K. J. Somaiya College of Engineering, Mumbai-77

(A Constituent college of Somaiya Vidyavihar University)

Batch: A1 **Roll No.: 16010120015**
16010120013
16010120020

Experiment / assignment / tutorial No. 2

TITLE: Project Metric estimations for Mini Project

AIM: To enable the students learn different techniques for performing software size and cost estimation

Expected Course outcome of Experiment:

CO 1: Understand the software development process and Estimate different types of resources for the given project.

Books/ Journals/ Websites referred:

1. Roger Pressman, “Software Engineering”, sixth edition, Tata McGraw Hill.
2. http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html
3. http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html

Pre Lab/ Prior Concepts:

Software projects have tendency of going past their deadline, going over budget, or both. The problem lies in the estimation of the amount of effort required for the development of a project. The cost estimation is usually dependent upon the size estimate of the project, which may use lines of code or function points as metrics. There are several different techniques for performing software cost estimation, including expert judgement and algorithmic models. Estimation by expert judgement is a common way of estimating the effort required for a project. Unfortunately, this method of estimation does not emphasize re-estimation during the project life cycle, which is an important part of project tracking, because it allows the estimates to be improved during the project life cycle. The quality of a cost estimation model is not so much attributed to the initial estimate, but rather the speed at which the estimates converges to the actual cost of the project. COCOMO is a popular algorithmic model for cost estimation whose cost factors can be tailored to the individual development environment, which is important for the accuracy of the cost estimates. More than one method of cost estimation should be done so that there is some comparison available for the estimates. This is especially important for unique projects. Cost estimation must be done more diligently throughout the project life cycle so that in the future there are fewer surprises and unforeseen delays in the release of a product.

Estimation of size and cost of the developing project is required for the following major decision situations

- Financial decisions involving a software development effort
- Setting project budgets and schedules as a basis for planning and control
- Deciding on or negotiating tradeoffs among software cost, schedule, functionality, performance or quality factors
- Making software cost and schedule risk management decisions
- Deciding which parts of a software system to develop, reuse, lease, or purchase
- Making legacy software inventory decisions: what parts to modify, phase out, outsource, etc
- Deciding how to implement a process improvement strategy, such as that provided in the SEI CMM

Defining Cost estimation:

Cost estimation can be defined as the approximate judgement of the costs for a project. Cost estimation will never be an exact science because there are too many variables involved in the calculation for a cost estimate, such as human, technical, environmental, and political. Furthermore, any process that involves a significant human factor can never be exact because humans are far too complex to be entirely predictable. Furthermore, software development for any fair-sized project will inevitably include a number of tasks that have complexities that are difficult to judge because of the complexity of software systems.

Cost estimation is usually measured in terms of effort. The most common metric used is person months or years (or man months or years). The effort is the amount of time for one person to work for a certain period of time. It is important that the specific characteristics of the development environment are taking into account when comparing the effort of two or more projects because no two development environments are the same. A clear example of differences in development environments are the amount of time people work in different countries; the typical workweek in North America is 40 hours per week, while in Europe the typical workweek is 35 hours per week. Thus, when comparing a project from North America with a project from Europe, a conversion factor would have to be used to all for an accurate comparison. Different variables can be used for cost estimation, which leads to a difficulty when comparing projects if standard models or tools are not used. For example, a cost estimate can include factors from management, development (e.g., training, quality assurance), and other areas specific to an organization.

Estimator:

The people who do the cost estimates could be either directly or indirectly responsible for the implementation for a project, such as a developer or manager, respectively. Someone who has knowledge of the organization and previous projects could use an analogy-based approach to compare the current project with previous projects, which is a common method of estimation for small organizations and small projects. The historical data is often limited to the memory of the estimator. In this case, the estimator would need to be experienced and would likely have been with the company for awhile.

Some people believe it is better if the estimates are done by outsiders so that there is less chance of bias. It is true that people outside an organization will likely have to deal with fewer company politics than people within the organization. For example, the developer for a company may want to please the manager and so give an estimate that is overly-optimistic. The disadvantage of having an outside estimate is that the person would have less knowledge of the development environment, especially if the person is from outside the company. An empirical method of estimation would then be required, such as the Constructive Cost Model (COCOMO. Empirical methods of estimation can be used by all types of estimators. There may be some resistance to using an empirical method of estimation because there may be some question on whether a model could outperform an expert. People who are accurate estimators are rare in our experience, and so it is best to get the opinion of several people or tools.

Cost estimation using different COCOMO models:

(Attach screen shots of the size/cost estimations using different COCOMO tools. Justify estimation details)

COCOMO Model using KLOC:

