Project Title: UrbanGarden

Mohib Abbas Sayed – 2103158

Hamza Sayyed – 2103159

Om Shete – 2103163

EXPERIMENT NO: 09

Aim: Application of the COCOMO model for cost estimation of the project.

Theory:

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e. number of Lines of Code. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality.
The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort and schedule:

- Effort: Amount of labour that will be required to complete a task. It is measured in person-months units.
- Schedule: Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, and months.

COCOMO 1:

Software development projects can be classified into the following categories based on the development complexity:

1. **Organic**: A software project is said to be organic if the team size required is adequately small, the problem is well understood and has been solved in the past, and the team members have a nominal experience regarding the problem.

2. **Semi-Detached**: A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environments lie in between organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and challenging to develop than organic ones and require more experience better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered Semi-Detached types.

3. **Embedded**: A software project requiring the highest complexity, creativity, and experience requirement falls under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

Estimation of Effort:

Organic: Effort = $2.4(KLOC)^{1.05}$ PM

Semi-detached: Effort = $3.0(KLOC)^{1.12}$ PM

Embedded: Effort = $3.6(KLOC)^{1.20}$ PM

Estimation of Development Time:

Organic: Tdev= $2.5(Effort)^{0.38}$ Months

Semi-detached: Tdev= $2.5(Effort)^{0.35}$ Months

Embedded: Tdev= $2.5(Effort)^{0.32}$ Months

Question:

We have determined our project fits the characteristics of Basic, Semi-Detached mode. We estimate our project will have 83,000 Delivered Source Instructions.

Find the Effort, Schedule, Productivity, and average staffing required for the project.

Answer:

BASIC MODE:

Using the formulas, we can estimate:

1. **Organic -**

Effort	= 2.4 * $(KLOC)^{1.05}$ PM
	= 2.4 * $(83000)^{1.05}$ PM
	= 350948 PM
Schedule	= 2.5 * $(Effort)^{0.38}$ Months
	= 2.5 * $(350948)^{0.38}$ Months
	= 320 Months
Productivity	= DSI / Effort
	= 83000 DSI / 350948 PM
	= 0.24 DSI/PM

Average. Staffing = Effort / Schedule
= 350948 / 320 FSP
= 1097 FSP

2. Semi-Detached -

Effort = $3.0 * (KLOC)^{1.12}$ PM
= $3.0 * (83000)^{1.12}$ PM
= 969368 PM

Schedule = $2.5 * (Effort)^{0.35}$ Months
= $2.5 * (969368)^{0.35}$ Months
= 311 Months

Productivity = DSI / Effort
= 83000 DSI / 969368 PM
= 0.086 DSI/PM

Average. Staffing = Effort / Schedule
= 969368 / 311 FSP
= 3117 FSP

3. Embedded -

Effort = $3.6 * (KLOC)^{1.20}$ PM
= $3.6 * (83000)^{1.20}$ PM
= 2878699 PM

Schedule = $2.5 * (Effort)^{0.32}$ Months
= $2.5 * (2878699)^{0.32}$ Months
= 292 Months

Productivity = DSI / Effort
= 83000 DSI / 2878699 PM
= 0.029 DSI/PM

Average. Staffing = Effort / Schedule
= 2878699 / 292 FSP
= 9858 FSP

INTERMEDIATE MODE:

COST DRIVER	DESCRIPTION
RELY	Required software reliability
DATA	Database size
CPLX	Product complexity
TIME	Execution time constraints
STOR	Main storage constraints
VIRT	Virtual machine volatility - degree to which the operating system changes
TURN	Computer turn around time
ACAP	Analyst capability
AEXP	Application experience
PCAP	Programmer capability
VEXP	Virtual machine (i.e. operating system) experience
LEXP	Programming language experience
MODP	Use of modern programming practices
TOOL	Use of software tools
SCED	Required development schedule

