

**NAME:- ANAS KAPADIA   ROLL NO:- 22DCO01 BATCH:- 4**

**Experiment No. 07**

**LOC and FP Analysis**

**Aim :** To estimate the cost of a project using software metrics (LOC).

# **Theory :**

A software metric is a measure of software characteristics which are measurable or countable. Software metrics are valuable for many reasons, including measuring software performance, planning, work items, measuring productivity, and many other uses. Within the software development process, many metrics are all connected. Software metrics are similar to the four functions of management: Planning, Organization, Control, or Improvement.

# **LOC Metrics :**

It is one of the earliest and simpler metrics for calculating the size of the computer program. It is generally used in calculating and comparing the productivity of programmers. These metrics are derived by normalizing the quality and productivity measures by considering the size of the product as a metric.

Following are the points regarding LOC measures:

* In size-oriented metrics, LOC is considered to be the normalization value.
* It is an older method that was developed when FORTRAN and COBOL programming were very popular.
* Productivity is defined as KLOC / EFFORT, where effort is measured in person-months.
* Size-oriented metrics depend on the programming language used.
* As productivity depends on KLOC, assembly language code will have more productivity.
* LOC measure requires a level of detail which may not be practically achievable.
* The more expressive the programming language, the lower is the prouctivity.

The LOC method of measurement does not apply to projects that deal with visual (GUI-based)programming. As already explained, Graphical User Interfaces (GUIs) use forms basically.

* The LOC metric is not applicable here.
* It requires that all organizations must use the same method for counting LOC. This is so because some organizations use only executable statements, some useful comments, and some do not.
* Thus, the standard needs to be established.
* These metrics are not universally accepted.

# **Advantages of LOC :**

* Simple to measure

# **Disadvantage of LOC :**

* It is defined on the code. For example, it cannot measure the size of the specification.
* It characterizes only one specific view of size, namely length, it takes no account of functionality or complexity.
* Bad software design may cause an excessive line of code.
* It is language dependent.
* Users cannot easily understand it.

# **LOC based Estimation :**

* Size oriented measure is derived by considering the size of software that has been produced.
* The organization builds a simple record of size measure for the software projects
* It is built on past experiences of organizations.
* It is a direct measure of software.

| Project | LOC | Effort | Cost ($) | Doc. (pgs) | Errors | Defects | People |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ABC | 10,000 | 20 | 170 | 400 | 100 | 12 | 4 |
| PQR | 20,000 | 60 | 300 | 1000 | 129 | 32 | 6 |

| XYZ | 35,000 | 65 | 522 | 1290 | 280 | 87 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- |

A simple set of size measure that can be developed is as given below:

* Size Kilo Lines of Code (KLOC)
* Effort Person/month
* Productivity = KLOC/person-month
* Quality Number of faults/KLOC
* Cost-S/KLOC
* Documentation-Pages of documentation/KLOC

The size measure is based on the lines of code computation. The lines of code are defined as one line of text in a source file.

While counting the lines of code the simplest standard is

* Don't count blank lines
* Don't count comments.
* Count everything else.

The size oriented measure is not a universally accepted method.

# **Advantages :**

1. Artifact of software development which is easily counted.
2. Many existing methods use LOC as a key input.
3. A large body of literature and data based on LOC already exists.

# **Disadvantages :**

1. This measure is dependent upon the programming language
2. This method is well designed but shorter programs may suffer.
3. It does not accommodate non procedural languages.
4. In the early stage of development it is difficult to estimate LOC.