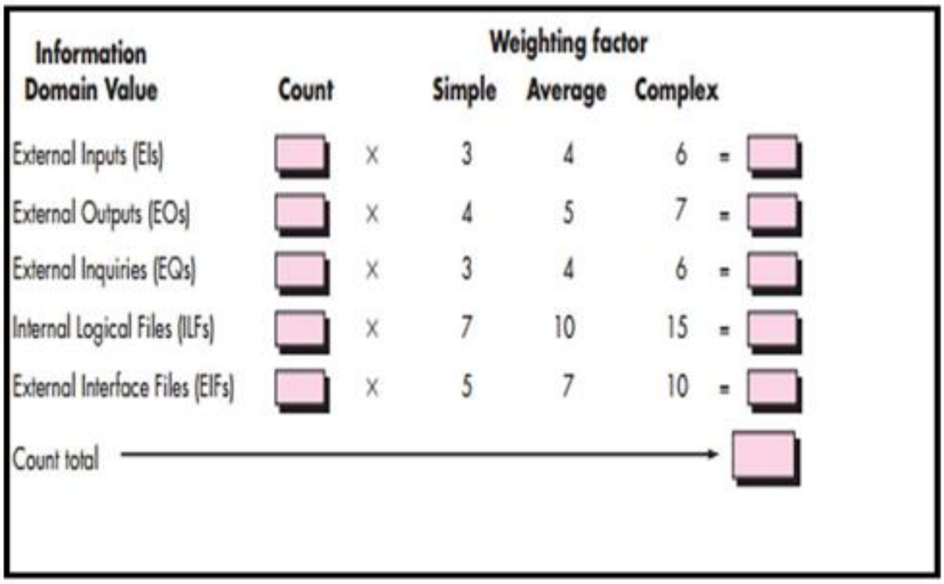
Lines of code: 8000

KLOC: 8

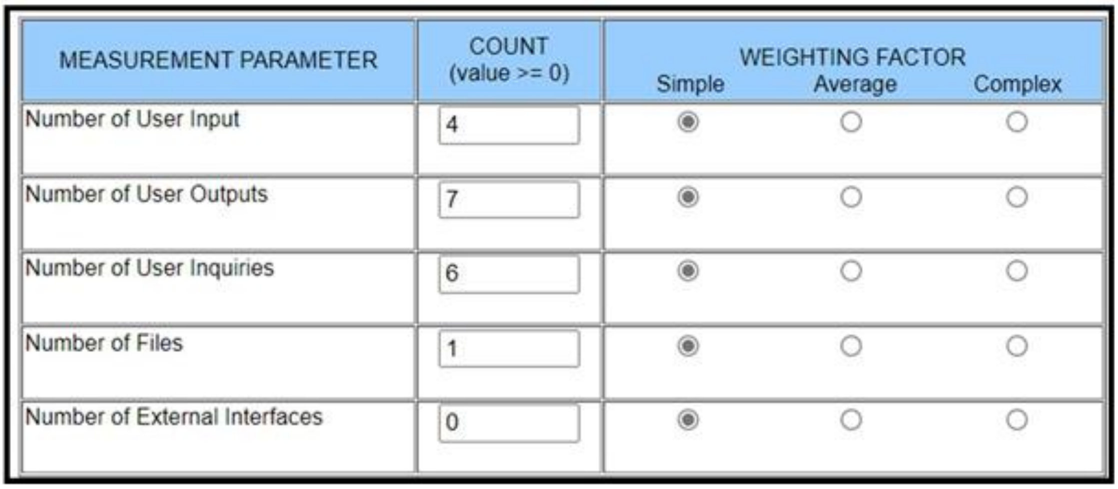
Model used	Formula	Result (efforts in function months)
Walston-Felix Model	*E = 5.2*(KLOC)0.91*	*34.49*
Bailey-Basili Model	*E = 5.5 + 0.73*(KLOC)1.16*	*13.64*
Boehm Simple Model	*E = 3.2 * (KLOC)1.05*	*28.40*

COCOMO model using Function Pointer:

Calculation of FP:

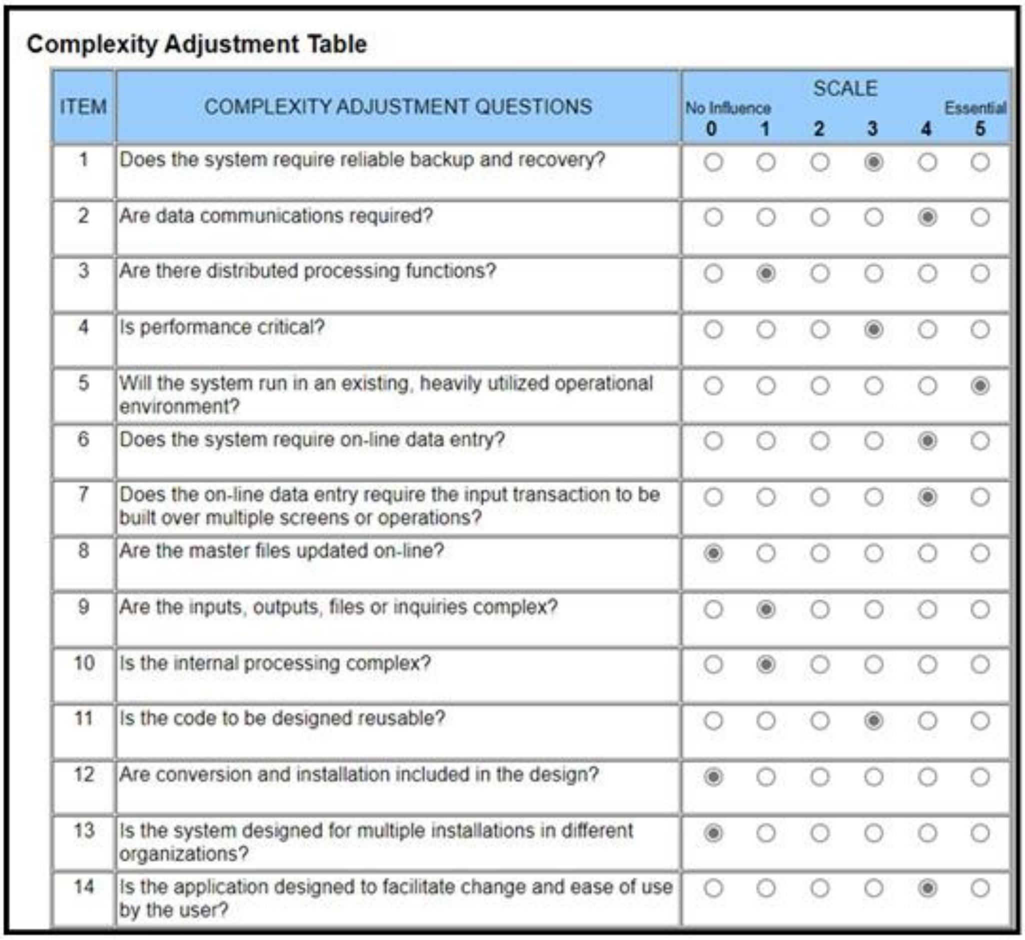


Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)	□	x	3	4	6	=	□
External Outputs (EOs)	□	x	4	5	7	=	□
External Inquiries (EQs)	□	x	3	4	6	=	□
Internal Logical Files (ILFs)	□	x	7	10	15	=	□
External Interface Files (EIFs)	□	x	5	7	10	=	□
Count total						→	□