COCOMO - COST DRIVERS

		RATING					
	COST DRIVER	V.LOW	LOW	NOMINAL	HIGH	V.HIGH	EX. HIGH
(PRODUCT)	RELY	0.75	0.88	1.00	1.15	1.40	.
..	DATA	.	0.94	1.00	1.08	1.16	.
..	CPLX	0.70	0.85	1.00	1.15	1.30	1.65
(COMPUTER)	TIME	.	.	1.00	1.11	1.30	1.66
..	STOR	.	.	1.00	1.06	1.21	1.56
..	VIRT	.	0.87	1.00	1.15	1.30	.
..	TURN	.	0.87	1.00	1.07	1.15	.
(PERSONNEL)	ACAP	1.46	1.19	1.00	0.86	0.71	.
..	AEXP	1.29	1.13	1.00	0.91	0.82	.
..	PCAP	1.42	1.17	1.00	0.86	0.70	.
..	VEXP	1.21	1.10	1.00	0.90	.	.
..	LEXP	1.14	1.07	1.00	0.95	.	.
(PROJECT)	MODP	1.24	1.10	1.00	0.91	0.82	.
..	TOOL	1.24	1.10	1.00	0.91	0.83	.
..	SCED	1.23	1.08	1.00	1.04	1.10	.

Software Project	a1	a2	b1	b2
Organic	3.2	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Question:

Project A is to be an **83,000 DSI organic, semi-detached, embedded** software. It is in a mission-critical area, so the **reliability** is high (RELY=high=1.15).

Find the Effort, Schedule, Productivity, and average staffing required for the project.

Answer:

Using the formulas, we can estimate:

1. **Organic -**

Effort = $2.4 * (KLOC)^{1.05} * 1.15$ PM
= $2.4 * (83000)^{1.05}$ PM
= 403590 PM

Schedule = $2.5 * (Effort)^{0.38}$ Months
= $2.5 * (403590)^{0.38}$ Months
= 337 Months

Productivity = DSI / Effort
= 83000 DSI / 403590 PM
= 0.21 DSI/PM

Average. Staffing = Effort / Schedule
= 403590 / 337 FSP
= 1198 FSP

2. **Semi-Detached -**

Effort = $3.0*(KLOC)^{1.12} * 1.15$ PM
= $3.0*(83000)^{1.12}$ PM
= 1114773 PM

Schedule = $2.5*(Effort)^{0.35}$ Months
= $2.5*(1114773)^{0.35}$ Months

= 327 Months

Productivity = DSI / Effort
= 83000 DSI / 1114773 PM
= 0.075 DSI/PM

Average. Staffing = Effort / Schedule
= 1114773 / 327 FSP
= 3409 FSP

3. **Embedded -**

Effort = $3.6*(KLOC)^{1.20}$ * 1.15 PM
= $3.6*(83000)^{1.20}$ PM
= 3310504 PM

Schedule = $2.5*(Effort)^{0.32}$ Months
= $2.5*(3310504)^{0.32}$ Months
= 305 Months

Productivity = DSI / Effort
= 83000 DSI / 3310504 PM
= 0.025 DSI/PM

Average. Staffing = Effort / Schedule
= 3310504 / 305 FSP
= 10854 FSP

Question:

As an example of how the intermediate COCOMO model works, the following is a calculation of the estimated effort for an organic, semi-detached, embedded project of 83 KLOC. The cost drivers are set as follows:

Product cost drivers- **very high**

Computer cost drivers – **nominal**

Personnel cost drivers - **high**

Project cost drivers - **low**

Solution:

Product cost drivers (from the table) set **Very High** = 1.40 x 1.16 x 1.30

= 2.11

Computer cost drivers (from the table) set **nominal** = 1.00

Personnel cost drivers (from the table) set **high** = 0.86 x 0.91 x 0.86 x 0.95 x 0.90

= 0.57

Project cost drivers (from the table) set **low** = 1.10 x 1.10 x 1.08

= 1.30

product(cost drivers) = 2.11 x 1.00 x 0.57 x 1.30

= 1.56

For an **organic** project of 83 KLOC: **a** = 3.2 **b** = 1.05 **S** = 83

E = a(S)b x product(cost drivers)

E = 3.2 x (83)^1.05 x 1.56

E = 156.78 person-months

For a **semi-detached** project of 83 KLOC: **a** = 3.0 **b** = 1.12 **S** = 56

E = a(S)b x product(cost drivers)

E = 3.0 x (83)^1.12 x 1.56

E = 660.11 person-months

For a **semi-detached** project of 83 KLOC: **a** = 2.8 **b** = 1.20 **S** = 56

E = a(S)b x product(cost drivers)

