (such as ATMs or automotive control systems).

Estimation Models

COCOMO 1 comes in three “levels” or forms which offer progressively more detailed estimation:

- Basic
- Intermediate
- Detailed

Basic Model

The Basic COCOMO model estimates effort solely as a function of the project size (KLOC) using a simple power-law equation:

- **Efforts:**

$$E = a \times (KLOC)^b \quad PM$$

- **Development Time:**

$$D = c \times E^d \quad M$$

- **Average Staff Size:**

$$staff = {}^{E}/_{D} \qquad P$$

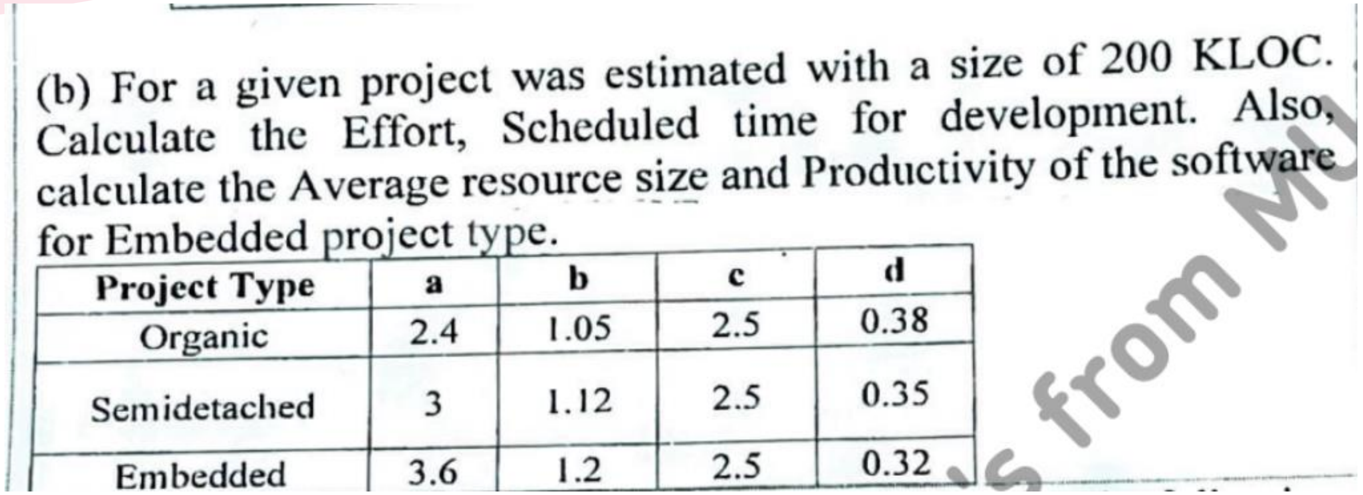
- **Productivity:**

$$Prod = {}^{KLOC}/_{E}$$

Coeff of Basic Model

Project Type	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Numerical 1



(b) For a given project was estimated with a size of 200 KLOC. Calculate the Effort, Scheduled time for development. Also, calculate the Average resource size and Productivity of the software for Embedded project type.

Project Type	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Solution

- Given
  KLOC = 200
  Proj Type = Embedded

  so a = 3.6, b = 1.2, c = 2.5, d = 0.32

  substituting in equations

- **Efforts:**

$$E = a \times (KLOC)^{b} = 3.6 \times (200)^{1.2} = 2077.48\ PM$$

- **Development Time:**

$$D = c \times E^{d} = 2.5 \times 2077.48^{0.32} = 28.81\ M$$

Solution

- **Average Staff Size:**

$$staff = {}^{E}/_{D} = {}^{2077.48}/_{28.81} = 72.11\,P$$

- **Productivity:**

$$Prod = {}^{KLOC}/_{E} = {}^{200}/_{2077.48} = 0.096\,{}^{KLOC}/_{PM}$$

Intermediate COCOMO Model

Intermediate COCOMO extends the basic model by incorporating 15 cost drivers (also called “effort multipliers”) that adjust the nominal effort estimate. These drivers reflect factors related to the product, hardware, personnel, and project attributes.

Efforts:

$$E = a \times (KLOC)^b \times EAF \quad PM$$

Coeff of Intermediate Model

Project Type	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-Organic	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Cost Drivers

There are 15 cost drivers , divided into four categories:

- Product Attribute
- Hardware Attribute
- Personal Attribute
- Project Attribute

Each cost driver is rated (e.g., Very Low, Low, Nominal, High, Very High, Extra High) and has a corresponding multiplier. The product of all these multipliers yields the EAF, which typically ranges from about 0.9 to 1.4.

Product Attributes:

- Required software reliability (RELY)
- Database size (DATA)
- Product complexity (CPLX)

Hardware Attributes:

- Execution time constraints (TIME)
- Main storage constraints (STOR)
- Virtual machine volatility (VIRT)
- Turnaround time (TURN)

Personnel Attributes:

- Analyst capability (ACAP)
- Software engineering capability (AEXP/PCAP)
- Programmer capability (PCAP)
- Virtual machine experience (VEXP)
- Programming language experience (LEXP)

Project Attributes:

- Use of modern programming practices (MODP)
- Use of software tools (TOOL)
- Required development schedule (SCED)

Cost Driver	Very Low	Low	Nominal	High	Veri High	Extra High
RELY	0.75	0.88	1	1.15	1.40	-
DATA	-	0.94	1	1.08	1.16	-
CPLX	0.7	0.85	1	1.15	1.30	1.65
TIME	-	-	1	1.11	1.30	1.66
STOR	-	-	1	1.06	1.21	1.56
VIRT	-	0.87	1	1.15	1.30	-
TURN	-	0.87	1	1.07	1.15	-
ACAP	1.46	1.19	1	0.86	0.71	-
AEXP	1.29	1.13	1	0.91	0.82	-
PCAP	1.42	1.17	1	0.86	0.70	-
VEXP	1.21	1.10	1	0.9	-	-
LEXP	1.14	1.07	1	0.95	-	-
MODP	1.24	1.10	1	0.91	0.82	-
TOOL	1.29	1.10	1	0.91	0.83	-
SCED	1.24	1.10	1	1.04	1.1	-

Numerical 2

A new project with estimated 400 KLOC embedded system has to be developed project manager has a choice of hiring from 2 pools of developers

1) very high application experience with little experience in programming language

2) Developers of low App experience but a High programming language experience

which is better choice in terms of two pools?

Solution

- Given

  Cost Drivers:-
  - AEXP
  - LEXP

- Case 1

  EAF = 0.82 x 1.14 = 0.934

  Efforts:

  $$E = a \times (KLOC)^b \times EAF = 2.8 \times 400^{1.2} \times 0.934 = 3470PM$$

  Development Time:

  $$D = c \times E^d = 2.5 \times 3470^{0.32} = 33.9M$$

- Case 2

  EAF = 1.29 x 0.95 = 1.22

  Efforts:

  $$E = a \times (KLOC)^b \times EAF = 2.8 \times 400^{1.2} \times 1.22 = 4528PM$$

  Development Time:

  $$D = c \times E^d = 2.5 \times 4528^{0.32} = 36.9M$$

  Since E and D of case one is less, pool one developers are the best option

Detailed Model

The Detailed COCOMO model goes further by partitioning the project into its constituent phases or modules (such as requirements, design, coding, testing, integration). The effort for each phase is estimated separately by applying the basic or intermediate model with additional “phase-sensitive” cost drivers. The overall project effort is the sum of the estimated efforts for each phase.

Development Phases typically include:

- Planning and Requirements
- System Design
- Detailed Design
- Module Code and Test
- Integration and Test
- Cost Construction (final summing)

***For each module, the formulas similar to the intermediate model are applied. Then, effort distribution percentages are applied to each phase to determine phase-specific schedules and staffing*.**

Use of Detailed Estimates

By breaking down the project, the detailed model allows:

- Fine-tuned estimations that account for varying productivity levels in different phases.
- Better insight into staffing and scheduling per phase.
- Improved accuracy when sufficient information is available (this is especially useful for projects with significant complexity or variability among modules).

Numerical 3

Q. Consider a project with the following main components

(1) Screen Edit
(2) CLI
(3) File I/P and O/P
(4) Cursor Movt
(5) Screen Movement

The sizes for these are estimated to be 4k, 2k, 1k, 2k, 3k LOC.
Using COCOMO, determine

I: Overall cost and schedule estimates
(assume values for **cost drivers**, with at least 3 being different from 1.0)

II: Cost and schedule estimates for different phases.