MEASUREMENT PARAMETER	COUNT (value >= 0)	WEIGHTING FACTOR		
		Simple	Average	Complex
Number of User Input	4	●	○	○
Number of User Outputs	7	●	○	○
Number of User Inquiries	6	●	○	○
Number of Files	1	●	○	○
Number of External Interfaces	0	●	○	○

Count Total = 65



Complexity Adjustment Table

ITEM	COMPLEXITY ADJUSTMENT QUESTIONS	SCALE					
		No Influence					Essential
		0	1	2	3	4	5
1	Does the system require reliable backup and recovery?	○	○	○	●	○	○
2	Are data communications required?	○	○	○	○	●	○
3	Are there distributed processing functions?	○	●	○	○	○	○
4	Is performance critical?	○	○	○	●	○	○
5	Will the system run in an existing, heavily utilized operational environment?	○	○	○	○	○	●
6	Does the system require on-line data entry?	○	○	○	○	●	○
7	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	○	○	○	○	●	○
8	Are the master files updated on-line?	●	○	○	○	○	○
9	Are the inputs, outputs, files or inquiries complex?	○	●	○	○	○	○
10	Is the internal processing complex?	○	●	○	○	○	○
11	Is the code to be designed reusable?	○	○	○	●	○	○
12	Are conversion and installation included in the design?	●	○	○	○	○	○
13	Is the system designed for multiple installations in different organizations?	●	○	○	○	○	○
14	Is the application designed to facilitate change and ease of use by the user?	○	○	○	○	●	○

Σ Fi = 33

Calculation of Function Point

FP = countTotal * [0.65 + 0.01 (Σ Fi)]

= 65*[0.65 + 0.01 * 33]

FP = 63.69

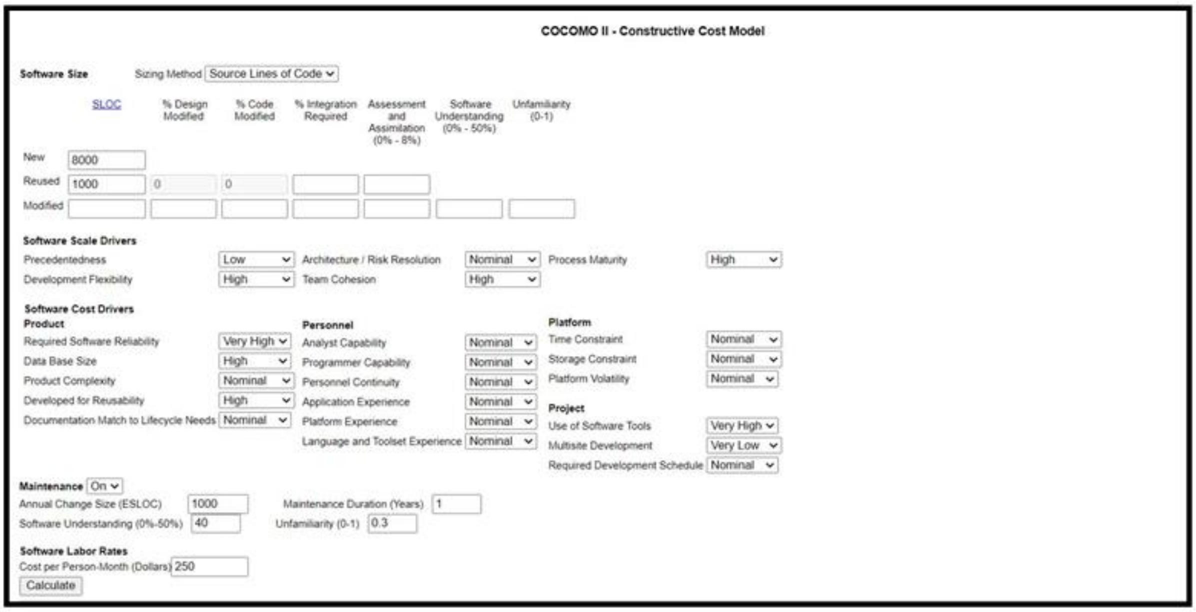
Model	Formula	Result
Kemerer Model	E = -37 + 0.96*FP	24.1424
Small Project Regression Model	E = -12.88 + 0.405*FP	12.914

COCOMO-II model:

Sizing method: Lines of Code

Webiste used for screenshot :

http://softwarecost.org/tools/COCOMO/



COCOMO II - Constructive Cost Model

Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	8000						
Reused	1000	0	0				
Modified							

Software Scale Drivers

Precedentedness	Low	Architecture / Risk Resolution	Nominal	Process Maturity	High
Development Flexibility	High	Team Cohesion	High		

Software Cost Drivers

Product		Personnel		Platform	
Required Software Reliability	Very High	Analyst Capability	Nominal	Time Constraint	Nominal
Data Base Size	High	Programmer Capability	Nominal	Storage Constraint	Nominal
Product Complexity	Nominal	Personnel Continuity	Nominal	Platform Volatility	Nominal
Developed for Reusability	High	Application Experience	Nominal	**Project**	
Documentation Match to Lifecycle Needs	Nominal	Platform Experience	Nominal	Use of Software Tools	Very High
		Language and Toolset Experience	Nominal	Multisite Development	Very Low
				Required Development Schedule	Nominal