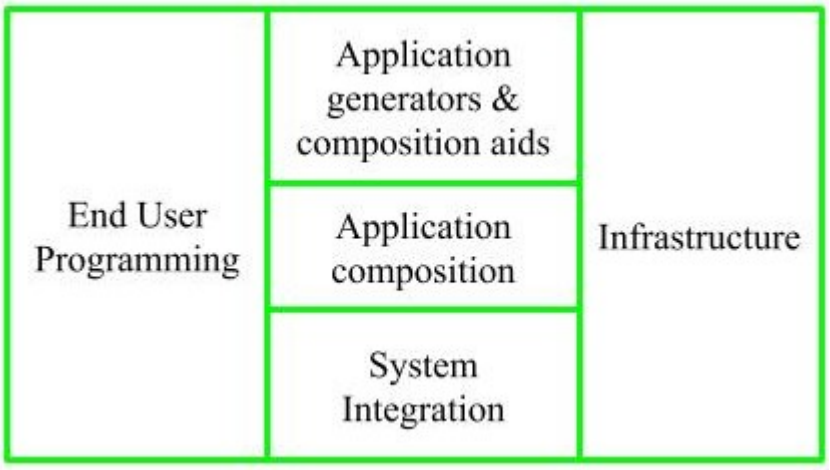
E = 2.8 x (83)^1.20 x 1.56

E = 877.36 person-months

COCOMO 2:

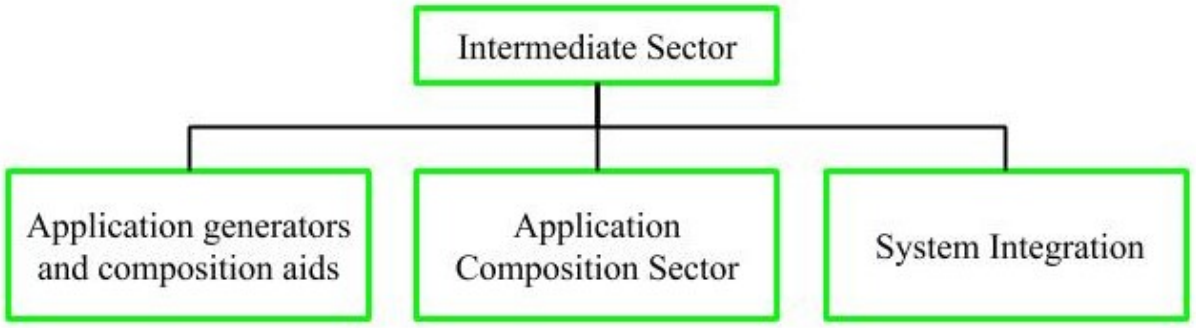
COCOMO-II is the revised version of the original Cocomo (Constructive Cost Model) and was developed at the University of Southern California. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.

It consists of three sub-models:



1. End-User Programming:
Application generators are used in this sub-model. End users write the code by using these application generators.
Examples are spreadsheets, report generators, etc.

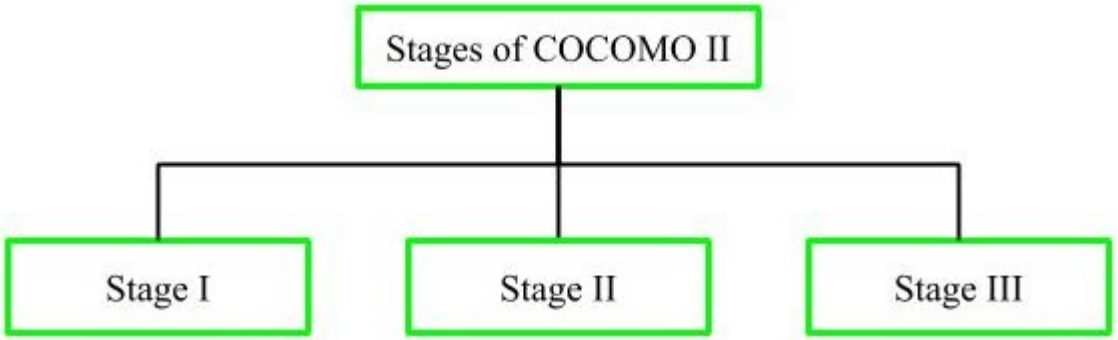
2. Intermediate Sector:



(a). Application Generators and Composition Aids –
This category will create largely prepackaged capabilities for user programming. Their product will have many reusable components. Typical firms operating in this sector are Microsoft, Lotus,
Oracle, IBM, Borland, Novell.