Phase sensitive multiplier for effort

- Planning : 0.06
- System design : 0.16
- Detailed design : 0.26
- Code : 0.42
- Testing : 0.16

Phase sensitive multiplier for duration

- Planning : 0.10
- System design : 0.19
- Detailed design : 0.24
- Code : 0.39
- Testing : 0.18

Solution

Assuming cost drivers:

- RELY= high
- LEXP = low
- CPLX = high
- ACAP = high

EAF = 1.216

Efforts:

$$E = a \times (KLOC)^b \times EAF = 3.2 \times 400^{1.05} \times 1.216 = 52.9PM$$

Development Time:

$$D = c \times E^d = 2.5 \times 52.9^{0.38} = 11.29M$$

solution

- Effort:

  Plan = 0.06 x 52.9 = 3.174

  Design = 0.16 x 52.9 = 8.464

  Detailed design = 0.26 x 52.9 = 13.754

  Code = 0.42 x 52.9 = 22.218

  Test = 0.16 x 52.9 =8.464

- Development time:

  Plan = 0.1 x 11.29 = 1.129

  Design = 0.19 x 11.29 = 2.1451

  Detailed design = 0.24 x 11.29 = 2.70

  Code = 0.39 x 11.29 = 4.4031

  Test = 0.18 x 11.29 = 2.03

Limitations of COCOMO 1

- **Accuracy:**
  COCOMO 1 provides nominal estimates. Actual costs can vary if, for example, a project is forced to be completed in less time than estimated.
- **Dependency on KLOC:**
  The model is sensitive to the estimated size of the code. Early in a project, estimating KLOC accurately can be challenging.
- **Cost Drivers Subjectivity:**
  In the Intermediate and Detailed models, ratings for cost drivers are subjective and require historical data or expert judgment.
- **Not Agile-Friendly:**
  COCOMO 1 was developed for waterfall-style projects and may be less applicable to iterative or agile environments without modifications.

Comparison with Other Methods

- **Expert Judgment:**
  While expert judgment can be quick, its subjectivity often leads to inconsistent estimates. COCOMO’s reliance on historical data and mathematical models reduces bias.
- **Function Point Analysis (FPA):**
  FPA measures functionality delivered rather than lines of code and can be very effective, especially in early design phases. However, FPA requires significant effort to standardize the process and is subject to variability in counting and weighting. COCOMO (especially in its intermediate/detailed forms) complements FPA by translating size estimates into effort and schedule predictions using a well-understood cost model.

Conclusion

The COCOMO 1 model remains a classic tool in software engineering for estimating project effort and schedule. Its three forms—Basic, Intermediate, and Detailed—provide a range of estimation granularity:

- The **Basic Model** is useful for early, rough estimates based solely on size.
- The **Intermediate Model** improves accuracy by incorporating 15 cost drivers.
- The **Detailed Model** divides the project into phases or modules for an even more refined estimate.
- Understanding each project type (Organic, Semi-detached, and Embedded) and applying the proper coefficients are key to using the model effectively. Although modern practices sometimes call for more agile estimation techniques, COCOMO’s structured approach and empirical foundation offer a valuable baseline against which many other models are compared.

COCOMO-2

RADHIKAA DUGGAL

23FE10CAI00032

Overview

COCOMO II was developed as a successor to the original COCOMO model to address modern software development practices such as rapid prototyping, reuse, and distributed development. It is built on an empirical basis like its predecessor but has been extended and refined over time to better reflect contemporary development environments.

3 Sub-models

COCOMO II is structured into three sub models which are used at different stages of the software development lifecycle:

- **Application Composition Model:**
  Used early in the development process (often during prototyping or when high-level object points can be determined). It relies primarily on object points and estimates the effort needed based on new object points after accounting for reuse.
- **Early Design Model:**
  This sub model is applied once a rough architecture is in place. It uses high-level size measures combined with a small set of cost drivers and scale factors to predict effort.
- **Post-Architecture Model:**
  This is the most detailed model used after the system architecture has been defined. It uses a larger number of cost drivers and scale factors to produce a precise, phase-sensitive effort estimate.

Application Composition Model

The Application Composition Model is typically used during the early stages of a project when a significant amount of the system is to be developed using component-based or rapid application development (RAD) approaches. In this model, the size is not measured in KLOC but in **object points** (which represent screens, reports, and 3GL components). The estimate considers the percentage of reuse to derive the new object points (NOP).

Key Steps & Formula:

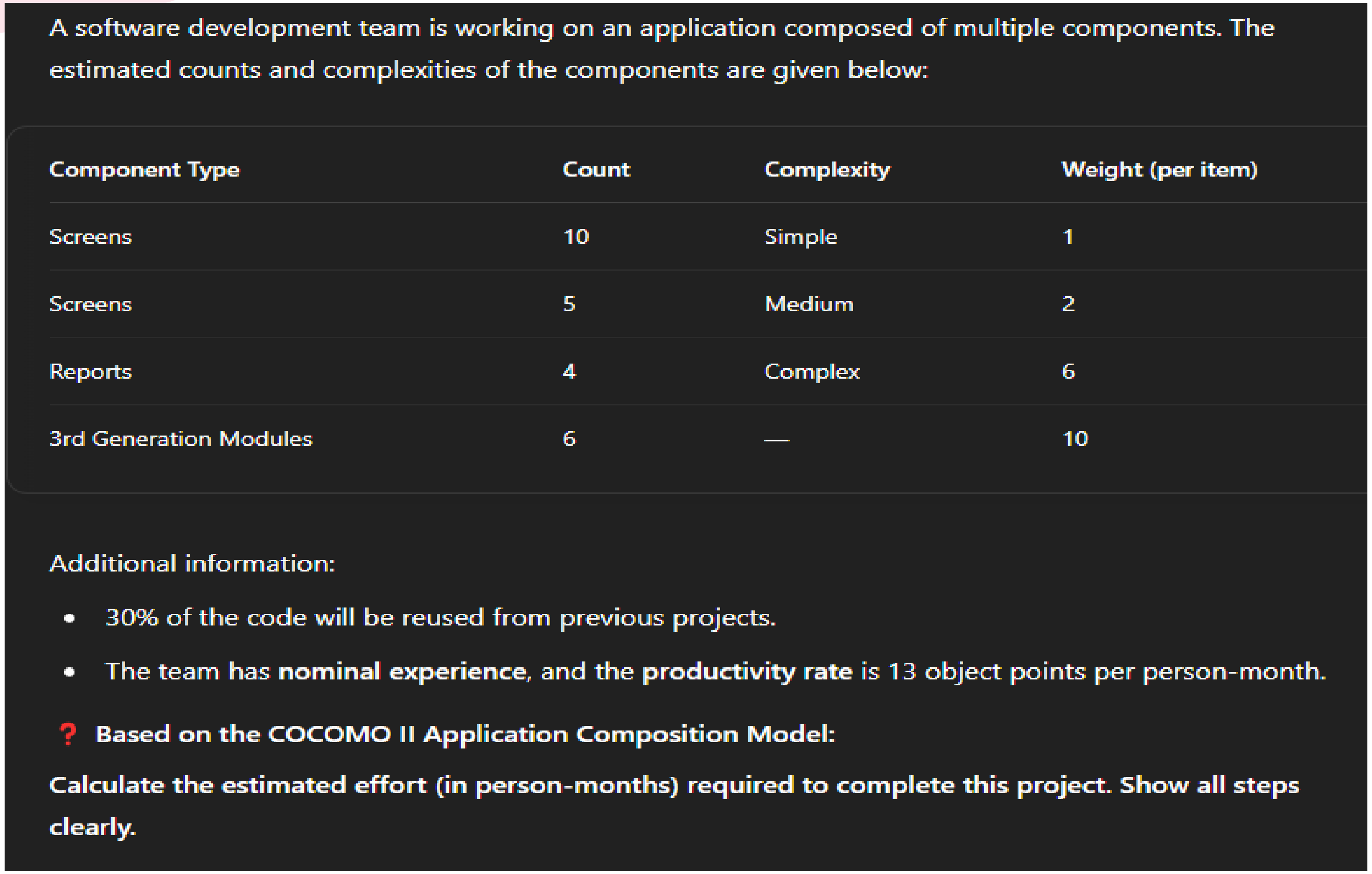
- **Calculate Object Points (OP):**
  Object points are determined by identifying and classifying components such as screens, reports, and 3GL components and weighting them (simple, average, or complex).
- **Adjust for Reuse:**

$$NOP = OP \times \left(\frac{100 \ - \%reuse}{100}\right)$$

- **Effort Estimation:**
  The estimated effort (in person-months) is then obtained using a productivity factor (PROD):

$$Effort = \frac{NOP}{PROD}$$

Numerical 1



A software development team is working on an application composed of multiple components. The estimated counts and complexities of the components are given below:

Component Type	Count	Complexity	Weight (per item)
Screens	10	Simple	1
Screens	5	Medium	2
Reports	4	Complex	6
3rd Generation Modules	6	—	10