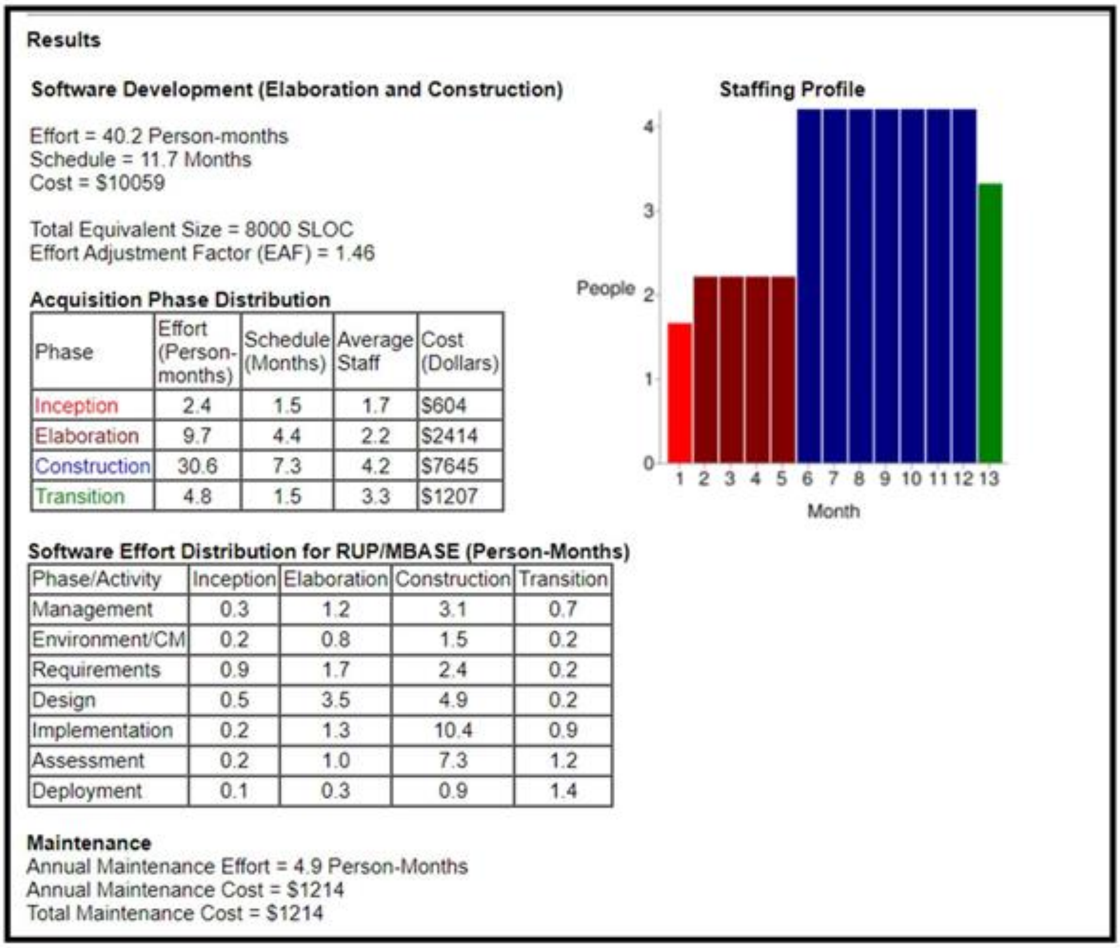
Maintenance On

Annual Change Size (ESLOC) 1000 Maintenance Duration (Years) 1

Software Understanding (0%-50%) 40 Unfamiliarity (0-1) 0.3

Software Labor Rates

Cost per Person-Month (Dollars) 250



Results

Software Development (Elaboration and Construction)

Effort = 40.2 Person-months
Schedule = 11.7 Months
Cost = $10059

Total Equivalent Size = 8000 SLOC
Effort Adjustment Factor (EAF) = 1.46

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	2.4	1.5	1.7	$604
Elaboration	9.7	4.4	2.2	$2414
Construction	30.6	7.3	4.2	$7645
Transition	4.8	1.5	3.3	$1207

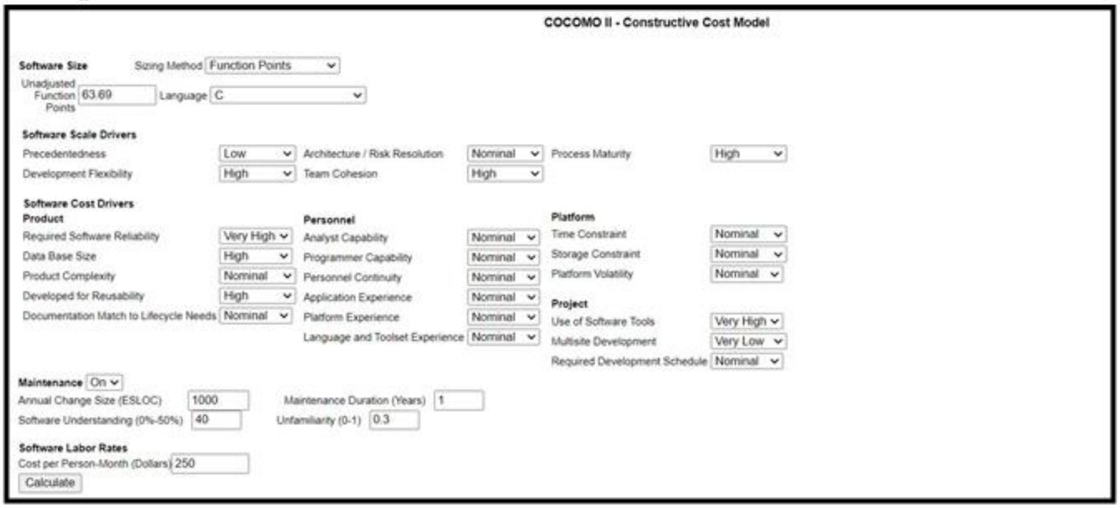
Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.3	1.2	3.1	0.7
Environment/CM	0.2	0.8	1.5	0.2
Requirements	0.9	1.7	2.4	0.2
Design	0.5	3.5	4.9	0.2
Implementation	0.2	1.3	10.4	0.9
Assessment	0.2	1.0	7.3	1.2
Deployment	0.1	0.3	0.9	1.4

