Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT IV

Chapter 3 – Software Measurement

Concepts:

Introduction

Size Oriented Metrics

Function Oriented Metrics

Function Point Metrics

Introduction

Software is measured:

1. To establish the quality of the current product or process

2. To predict future qualities of the product or process

3. To improve the quality of a product or process

4. To determine the state of the project in relation to budget and schedule

Measurement and Metrics

A measurement is a manifestation of the size, quantity, amount, or dimension of a particular attribute of a product or process. For example, the number of errors in a system is a measurement. With regards to software, we can measure Product, Process and People known as 3Ps of software measurement.

Measurements in the physical world can be categorized in two ways: direct measures (e.g., the length of a bolt) **and indirect measures** (e.g., the "quality" of bolts produced, measured by counting rejects). Software metrics can be categorized similarly.

Direct measures of the software engineering process include cost and effort applied. Direct measures of the product include lines of code (LOC) produced, execution speed, memory size, and defects reported over some set period of time. **Indirect measures** of the product include functionality, quality, complexity, efficiency, reliability, maintainability, and many other "abilities".

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.

Software metrics are classified as product metrics and process metrics.

1. Product Metrics: These are the measures of various characteristics of the software product. The two important software characteristics are:

1. Size and complexity of software.
2. Quality and reliability of software.

These metrics can be computed for different stages of SDLC.

2. Process Metrics: These are the measures of various characteristics of the software development process. For example, the efficiency of fault detection. They are used to measure the characteristics of methods, techniques, and tools that are used for developing software.

Size oriented Metrics

Size-oriented software metrics are derived by normalizing quality and/or productivity measures by considering the size of the software that has been produced. Size oriented metrics built on the past experiences of organizations. It uses direct measure of the software measurement. **The size-oriented metrics can be developed for each project by using Line of source code per programmer-month (LOC/pm)** is widely used software productivity metrics.

We can compute LOC/pm by counting the total number of lines of source code that are delivered then divide the count by total time in programmer-months required to complete the project. LOC metric applies only to the programming process. Therefore here the time includes the time required for requirement , designing, coding, testing and documentation.

Size oriented software metrics are derived by normalizing quality and productivity measures by considering the size of the software that has been produced:

1. Size of a product = Kilo Lines of Code (KLOC)

2. Productivity = KLOC/person – month

3. Quality = number of faults/KLOC

4. Cost = $/KLOC

5. Documentation = Pages of Documentation/KLOC

Advantages of LOC

1. Simple to measure

Disadvantage of LOC

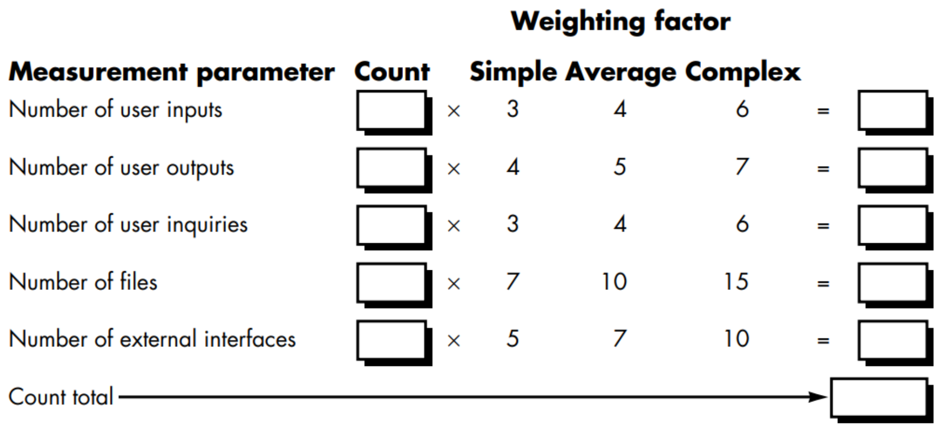
1. It is defined on the code. For example, it cannot measure the size of the specification.
2. It characterizes only one specific view of size, namely length, it takes no account of functionality or complexity
3. Bad software design may cause an excessive line of code
4. It is language dependent
5. Users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

Function oriented metrics

Function-oriented software metrics use a measure of the functionality delivered by the application as a normalization value. Since 'functionality' cannot be measured directly, it must be derived indirectly using other direct measures. Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity.