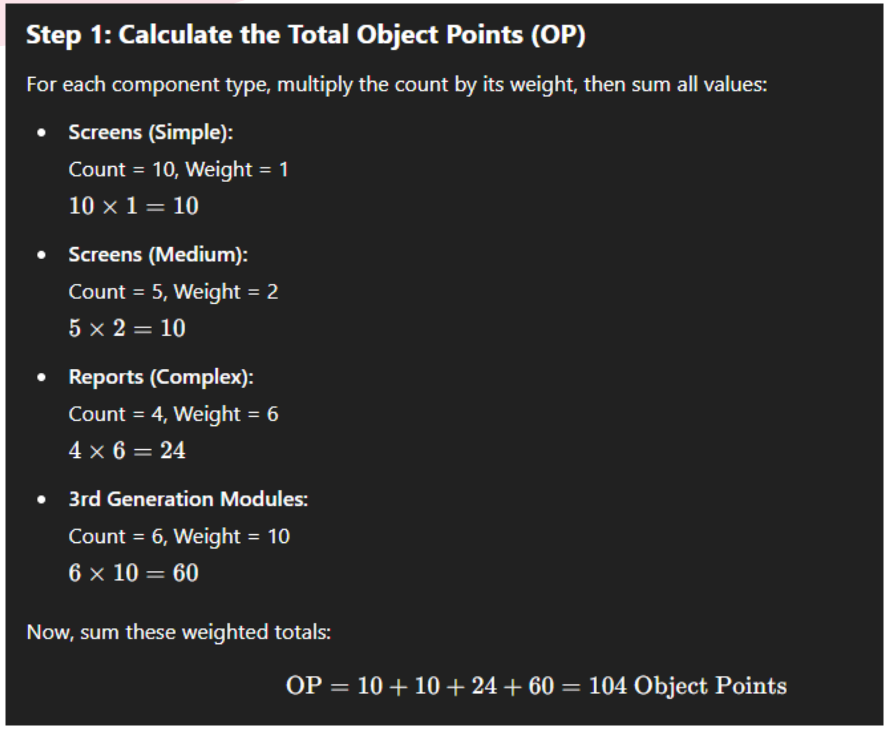
Additional information:

- 30% of the code will be reused from previous projects.
- The team has **nominal experience**, and the **productivity rate** is 13 object points per person-month.

❓ **Based on the COCOMO II Application Composition Model:**

Calculate the estimated effort (in person-months) required to complete this project. Show all steps clearly.

solution



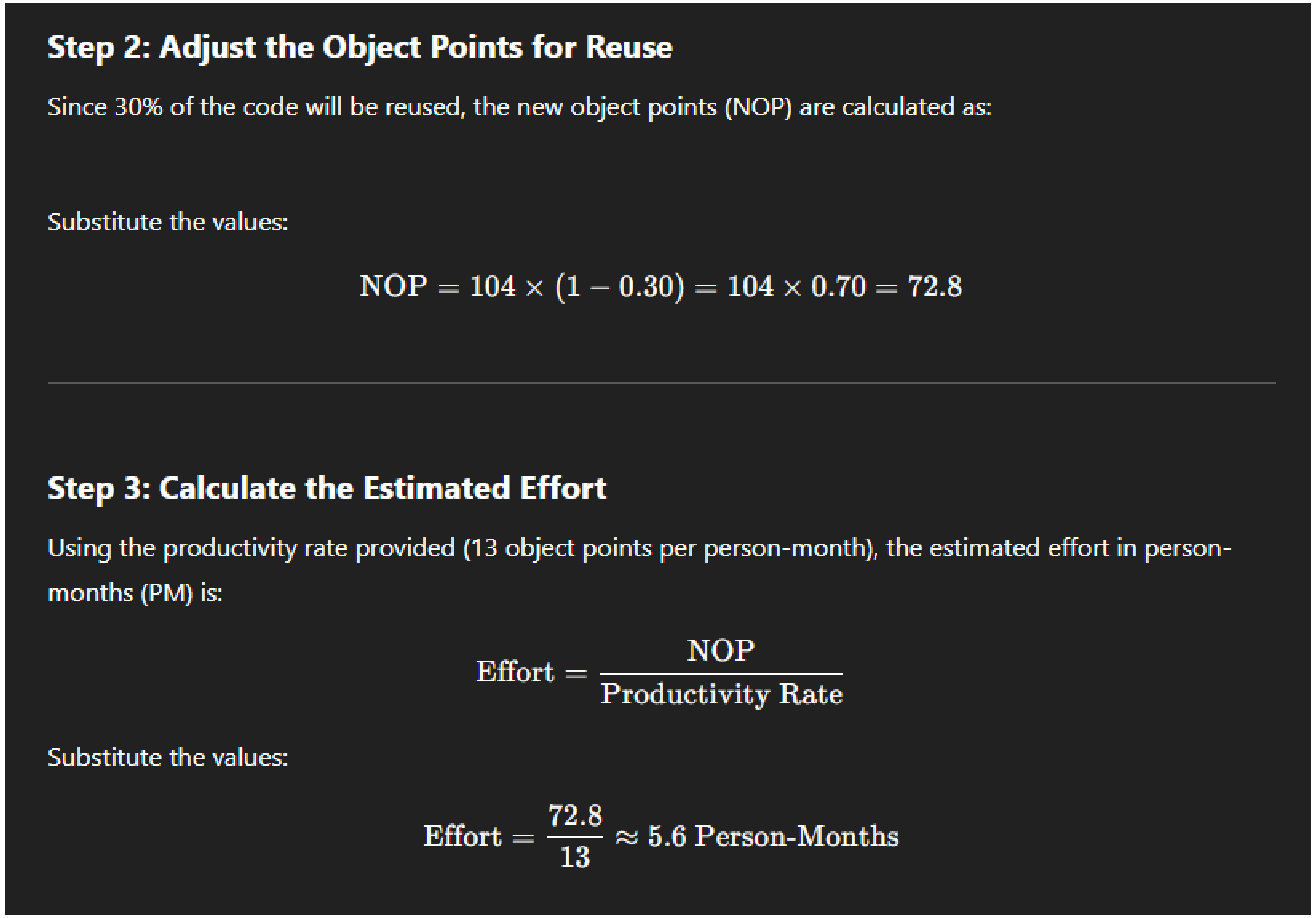
Step 1: Calculate the Total Object Points (OP)

For each component type, multiply the count by its weight, then sum all values:

- **Screens (Simple):**
  Count = 10, Weight = 1
  $10 \times 1 = 10$
- **Screens (Medium):**
  Count = 5, Weight = 2
  $5 \times 2 = 10$
- **Reports (Complex):**
  Count = 4, Weight = 6
  $4 \times 6 = 24$
- **3rd Generation Modules:**
  Count = 6, Weight = 10
  $6 \times 10 = 60$

Now, sum these weighted totals:

$$\text{OP} = 10 + 10 + 24 + 60 = 104 \text{ Object Points}$$



Step 2: Adjust the Object Points for Reuse

Since 30% of the code will be reused, the new object points (NOP) are calculated as:

Substitute the values:

$$\text{NOP} = 104 \times (1 - 0.30) = 104 \times 0.70 = 72.8$$

Step 3: Calculate the Estimated Effort

Using the productivity rate provided (13 object points per person-month), the estimated effort in person-months (PM) is:

$$\text{Effort} = \frac{\text{NOP}}{\text{Productivity Rate}}$$

Substitute the values:

$$\text{Effort} = \frac{72.8}{13} \approx 5.6 \text{ Person-Months}$$

Early Design Model

Once a preliminary system architecture is in place, the Early Design Model is used. It emphasizes the use of high-level cost drivers and scale factors with a moderate level of detail. Unlike COCOMO 81 (which primarily used KLOC), Early Design can incorporate function points or KLOC alongside a handful of cost drivers.