Maintenance

Annual Maintenance Effort = 4.9 Person-Months
Annual Maintenance Cost = $1214
Total Maintenance Cost = $1214

Sizing method: Function Point:



COCOMO II - Constructive Cost Model

Software Size Sizing Method: Function Points

Unadjusted Function Points: 63.69 Language: C

Software Scale Drivers

Precedentedness	Low	Architecture / Risk Resolution	Nominal	Process Maturity	High
Development Flexibility	High	Team Cohesion	High		

Software Cost Drivers

Product

Required Software Reliability	Very High
Data Base Size	High
Product Complexity	Nominal
Developed for Reusability	High
Documentation Match to Lifecycle Needs	Nominal

Personnel

Analyst Capability	Nominal
Programmer Capability	Nominal
Personnel Continuity	Nominal
Application Experience	Nominal
Platform Experience	Nominal
Language and Toolset Experience	Nominal

Platform

Time Constraint	Nominal
Storage Constraint	Nominal
Platform Volatility	Nominal

Project

Use of Software Tools	Very High
Multisite Development	Very Low
Required Development Schedule	Nominal

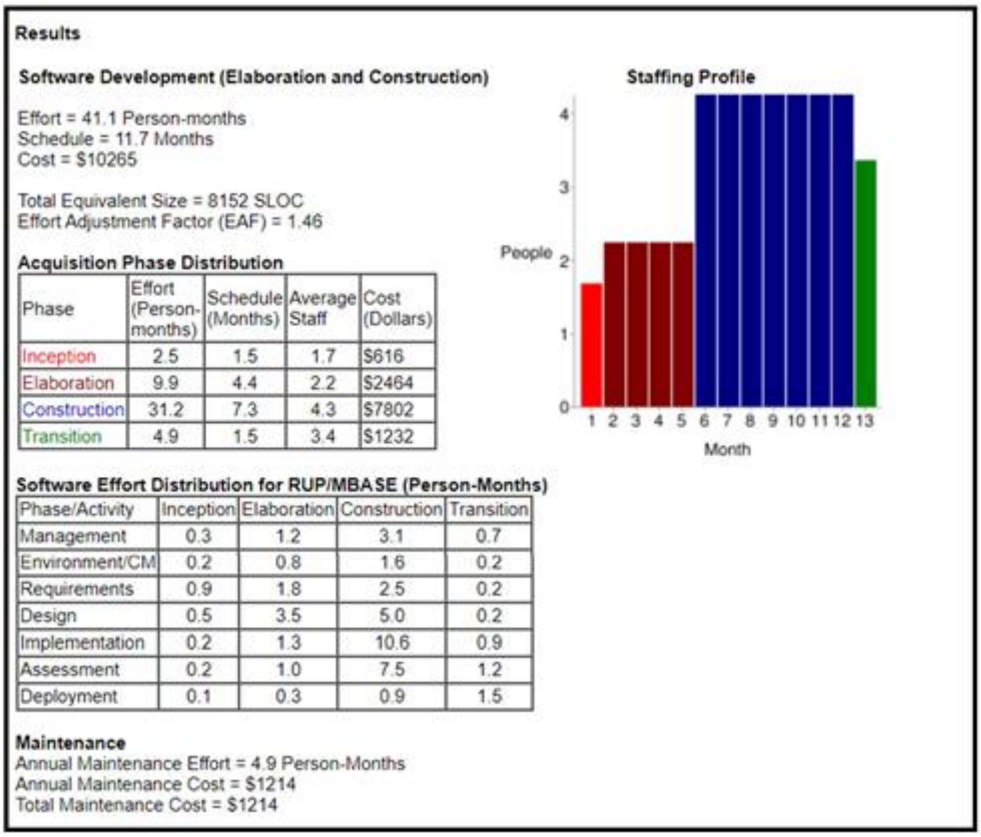
Maintenance On

Annual Change Size (ESLOC): 1000 Maintenance Duration (Years): 1

Software Understanding (0%-50%): 40 Unfamiliarity (0-1): 0.3

Software Labor Rates

Cost per Person-Month (Dollars): 250



Results

Software Development (Elaboration and Construction)

Effort = 41.1 Person-months
Schedule = 11.7 Months
Cost = $10265

Total Equivalent Size = 8152 SLOC
Effort Adjustment Factor (EAF) = 1.46

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	2.5	1.5	1.7	$616
Elaboration	9.9	4.4	2.2	$2464
Construction	31.2	7.3	4.3	$7802
Transition	4.9	1.5	3.4	$1232

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.3	1.2	3.1	0.7
Environment/CM	0.2	0.8	1.6	0.2
Requirements	0.9	1.8	2.5	0.2
Design	0.5	3.5	5.0	0.2
Implementation	0.2	1.3	10.6	0.9
Assessment	0.2	1.0	7.5	1.2
Deployment	0.1	0.3	0.9	1.5

Maintenance

Annual Maintenance Effort = 4.9 Person-Months
Annual Maintenance Cost = $1214
Total Maintenance Cost = $1214

Software Size :

1. The aim is to build a web application in which different pieces of code are put together in separate files, the estimated lines of code for our project is 8000.

2. Reusable code mainly comprises the frontend development code, in which templates will be reused from previous versions. Authentication Engine and ML Engine could also be trained and reused. Around 1000 lines of code is approximated to be reused.

Software Scale Drivers:

1. Precedentedness: Very low previous experience of development of similar products is required.

2. Architecture / Risk Resolution: There is low-nominal risk in architecture as the flow of the Spotify website is quite straightforward and there is interaction with the Server.

3. Process Maturity: The process maturity is high because new user need to be registered and it should be handled and displayed to user using the Spotify in the least time possible.