(b). Application Composition Sector –
This category is too diversified and to be handled by prepackaged solutions. It includes GUI, Databases, and domain-specific components such as financial, medical or industrial process control packages.
(c). System Integration –
This category deals with large-scale and highly embedded systems.

3. Infrastructure Sector:



This category provides infrastructure for software development like Operating Systems, Database Management Systems, User Interface Management Systems, Networking Systems, etc.

Stages of COCOMO II:

Stage-I:
It supports the estimation of prototyping. For this, it uses the Application Composition Estimation Model. This model is used for the prototyping stage of application generator and system integration.
Stage-II:
It supports estimation in the early design stage of the project when we less know about it. For this, it uses the Early Design Estimation Model. This model is used in the early design stage of application generators, infrastructure, and system integration.
Stage-III:
It supports estimation in the post-architecture stage of a project. For this, it uses the Post Architecture Estimation Model. This model is used after the completion of the detailed architecture of the application generator, infrastructure, and system integration.

Certainly,

QUESTION: let's adapt the question for an e-commerce website project. In this scenario, we'll consider the development of an e-commerce website with 2 main pages, each containing 4 different views, and 10 data tables for managing products, users, orders, and other related information. The website is designed to handle traffic from 1 server and 5 clients. Additionally, the project includes the generation of 3 different reports, each consisting of 7 sections, all of which are derived from the data stored in the 10 data tables. We'll also assume a 20% reuse of object points, and that the developer's experience and capability in a similar environment is Nominal.

Step 1: Determine the number of screens and reports.

- Number of screens = 2 main pages * 4 different views = 8 screens
- Number of reports = 3 reports

Step 2: Determine the factors for screens and reports.

For Screens:

- Number of views per screen = 4
- Number of data tables per screen = 10
- Number of servers = 1
- Number of clients = 5
- Complexity level for each screen = Nominal (since developer experience is considered Nominal)

For Reports:

- Number of sections per report = 7
- Number of data tables per report = 10
- Number of servers = 1
- Number of clients = 5
- Complexity level for each report = Nominal

Step 3: Assign complexity weights.

Now, you can assign complexity weights based on the provided information. Complexity weights are typically assigned on a scale from Low to High. In this case, since you've mentioned that the developer's experience is Nominal, we can assign weights as follows:

- Complexity weight for each screen = 5 (Nominal)
- Complexity weight for each report = 5 (Nominal)

Step 4: Calculate Object Point Count

First, let's calculate the Object Point Count for your e-commerce website project using the complexity weights determined earlier:

For Screens:

Object Point Count for Screens = (Number of screens * Complexity weight for each screen)

Object Point Count for Screens = (8 screens * 5) = 40 Object Points

For Reports:

Object Point Count for Reports = (Number of reports * Complexity weight for each report)

Object Point Count for Reports = (3 reports * 5) = 15 Object Points

Now, we can calculate the total Object Point Count:

Total Object Point Count = Object Point Count for Screens + Object Point Count for Reports

Total Object Point Count = 40 + 15 = 55 Object Points

Given that there is a 20% reuse of object points, we can calculate the Non-Reused Object Points (NOP):

Non-Reused Object Points (NOP) = [Total Object Points * (100 - % Reuse)] / 100

Non-Reused Object Points (NOP) = [55 * (100 - 20)] / 100

Non-Reused Object Points (NOP) = [55 * 80] / 100

Non-Reused Object Points (NOP) = 44

Step 6: Developer’s Experience and Capability

The developer’s experience and capability are Nominal

Using the information given about the developer and productivity rate table

Productivity rate (PROD) of given project = 13

Step 7: Effort Calculation

Effort = NOP / PROD

Effort = 44 / 13

Effort ≈ 3.38 person-days (rounded to two decimal places)