Key Steps & Formula:

The primary estimation formula is:

$$Effort = A \times (size)^{E} \times EAF$$

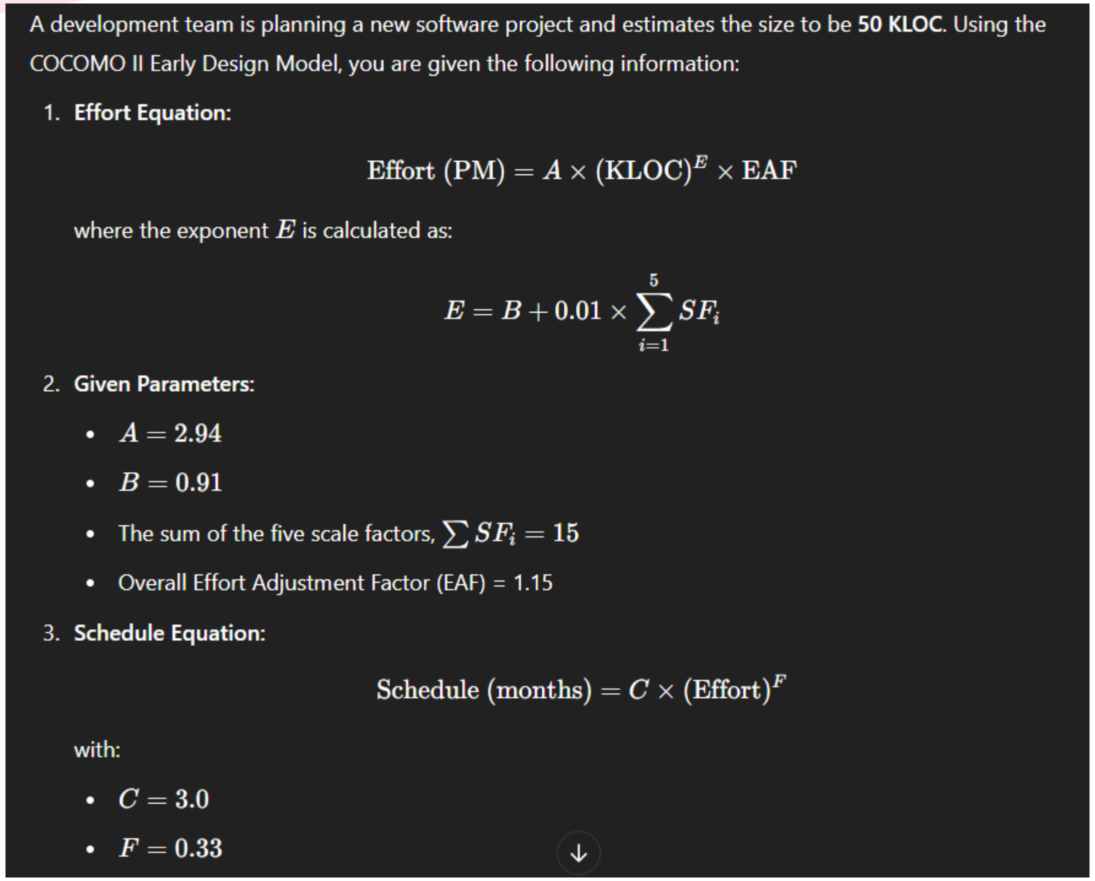
where:

- **Size** is the estimated size (typically in KLOC or adjusted function points).
- **A** is a constant (often around 2.94 or another calibrated value).
- **E** is an exponent derived from scale factors. It is computed as:

$$E = B + 0.01 \times \sum_{i=1}^{5} SF_i$$

  where $\boldsymbol{SF_i}$ are the scale factors and **B** is a baseline exponent (often around 0.91).
- **EAF** is the Effort Adjustment Factor calculated as the product of selected cost drivers

Numerical 2



A development team is planning a new software project and estimates the size to be **50 KLOC**. Using the COCOMO II Early Design Model, you are given the following information:

1. **Effort Equation:**

$$\text{Effort (PM)} = A \times (\text{KLOC})^E \times \text{EAF}$$

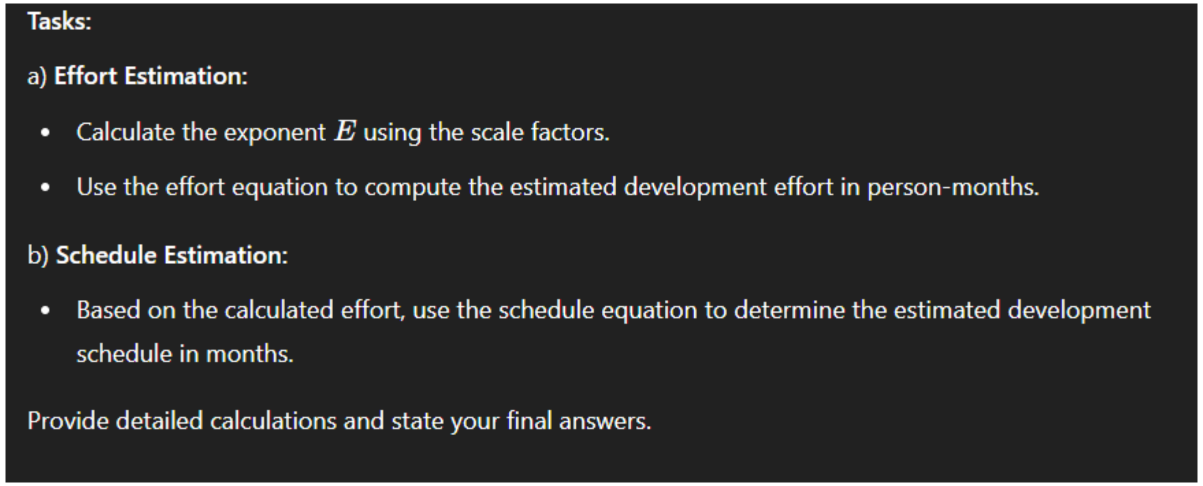
   where the exponent E is calculated as:

$$E = B + 0.01 \times \sum_{i=1}^{5} SF_i$$

2. **Given Parameters:**
   - $A = 2.94$
   - $B = 0.91$
   - The sum of the five scale factors, $\sum SF_i = 15$
   - Overall Effort Adjustment Factor (EAF) = 1.15
3. **Schedule Equation:**

$$\text{Schedule (months)} = C \times (\text{Effort})^F$$

   with:
   - $C = 3.0$
   - $F = 0.33$



Tasks:

a) **Effort Estimation:**

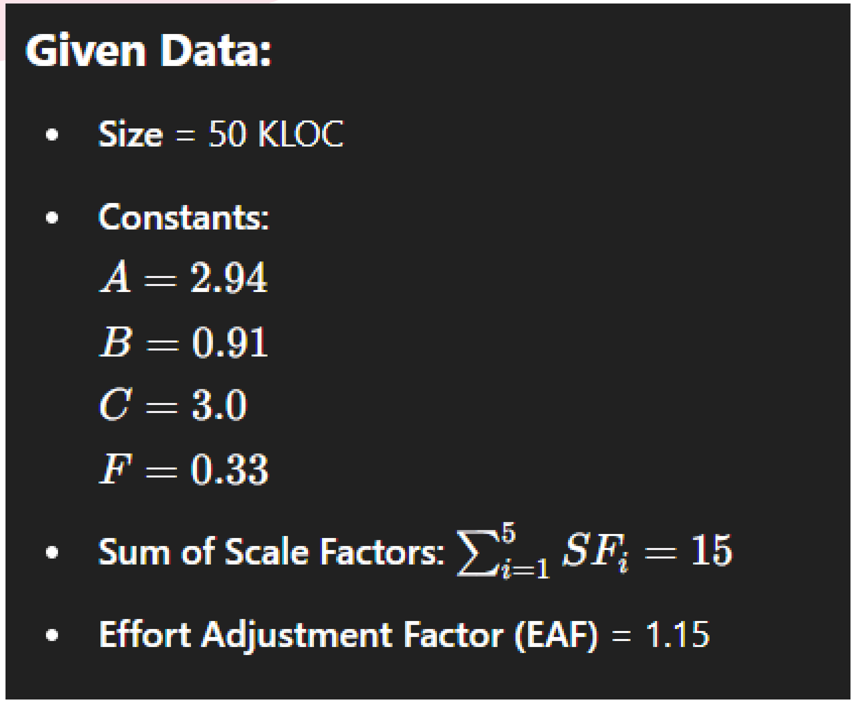
- Calculate the exponent E using the scale factors.
- Use the effort equation to compute the estimated development effort in person-months.

b) **Schedule Estimation:**

- Based on the calculated effort, use the schedule equation to determine the estimated development schedule in months.