4. Development Flexibility: A conventional Full-stack development process is to be used for the development, hence very flexible.

5. Team Cohesion: A team with strong coordination and cooperation is required to build this product on various platforms.

Software Cost Drivers:

Product:

1. Required Software Reliability: Required softwares for the project are frame-works which are popularly used like Django - A python web development library. The database used will be a relational database like SQLite. Any changes/updates/resolution of software is dependent on this platform which leads to high reliability.

2. Database Size: High as the type of data that is to be stored in it consists of mp4 file format and also there is requirement of separate tables for storing product of a particular type. The size of the database is only going to increase with an increasing number of users.

3. Product Complexity: Product internally has 2 parts, First Website/Interface part and Second data processing(backend) part. Thus for any change it is easy but the connections of these parts and their internal configurations, architecture is simple in complexity but have a long term benefit for updates and resolutions.

4. Developed for Reusability: High as in today's world the frameworks built on Python for web development will improve from the previous ones and for users to get good experience this algorithm will have to be updated there and thus the product will be reused and just the algorithms are changed for making the user experience better.

Platform:

1. Time Constraint: Nominal time constraints are required for the product. The posts made by a user should not take very long time to start playing on the website.

2. Storage Constraint: There is nominal storage constraint as data regarding the users need to be stored rather than data of every visiting user.

3. Platform Volatility: Changes to product (Subsciption) are required only in case there is replacement of the current algorithm with a better one or some minor bug.

Project:

1. Use of Software Tools: Various tools like VsCode, SQLite Databases., Apache Storm , Apache Beam , Google BigQuery, Python Programming Language and libraries are used for software development.

2. Multisite Development: Very high.

3. Required Development Schedule: A development schedule is required because some functionalities cannot be made before others. For eg. Product functionality cannot be made before user authentication.

Personnel:

1. Analyst Capability: Personnel with nominal analytical thinking can work on this project.

2. Programmer Capability: The capability of the programmer or developer must be very high and he /she must be willing to learn new things related to languages, platform, application very fast.

3. Personnel Continuity: The personnel continuity can be nominal as in this project, a new person can start listening from where the song was paused immediately.

4. Application Experience: Personnel should have at least knowledge about the application being built.

5. Platform Experience: The personnel should have worked prior on this platform and must be willing to learn new things about it.

6. Language and Toolset Experience: The basic knowledge about the languages is required.

Maintenance:

1. Annual Change Size (ESLOC): The number of lines of code added or removed every year. This comes out to be roughly 1,00,000 for the given project.

2. Maintenance Duration (Years): 2 years

3. Software Understanding (0%-50%): 40% of software understanding for the given project.

4. Unfamiliarity (0-1): 0.3 or 30% of the code or software or tools used is unfamiliar to the coders.

Software Labour Rates:

1. Cost per Person-Month - 2500$

Conclusion: In this experiment, we performed COCOMO model analysis using function pointers and the kloc approach. A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem. The project under consideration, holds well for the above mentioned metrics and hence we can consider spotify to be an Semi-detached type of software.

Post Lab Descriptive Questions

1. **You are appointed as an estimator to find out efforts required to implement following project. The goal of the project is to create a database of all Hindi films released since 2000. The software would allow one to generate a list of top ten hit films, top ten flop films, best comedy films, and so on. Using your prior experience you have decided the approximate sizes of each module of the software as follows:**
   Data entry (0.9 KDSI)
   Data update (0.7 KDSI)
   Query (0.9 KDSI)
   Report generation and display (2 KDSI)

Also take into consideration the following cost drivers with their ratings:

Storage constraints (Medium)
Experience in developing similar software (High)
Programming capabilities of the developers (Low)
Application of software engineering methods (High)
Use of software tools (Medium)
(All other cost drivers have nominal rating).

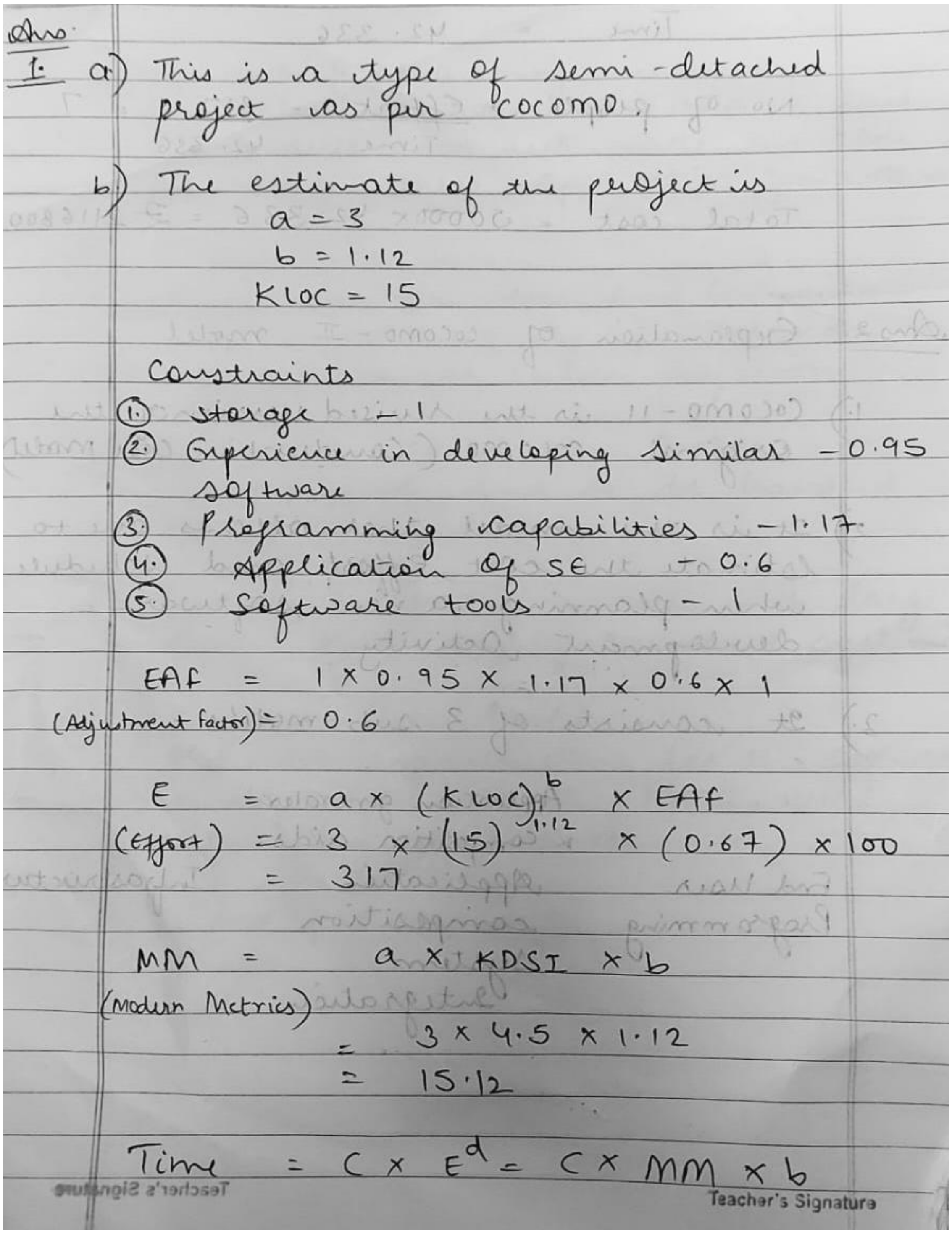
Now answer the following:

Applying intermediate COCOMO estimate the minimum size of the team you would require to develop this system
Assuming that your client would pay Rs. 50,000 per month of development, how much would be the likely billing?

a) **Identify type of a project as per COCOMO**
b) **Prepare an estimate of required effort and cost**

Limitations: Values presented here are arbitrary and doesn't relate to real life

2. Explain COCOMO II model.



Ans.

1. a) This is a type of semi-detached project as per COCOMO.

b) The estimate of the project is

$a = 3$

$b = 1.12$

$KLOC = 15$

Constraints

(1.) storage – 1
(2.) Experience in developing similar software – 0.95
(3.) Programming capabilities – 1.17
(4.) Application of SE – 0.6
(5.) Software tools – 1

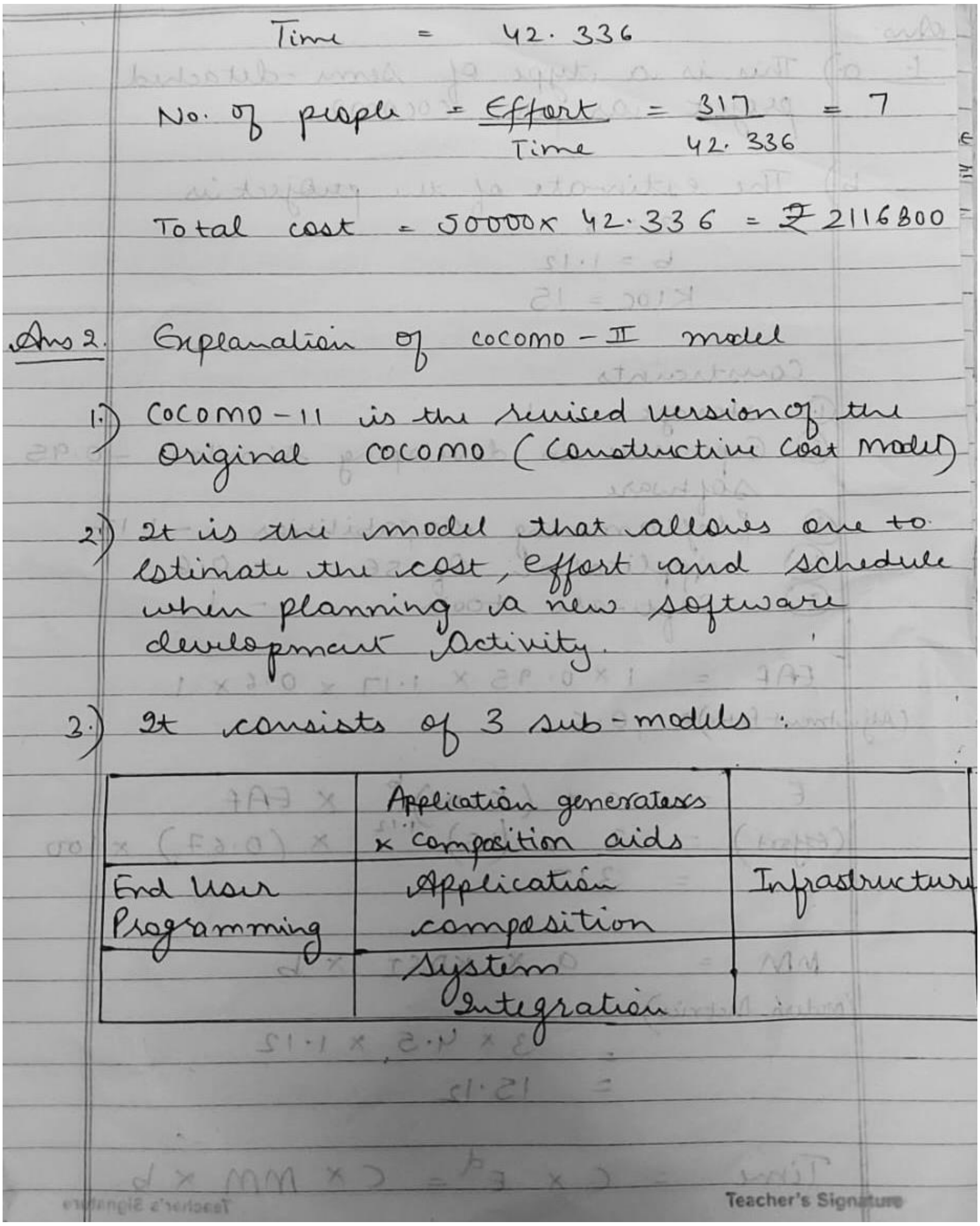
EAF $= 1 \times 0.95 \times 1.17 \times 0.6 \times 1$
(Adjustment factor) $= 0.6$

