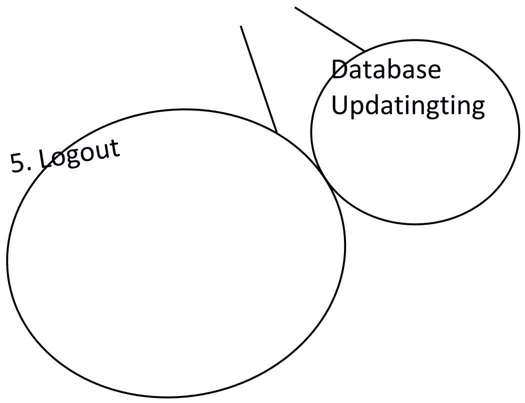

**Conclusion:** Hence using DFD for OARS we have studied the flow of data through the system and visualization of data processing as one of the way to create detailed architectural flow for selected Case study.

.

## Experiment No:05

**Aim:** Use of metrics to estimate the cost**.**

**Theory:**

**1) COCOMO Model:**

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current project characteristics.
COCOMO was first published in Boehm's 1981 book Software Engineering Economics as a model for estimating effort, cost, and schedule for software projects. It drew on a study of 63 projects at TRW Aerospace where Boehm was Director of Software Research and Technology. The study examined projects ranging in size from 2,000 to 100,000 lines of code, and programming languages ranging from assembly to PL/I. These projects were based on the waterfall model of software development which was the prevalent software development process in 1981.

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level,

-**Basic COCOMO** is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers).

-**Intermediate COCOMO** takes these Cost Drivers into account

-**Detailed COCOMO** additionally accounts for the influence of individual project phases.

COCOMO applies to three classes of software projects:

**Organic projects** - "small" teams with "good" experience working with "less than rigid" requirements

**Semi-detached projects** - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements

**Embedded projects** - developed within a set of "tight" constraints .it is also combination of organic and semi-detached projects.(hardware, software, operational, ......)

Cocomo equations takes the form:

Efforts applied: E= (ab) $(KLOC)^{(bb)}$ PM

Development time: (D) = (cb)$(E)^{(db)}$ PM

Person required = Effort applied /Development Time

P=E/D

ab, cb & the exponents bb & db are given in table

The co-efficients ab, bb, cb and db are given in the following table

| Software project | Ab | Bb | Cb | db |
|---|---|---|---|---|
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi-detached | 3 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

In our project:

Efforts required (E) = (ab) $(KLOC)^{(bb)}$ =

3 *$(12000)^{(1.12)}$ = 111123.0749 per month

Development time (D) = (cb)$(E)^{(db)}$ =

2.5 *$(111123.0749)^{(0.35)}$ = 145.872 per month

Person required = E/D = 761.7856

= 762

**2) COCOMO II:**

COCOMO II is tuned to modern software life cycles. The original COCOMO model has been very successful, but it doesn't apply to newer software development practices as well as it does to traditional practices. COCOMO II targets the software projects of the 1990s and 2000s, and will continue to evolve over the next few years.

COCOMO II is really three different models:

**The Application Composition Model**

Suitable for projects built with modern GUI-builder tools. Based on new Object Points.

**The Early Design Model**

You can use this model to get rough estimates of a project's cost and duration before you've determined its entire architecture.

**The Post-Architecture Model.**

This is the most detailed COCOMO II model. You'll use it after you've developed your project's overall architecture

**3) FUNCTION POINTS:**

Function oriented metrics focus on program "functionality" or "utility". Albrecht first proposed function point method, which is a function oriented productivity measurement approach.

Five information domain characteristics are determined and counts for each are provided and recorded in a table.

- Number of user inputs
- Number of user outputs
- Number of user inquires
- Number of files
- Number of external interfaces

Once the data have been collected, a complexity value is associated with each count. Each entry can be simple, average or complex. Depending upon these complexity values is calculated.

To compute function points, we use

*FP* = count-total * [0.65 + 0.01 * SUM (Fi)]

Where, count-total is the sum of all entries obtained

*Fi*(I= 1 to 14) are complexity adjustment values based on response to questions(1-14) given below. The constant values in the equation and the weighing factors that are applied to information domain are determined empirically.

*Fi*

1. Does the system require reliable backup and recovery? 5+2+0+3
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 entry?
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?
8. Are the inputs, outputs, files, or inquiries complex?
9. Is the internal processing complex?
10. Is the code designed to be reusable?
11. Are master files updated on-line?
12. Are conversion and installations 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?

**4) COMPONENTS OF FUNCTION POINTS:**

| | **Information domain Value** | **Simple** | **Average** | **Complex** |
|---|---|---|---|---|
| 1 | No. of user i/p | 2 | 4 | 6 |
| 2 | No. of user o/p | 3 | 5 | 7 |
| 3 | No. of Inquiries | 2 | 4 | 6 |
| 4 | No. of Files | 5 | 10 | 15 |
| 5 | No. of External interfaces | 4 | 7 | 1 |

2*2+2*3+3*2+4*5+3*4=

A simple example:

**No of user inputs**

3 simplex 2=6

4 average x 4 =16

1 complex x 6 = 6

**No of user outputs**

6 average x 5 =30

2 complex X 7= 14

**No of inquiries**

8 aberage x 4 =32

**No of files**

5 complex x 15=75

**No of External interfaces**

3 average x 7=214 complex x 10 = 40

Unadjusted function point = 240

Project adjustment factor i.e. ∑(Fi)=17

Adjustment calculation:

Adjusted FP = count x [0.65 +(0.01 x ∑(Fi))]=240 x [0.65 + 0.01 x 17)] =240 x [0.82] =197

**Conclusion:** In this way the cost of the project is calculated successfully.