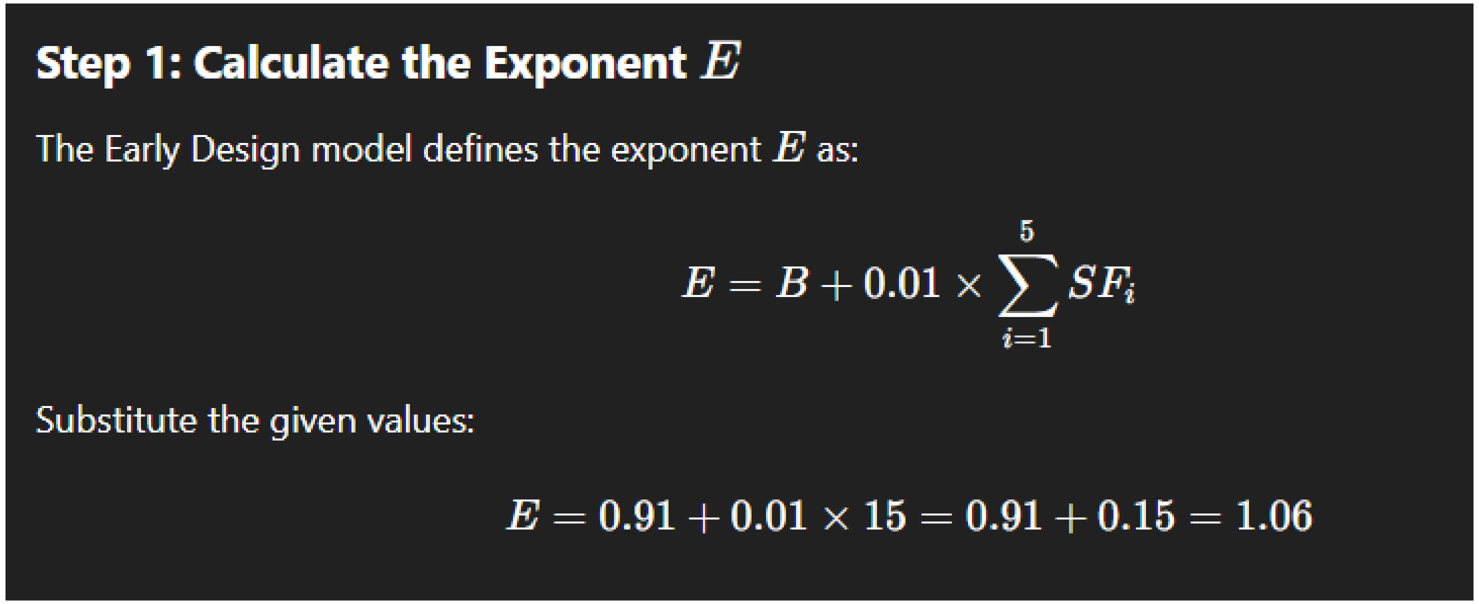
Provide detailed calculations and state your final answers.

Solution



Given Data:

- **Size** = 50 KLOC
- **Constants:**
  $A = 2.94$
  $B = 0.91$
  $C = 3.0$
  $F = 0.33$
- **Sum of Scale Factors:** $\sum_{i=1}^{5} SF_i = 15$
- **Effort Adjustment Factor (EAF)** = 1.15



Step 1: Calculate the Exponent E

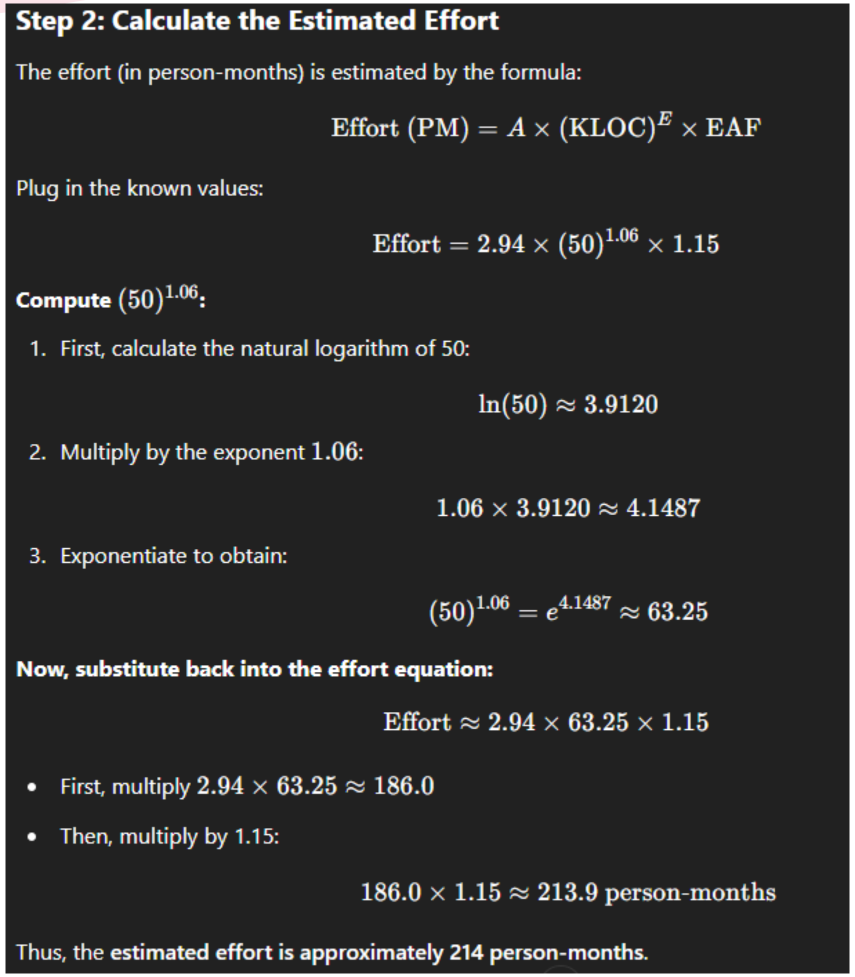
The Early Design model defines the exponent E as:

$$E = B + 0.01 \times \sum_{i=1}^{5} SF_i$$

Substitute the given values:

$$E = 0.91 + 0.01 \times 15 = 0.91 + 0.15 = 1.06$$

Solution



Step 2: Calculate the Estimated Effort

The effort (in person-months) is estimated by the formula:

$$\text{Effort (PM)} = A \times (\text{KLOC})^E \times \text{EAF}$$

Plug in the known values:

$$\text{Effort} = 2.94 \times (50)^{1.06} \times 1.15$$

Compute $(50)^{1.06}$**:**

1. First, calculate the natural logarithm of 50:

$$\ln(50) \approx 3.9120$$

2. Multiply by the exponent 1.06:

$$1.06 \times 3.9120 \approx 4.1487$$

3. Exponentiate to obtain:

$$(50)^{1.06} = e^{4.1487} \approx 63.25$$

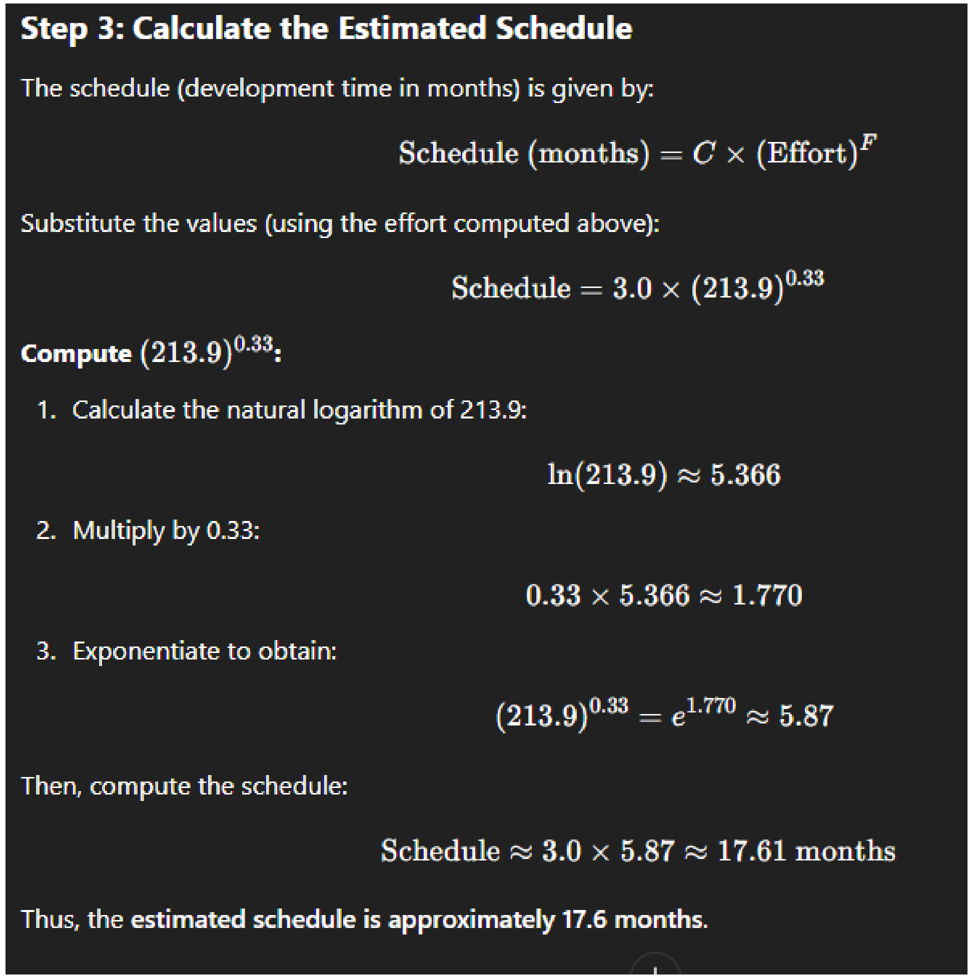
Now, substitute back into the effort equation:

$$\text{Effort} \approx 2.94 \times 63.25 \times 1.15$$

- First, multiply $2.94 \times 63.25 \approx 186.0$
- Then, multiply by 1.15:

$$186.0 \times 1.15 \approx 213.9 \text{ person-months}$$

Thus, the **estimated effort is approximately 214 person-months.**



Step 3: Calculate the Estimated Schedule

The schedule (development time in months) is given by:

$$\text{Schedule (months)} = C \times (\text{Effort})^F$$

Substitute the values (using the effort computed above):

$$\text{Schedule} = 3.0 \times (213.9)^{0.33}$$

Compute $(213.9)^{0.33}$**:**

1. Calculate the natural logarithm of 213.9:

$$\ln(213.9) \approx 5.366$$

2. Multiply by 0.33:

$$0.33 \times 5.366 \approx 1.770$$

3. Exponentiate to obtain:

$$(213.9)^{0.33} = e^{1.770} \approx 5.87$$

Then, compute the schedule:

$$\text{Schedule} \approx 3.0 \times 5.87 \approx 17.61 \text{ months}$$

Thus, the **estimated schedule is approximately 17.6 months.**

Post-Architecture Model

After the system architecture is fully defined, the Post-Architecture Model is used for detailed estimation. This model is more granular, using a comprehensive list of cost drivers (typically 17 in number) along with scale factors. It predicts effort across different phases of the software lifecycle.