Function points are computed [IFP94] by completing the table shown in Figure 4.5. Five information domain characteristics are determined and counts are provided in the appropriate table location. Information domain values are defined in the following manner:

FIGURE 4.5
Computing function points



			Weighting factor				
Measurement parameter	**Count**		**Simple**	**Average**	**Complex**		
Number of user inputs		×	3	4	6	=	
Number of user outputs		×	4	5	7	=	
Number of user inquiries		×	3	4	6	=	
Number of files		×	7	10	15	=	
Number of external interfaces		×	5	7	10	=	
Count total							

1. Number of user inputs - Each user input that provides distinct application- oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.

2. Number of user outputs - Each user output that provides application- oriented information to the user is counted. In this context output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.

3. Number of user inquiries - An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.

4. Number of files - Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted.

5. Number of external interfaces - All machine-readable interfaces (e.g., data files on storage media) that are used to transmit information to another system are counted.

Once these data have been collected, a complexity value is associated with each count. Organizations that use function point methods develop criteria for determining whether a particular entry is simple, average, or complex. Nonetheless, the determination of complexity is somewhat subjective.

To compute function points (FP), the following relationship is used:

$$\textbf{FP} = \textbf{count total } [0.65 + 0.01 \sum(F_i)]$$

where count total is the sum of all FP entries obtained from Figure 4.5.

Once function points have been calculated, they are used in a manner analogous to LOC as a way to normalize measures for software productivity, quality, and other attributes:

- Errors per FP.
- Defects per FP.
- $ per FP.
- Pages of documentation per FP.
- FP per person-month

Extended Function Point metrics

The function point measure was originally designed to be applied to business information systems applications. **A function point extension called feature points [JON91], is a superset of the function point measure that can be applied to systems and engineering software applications.**

FP metric has been further extended to compute:

1. Feature points.
2. 3D function points.

Feature Points

1. Feature point is the superset of function point measure that can be applied to systems and engineering software applications.

2. The feature points are used in those applications in which the algorithmic complexity is high like real-time systems where time constraints are there, embedded systems, etc.

3. Feature points are computed by counting the information domain values and are weighed by only single weight.

4. Feature point includes another measurement parameter-ALGORITHM.

The table for the computation of feature point is as follows:

Feature Point Calculations

Measurement Parameter	Count		Weighing factor	
1. Number of external inputs (EI)	-	*	4	-
2. Number of external outputs (EO)	-	*	5	-
3. Number of external inquiries (EQ)	-	*	4	-
4. Number of internal files (ILF)	-	*	7	-
5. Number of external interfaces (EIF)	-	*	7	-
6.Algorithms used Count total →	-	*	3	-

The feature point is thus calculated with the following formula:

$$FP = \text{Count-total} * [0.65 + 0.01 * \sum(f_i)]$$
$$= \text{Count-total} * CAF$$

where count-total is obtained from the above table.

$$CAF = [0.65 + 0.01 * \sum(\mathbf{f_i})]$$

and $\sum(f_i)$ is the sum of all 14 questionnaires and show the complexity adjustment value/factor-CAF (where i ranges from 1 to 14). Usually, a student is provided with the value of $\sum(f_i)$.

6. Function point and feature point both represent systems functionality only.

7. For real-time applications that are very complex, the feature point is between 20 and 35% higher than the count determined using function point above.

3D function points

Three dimensions may be used to represent 3D function points data dimension, functional dimension, and control dimension.

2. The data dimension is evaluated as FPs are calculated. Herein, counts are made for inputs, outputs, inquiries, external interfaces, and files.

3. The functional dimension adds another feature-Transformation, that is, the sequence of steps which transforms input to output.

4. The control dimension that adds another feature-Transition that is defined as the total number of transitions between states. A state represents some externally observable mode.

A **state** represents some externally observable mode of behavior, and a transition occurs as a result of some event that causes the software or system to change its mode of behavior (i.e., to change state). For example, a wireless phone contains software that supports auto dial functions. To enter the auto-dial state from a resting state, the user presses an **Auto** key on the keypad. This event causes an LCD display to prompt for a code that will indicate the party to be called. Upon entry of the code and hitting the **Dial** key (another event), the wireless phone software makes a transition to the dialing state. When computing 3D function points, transitions are not assigned a complexity value.