E $= a \times (KLOC)^b \times EAF$
(Effort) $= 3 \times (15)^{1.12} \times (0.67) \times 100$
$= 317$

MM $= a \times KDSI \times b$
(Modern Metrics)
$= 3 \times 4.5 \times 1.12$
$= 15.12$

Time $= C \times E^d = C \times MM \times b$

Teacher's Signature



Time = 42.336

No. of people = $\frac{\text{Effort}}{\text{Time}} = \frac{317}{42.336} = 7$

Total cost = 50000x 42.336 = ₹ 2116800

Ans 2. Explanation of COCOMO – II model

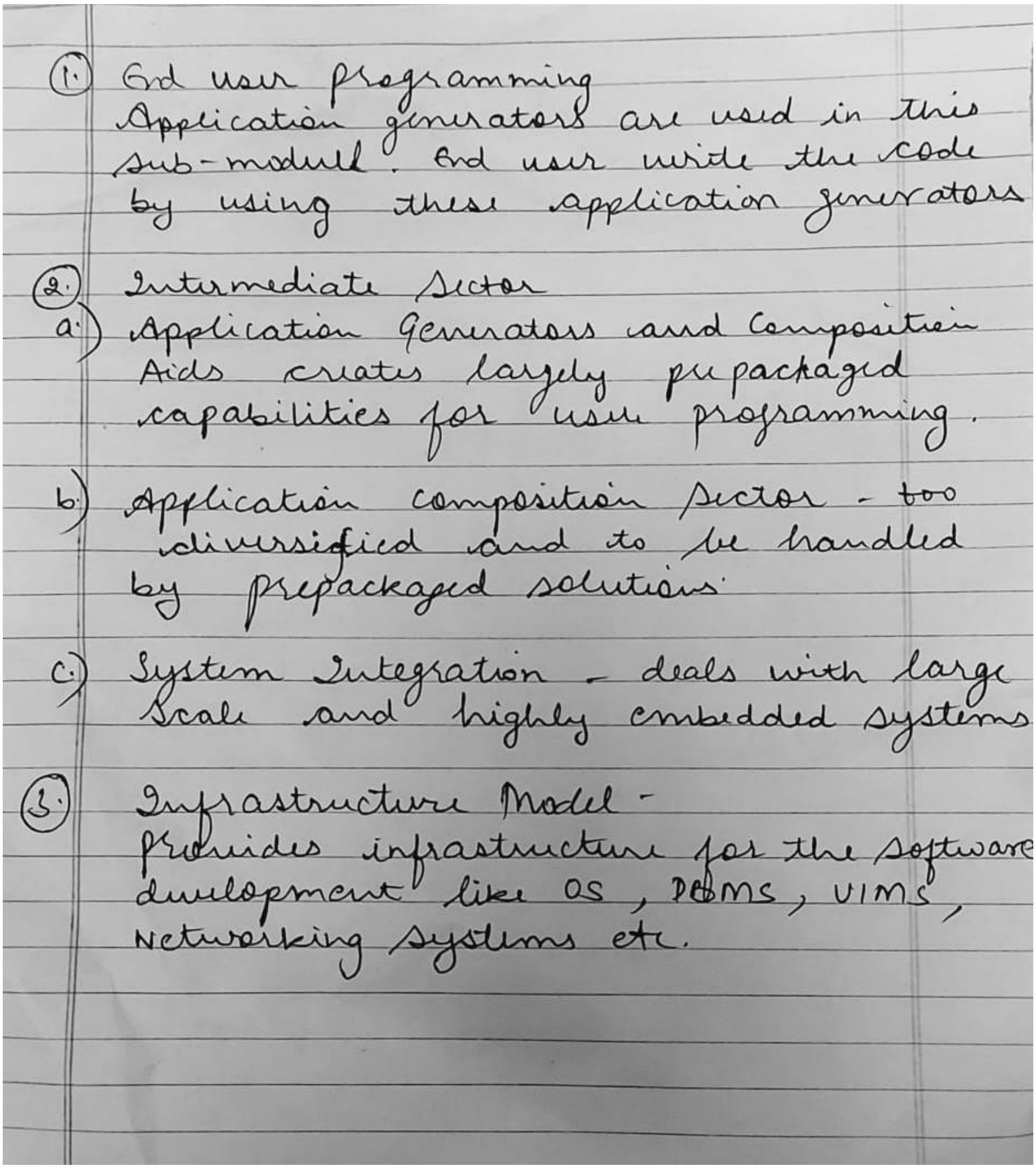
1.) COCOMO – 11 is the revised version of the Original COCOMO (Constructive Cost Model)

2.) It is the model that allows one to estimate the cost, effort and schedule when planning a new software development Activity.

3.) It consists of 3 sub-models :

	Application generators x composition aids	
End User Programming	Application composition	Infrastructure
	System Integration	

Teacher's Signature



1. End user programming
Application generators are used in this sub-module. End user write the code by using these application generators

2. Intermediate Sector
a.) Application Generators and Composition Aids creates largely prepackaged capabilities for user programming.

b.) Application composition sector - too diversified and to be handled by prepackaged solutions.

c.) System Integration - deals with large Scale and highly embedded systems

3. Infrastructure Model -
Provides infrastructure for the software development like OS, DBMS, UIMS, Networking systems etc.