Key Steps & Formula:

The effort estimate follows a similar equation:

$$Effort = A \times (size)^{E} \times \prod_{i=1}^{n} EM_i$$

where:

- **A** is a calibrated constant.
- **Size** is still the estimated size in KLOC.
- **E** is the exponent calculated as in the Early Design model:

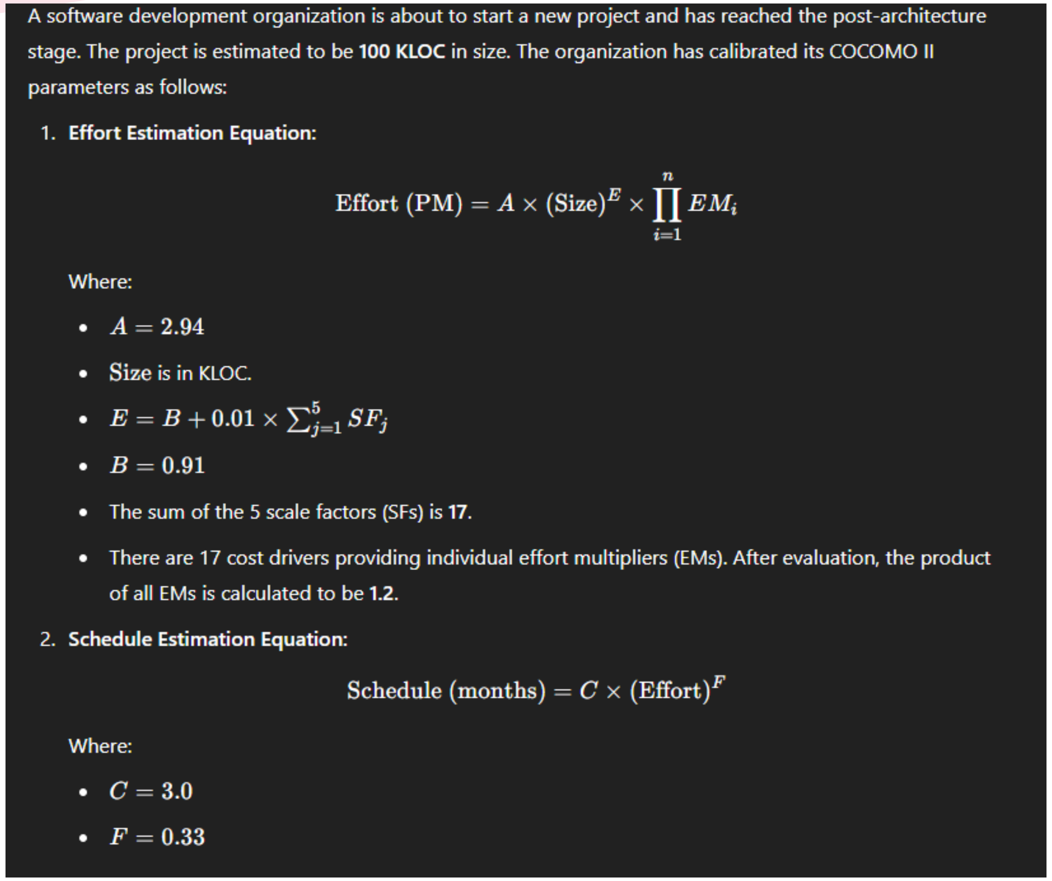
$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

- $\boldsymbol{EM_i}$ are the effort multipliers (cost drivers) for each of the n cost drivers.

Additional Aspects:

- **Phase Sensitivity:**
  The model might further break down effort by phase (e.g., requirements, design, implementation, testing) using additional multipliers.
- **Calibration:**
  Organizations often calibrate constants (A, B) and multipliers using data from past projects to improve the accuracy of estimates.

Numerical



A software development organization is about to start a new project and has reached the post-architecture stage. The project is estimated to be **100 KLOC** in size. The organization has calibrated its COCOMO II parameters as follows:

1. **Effort Estimation Equation:**

$$\text{Effort (PM)} = A \times (\text{Size})^E \times \prod_{i=1}^{n} EM_i$$

   Where:

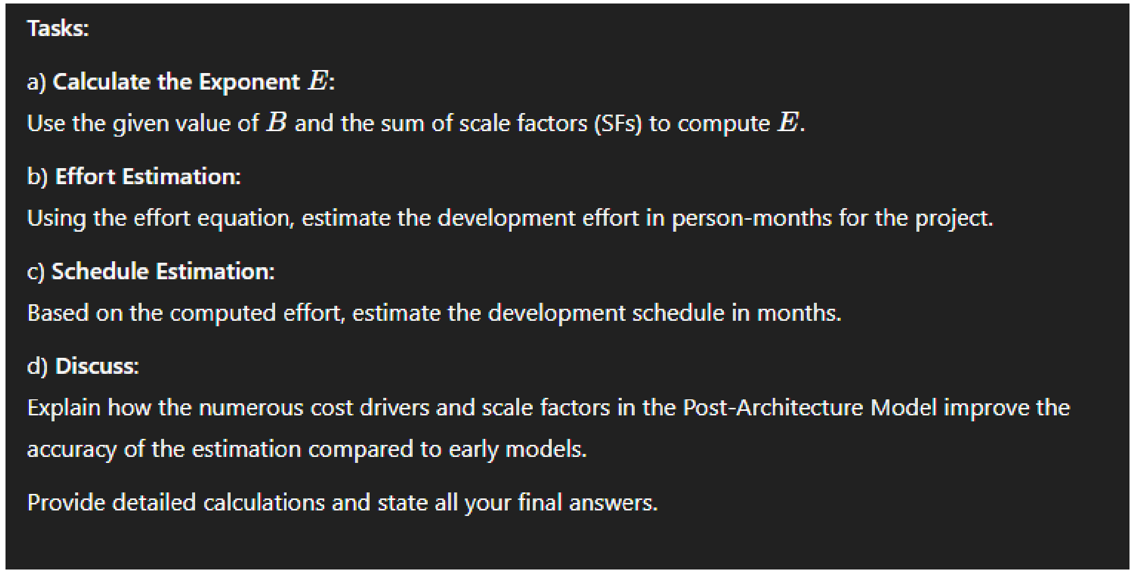
   - $A = 2.94$
   - Size is in KLOC.
   - $E = B + 0.01 \times \sum_{j=1}^{5} SF_j$
   - $B = 0.91$
   - The sum of the 5 scale factors (SFs) is **17**.
   - There are 17 cost drivers providing individual effort multipliers (EMs). After evaluation, the product of all EMs is calculated to be **1.2**.