# **Example of LOC based Estimation :**

Consider an ABC project with some important modules such as

1. User interface and control facilities
2. 2D graphics analysis
3. 3D graphics analysis
4. Database management
5. Computer graphics display facility
6. Peripheral control function
7. Design analysis models

# **Solution :**

For estimating the given application we consider each module as separate function and corresponding lines of code can be estimated in the following table as :

| **Function** | **Estimated LOC** |
| --- | --- |
| User interface and control facilities (UICF) | 2500 |
| Two-dimensional geometric analysis (2DGA) | 5600 |
| Two-dimensional geometric analysis (2DGA) | 6450 |
| Database management (DBM) | 3100 |
| Computer graphics display facilities (CGDF) | 4740 |
| Peripheral control function (PCF) | 2250 |
| Design analysis modules (DAM) | 7980 |
| **Estimated lines of code** | **32,620** |

Expected LOC for 3D Geometric analysis function based on three point estimation is :

| * Optimistic estimation | 4700 |
| --- | --- |
| * Most likely estimation | 6000 |
| * Pessimistic estimation | 10,000 |

S = [Sopt + (4 \* Sm) + Spess] / 6

Expected value = [4700 + (4 \* 6000) + 10,000] / 6 → 6450

A review of historical data indicates :

1. Average productivity is 500 LOC per month.
2. Average labor cost is $6000 per month.

Then cost for lines of code can be estimated as

# **cost/LOC = (6000/500)= $12**

By considering total estimated LOC as 32620

* + Total estimated project cost = (32620\*12) = $391440.
  + Total estimated project effort = (32620/500)= 65 Person-months.

**EXP -7B**

**Aim :** To estimate the cost of a project using software metrics (FP analysis).

# **Theory :**

Allan J. Albrecht initially developed function Point Analysis in 1979 at IBM and it has been further modified by the International Function Point Users Group (IFPUG). FPA is used to make an estimate of the software project, including its testing in terms of functionality or function size of the software product. The functional size of the product is measured in terms of the function point, which

is a standard of measurement to measure the software application. The basic and primary purpose of the functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request. Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

Following are the points regarding FPs:

FPs of an application is found out by counting the number and types of functions used in the applications. Various functions used in an application can be put under five types, as shown:

| **Measurements Parameters** | **Examples** |
| --- | --- |
| Number of External Inputs (EI) | Input screen and tables |
| Number of External Output (EO) | Output screens and reports |
| Number of external inquiries (EQ) | Prompts and interrupts. |
| Number of internal files (ILF) | Databases and directories |
| Number of external interfaces (EIF) | Shared databases and shared routines. |

**Function Point Analysis :**

| **No.** | **Count** |
| --- | --- |
| No. of User Input | 50 |
| No. of User Output | 40 |
| No. of User Enquiries | 35 |
| No. of User Fills | 06 |
| No. of External Interface | 04 |

To calculate Function Point Value = **FP(value)**

| **Function Type** | **Simple** | **Average** | **Complex** |
| --- | --- | --- | --- |
| External Input | 3 | 4 | 6 |
| External Output | 4 | 5 | 7 |
| External Inquiry | 3 | 4 | 6 |
| External Logical Files | 7 | 10 | 15 |
| External Internal Files | 5 | 7 | 10 |

**Unadjusted FP :**

| **Domain** | **Count \* Average** | **FP Count** |
| --- | --- | --- |
| User Input | 50 \* 4 | 200 |
| User Output | 40 \* 5 | 200 |
| User Enquiry | 35 \* 4 | 140 |
| Internal Logical Files (ILF) | 06 \* 10 | 60 |
| External Internal Files (EIF) | 04 \* 7 | 28 |

**Complexity Adjustment Factor :** 3 and -14 Questions

∴ **Σ (f1)** = 14 \* 3 = 42

**FP(value)** = Total Count \* [0.65 + 0.01 + Σ (f1)]

= 628 \* [0.65 + (0.01 \* 42)]

= 628 \* [0.65 + 0.42]

= 628 \* 1.07

= 671.96

**FP(value)** = 671.96

**Productivity FP/Effort** = 408/36.9 = 11.1

**Total pages of documentation** = Technical document + User document = 265 + 122 = 387 pages

**Documentation** = Pages of documentation/FP = 387/408 = 0.94

**Cost per function** = Cost/Productivity = 7744/11.1 = $ 700