2. **Schedule Estimation Equation:**

$$\text{Schedule (months)} = C \times (\text{Effort})^F$$

   Where:

   - $C = 3.0$
   - $F = 0.33$



Tasks:

a) **Calculate the Exponent E:**

Use the given value of B and the sum of scale factors (SFs) to compute E.

b) **Effort Estimation:**

Using the effort equation, estimate the development effort in person-months for the project.

c) **Schedule Estimation:**

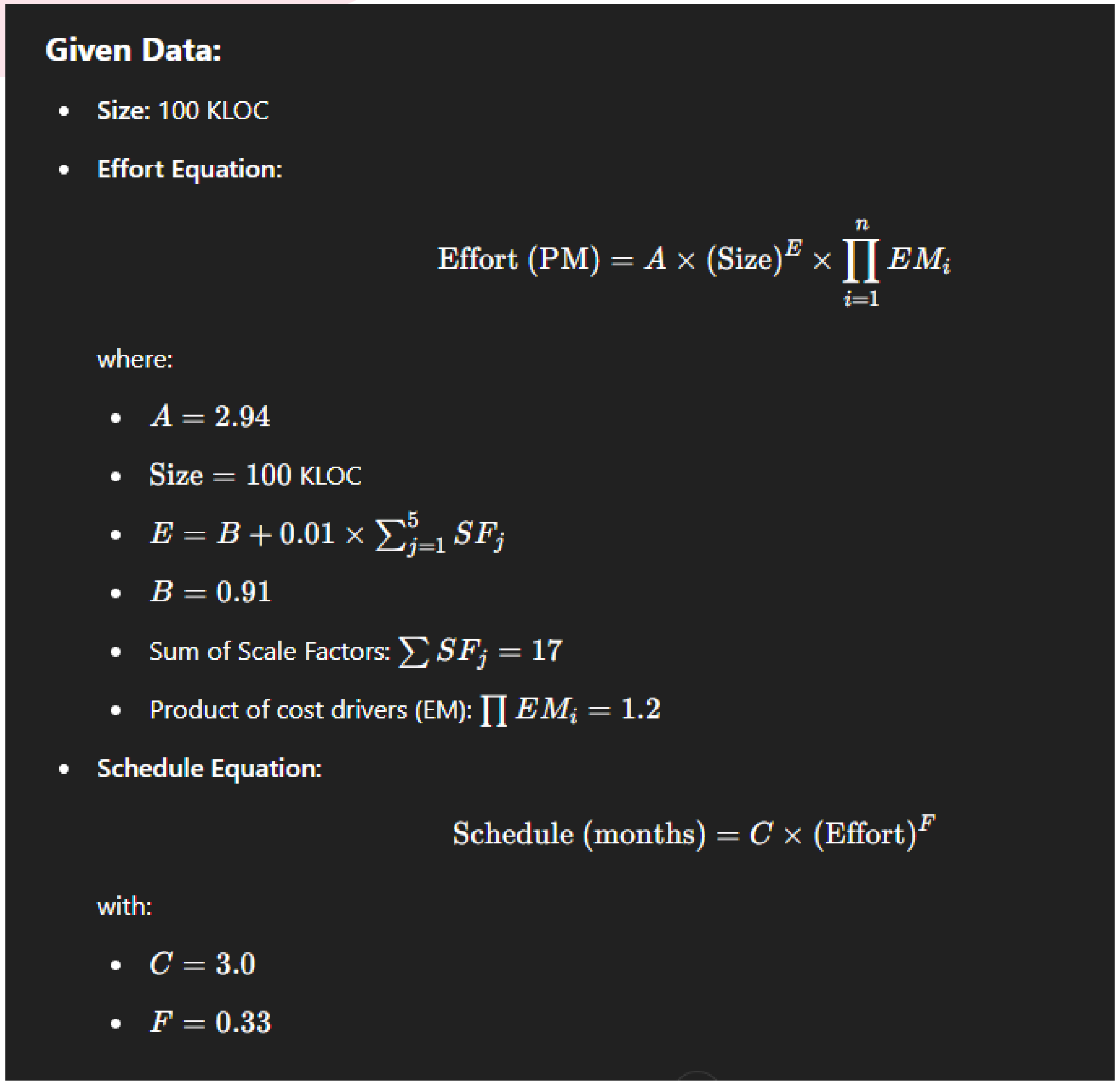
Based on the computed effort, estimate the development schedule in months.

d) **Discuss:**

Explain how the numerous cost drivers and scale factors in the Post-Architecture Model improve the accuracy of the estimation compared to early models.

Provide detailed calculations and state all your final answers.

Solution



Given Data:

- **Size:** 100 KLOC
- **Effort Equation:**

$$\text{Effort (PM)} = A \times (\text{Size})^E \times \prod_{i=1}^{n} EM_i$$

where:

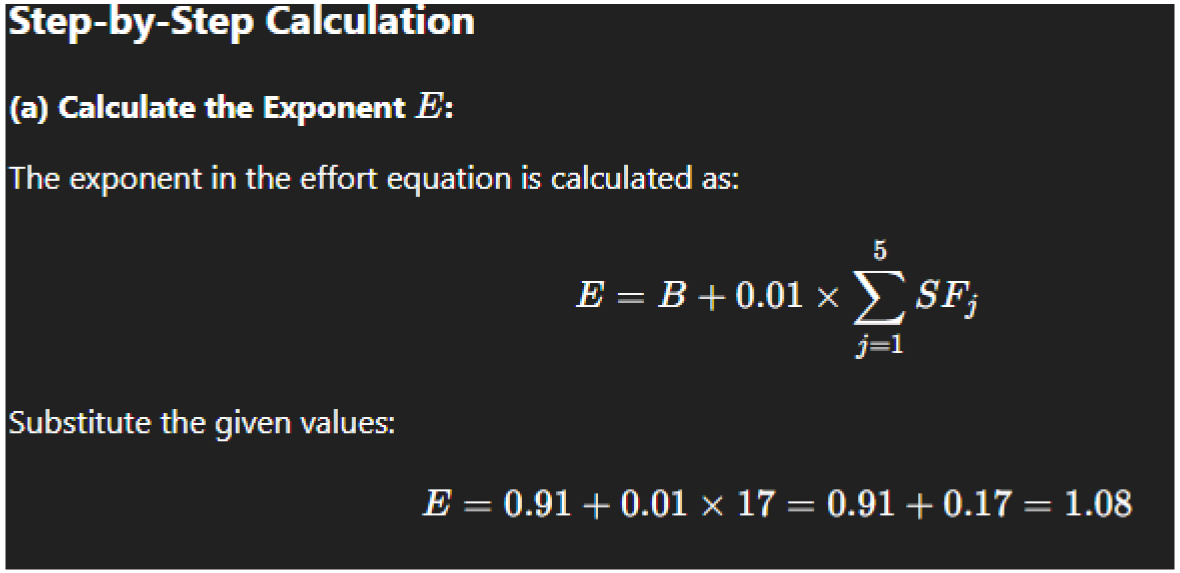
- $A = 2.94$
- $\text{Size} = 100$ KLOC
- $E = B + 0.01 \times \sum_{j=1}^{5} SF_j$
- $B = 0.91$
- Sum of Scale Factors: $\sum SF_j = 17$
- Product of cost drivers (EM): $\prod EM_i = 1.2$

- **Schedule Equation:**

$$\text{Schedule (months)} = C \times (\text{Effort})^F$$

with:

- $C = 3.0$
- $F = 0.33$



Step-by-Step Calculation

(a) Calculate the Exponent E:

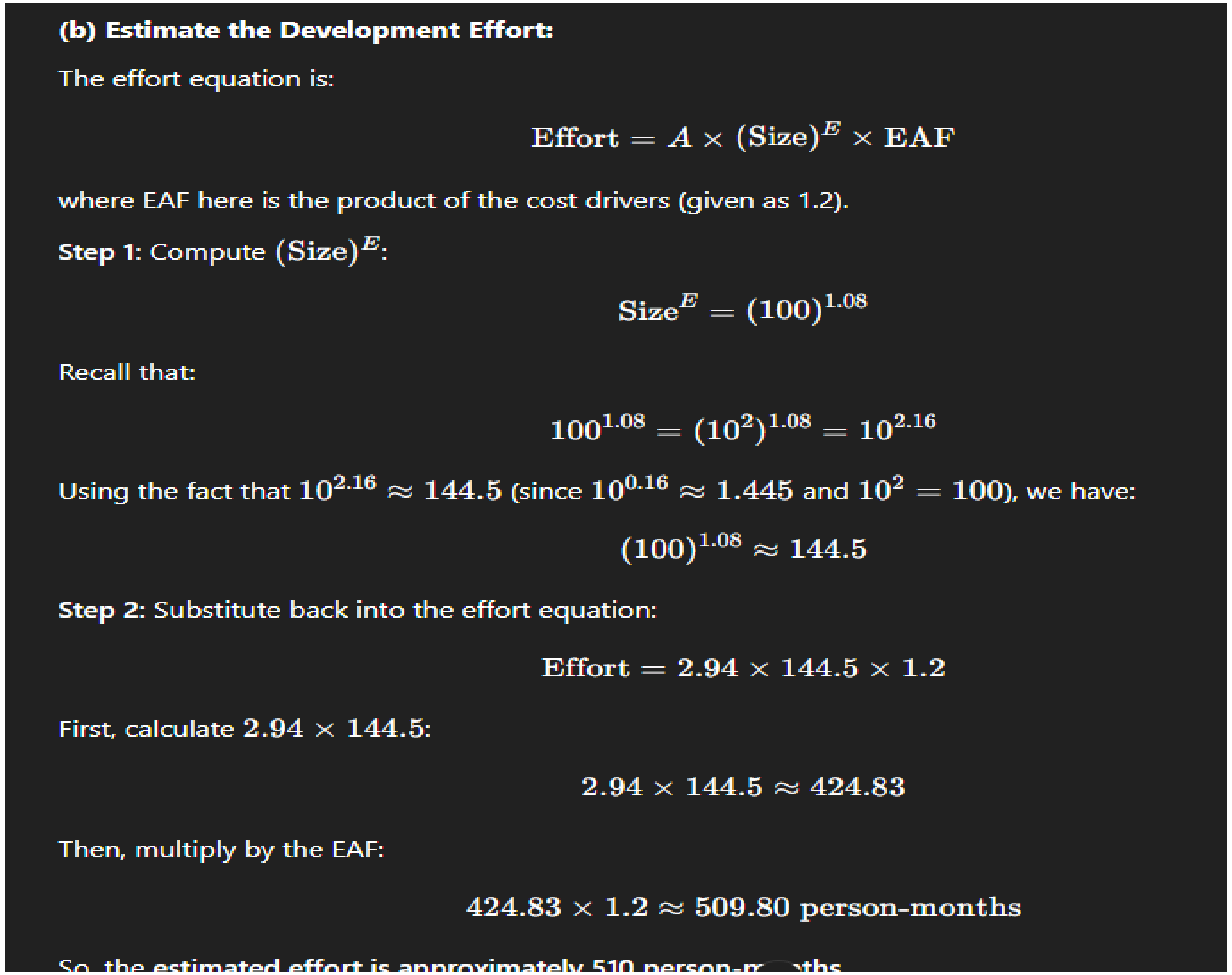
The exponent in the effort equation is calculated as:

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

Substitute the given values:

$$E = 0.91 + 0.01 \times 17 = 0.91 + 0.17 = 1.08$$

Solution



(b) Estimate the Development Effort:

The effort equation is:

$$\text{Effort} = A \times (\text{Size})^E \times \text{EAF}$$

where EAF here is the product of the cost drivers (given as 1.2).

Step 1: Compute $(\text{Size})^E$:

$$\text{Size}^E = (100)^{1.08}$$

Recall that:

$$100^{1.08} = (10^2)^{1.08} = 10^{2.16}$$

Using the fact that $10^{2.16} \approx 144.5$ (since $10^{0.16} \approx 1.445$ and $10^2 = 100$), we have:

$$(100)^{1.08} \approx 144.5$$

Step 2: Substitute back into the effort equation:

$$\text{Effort} = 2.94 \times 144.5 \times 1.2$$

First, calculate 2.94×144.5:

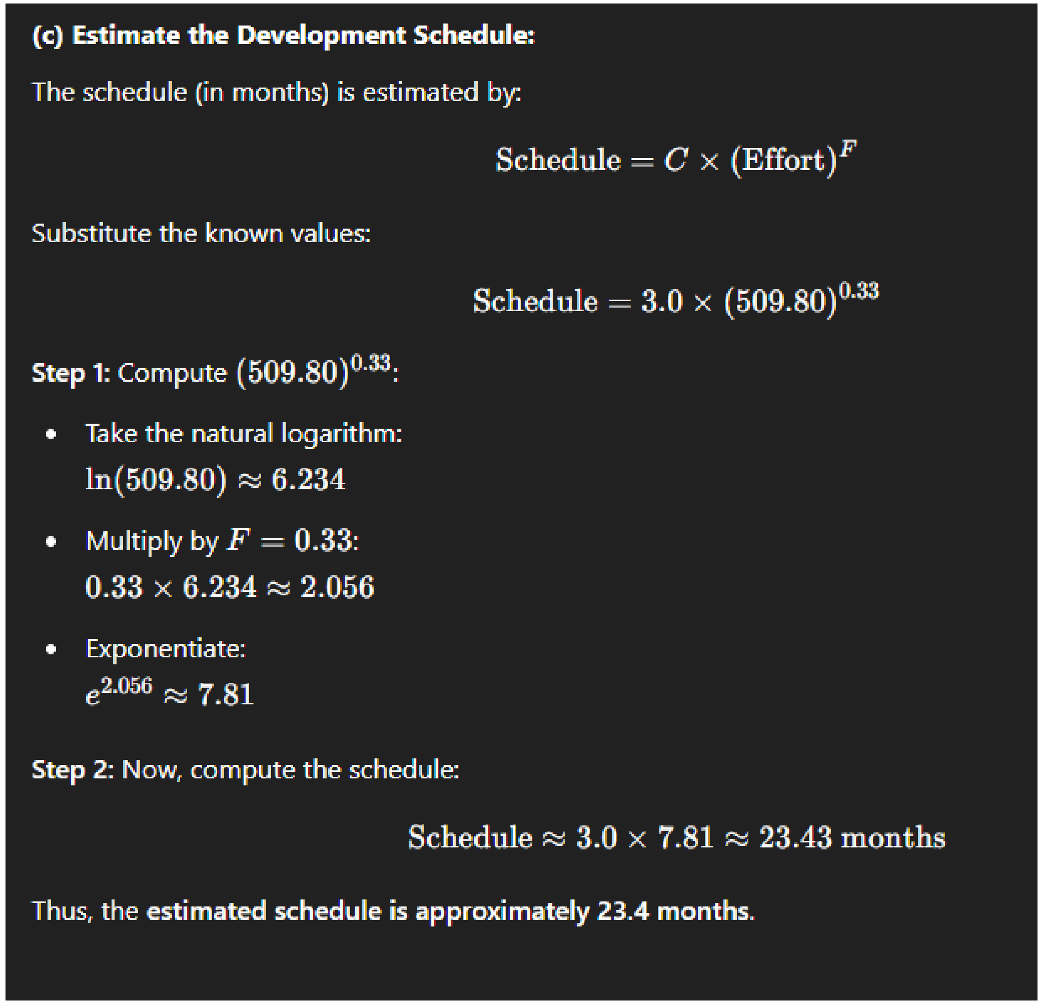
$$2.94 \times 144.5 \approx 424.83$$

Then, multiply by the EAF:

$$424.83 \times 1.2 \approx 509.80 \text{ person-months}$$

So, the estimated effort is approximately 510 person-m[illegible]ths

Solution



(c) Estimate the Development Schedule:

The schedule (in months) is estimated by:

$$\text{Schedule} = C \times (\text{Effort})^F$$

Substitute the known values:

$$\text{Schedule} = 3.0 \times (509.80)^{0.33}$$

Step 1: Compute $(509.80)^{0.33}$:

- Take the natural logarithm:
  $\ln(509.80) \approx 6.234$
- Multiply by $F = 0.33$:
  $0.33 \times 6.234 \approx 2.056$
- Exponentiate:
  $e^{2.056} \approx 7.81$

Step 2: Now, compute the schedule:

$$\text{Schedule} \approx 3.0 \times 7.81 \approx 23.43 \text{ months}$$

Thus, the **estimated schedule is approximately 23.4 months.**

Differences Between COCOMO I and COCOMO II

Purpose and Scope

COCOMO I:

- Designed in the early 1980s.
- Based on the waterfall development model.
- Uses KLOC as the principal size metric.
- Has three modes with relatively static coefficients calibrated on 63 projects.
- Simpler cost drivers and does not address reuse or modern development practices extensively.

COCOMO II:

- Developed to address the shortcomings of COCOMO 81 in the context of modern software development.
- Incorporates new size measures
- Provides three sub models to be used at different stages of a project.
- Uses a more extensive set of cost drivers and scale factors to reflect complexities in modern projects.
- Better suited for iterative and evolutionary development processes.

Estimation Methods and Detail

COCOMO I:

- Predominantly a “size-based” model using a power-law relationship between KLOC and effort.
- Offers Basic, Intermediate, and Detailed versions; however, all of them assume a waterfall process.

COCOMO II:

- Introduces scale factors that adjust the exponent of the effort equation.
- Includes additional effort multipliers and drivers that cover more modern considerations.
- Provides separate sub models that are tailored for early-stage prototyping versus post-architecture detailed estimation.
- More adaptable to agile or iterative development, though originally still derived from traditional practices.

Estimation Methods and Detail

COCOMO I:

- Tends to work reasonably well for legacy projects but may lead to
under- or over-estimates when applied to modern projects with a high degree of reuse.

COCOMO II:

- Designed to be calibrated to an organization's historical data so
that it reflects current practices, technology, and team capabilities better than
the fixed coefficients used in COCOMO I.

Limitations

•A heavy reliance on size metrics (KLOC or function points) which are hard to predict early.

•The need for extensive, high-quality historical data for proper calibration.

•Complexity in input assessment due to numerous and often subjective cost drivers.

•Partial alignment with modern agile and iterative processes.

•A static estimation process that can lag behind rapid changes in project scope or requirements.

•Inflexibility when applied to non-traditional development environments.

Final Remarks

COCOMO II is widely regarded as a more modern and versatile estimation tool. It allows organizations to derive estimates from early prototyping (via the Application Composition Model) to detailed phase-by-phase estimates (via the Post-Architecture Model). Although both models have limitations, the improvements of COCOMO II make it better aligned with contemporary software engineering practices.

Thank You