- Out of these three components if one component is altered then it will impact other two factors. For example : If an organization makes use of new testing tool (*technology*) then the staff (*people*) must be trained to work with this tool. The organization must consider if this new tool helps in generating the desired **software product.**
- This process triangle resides within the circle. This **circle** specifies the **environmental conditions** such as :
  - Customer characteristics (communication and collaboration between user and developer).
  - Business conditions (organizational policies, business rules)
  - Development environment (use of new technologies, use of automated tools).

Grady suggested some guideline for collecting software metrics. These are known as **Software Metrics Etiquette.** These are as given below -

1. Never use metrics to please the individuals or to threaten the individuals or team.
2. Use common sense and organizational sensitivity while understanding the metrics data.
3. Collectively together with project manager and software team set the goals and use of metrics for the software.
4. The metrics data directs the area of process improvement.
5. Do not focus on single metrics so that other important metrics may get omitted.

Review Question

1. *Write a note on process and project metrics.*

8.2 Software Measurement

MU - May-09,10, Dec.-09,11, Marks 10

Software measurement is an ability to measure attributes of software and software development process so that the software engineering activities can be improved.

The measurement of software can be classified into two categories -

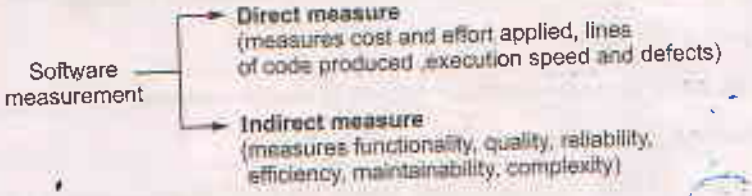


Fig. 8.2.1 Software measurement

Let us discuss the metrics based on these approaches -

8.2.1 Size Oriented Metrics

- 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
:	:	:	:	:	:	:	:

Table 8.2.1 Size measure

- 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 = $/KLOC
  - Documentation = Pages of documentation/KLOC
- The size measure is based on the lines of code computation. The lines of code is 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 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 program may get suffered.
3. It does not accommodate non procedural languages.
4. In early stage of development it is difficult to estimate LOC.

8.2.2 Function Oriented Metrics

- The oriented model is based on functionality of the delivered application.
- These are generally independent of the programming language used.
- This method is developed by Albrecht in 1979 for IBM. It uses function points.
- Function points are derived using :
  1. Countable measures of the software requirements domain.
  2. Assessments of the software complexity.

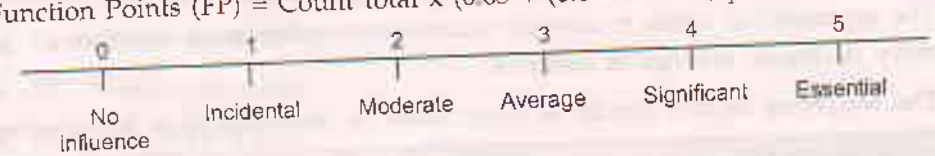
How to calculate function point ?

- The data for following information domain characteristics are collected :
  1. Number of user inputs - Each user input which provides distinct application data to the software is counted.
  2. Number of user outputs - Each user output that provides application data to the user is counted, e.g. screens, reports, error messages.
  3. Number of user inquiries - An on-line input that results in the generation of some immediate software response in the form of an output.
  4. Number of files - Each logical master file, i.e. a logical grouping of data that may be part of a database or a separate file.
  5. Number of external interfaces - All machine-readable interfaces that are used to transmit information to another system are counted.
- The organization needs to develop criteria which determine whether a particular entry is simple, average or complex.
- The weighting factors should be determined by observations or by experiments.

Domain characteristics	Count		Weighting factor			Count
			Simple	Average	Complex	
Number of user input		X	3	4	6	
Number of user output		X	4	5	7	
Number of user inquiries		X	3	4	6	

Number of files		X	7	10	15
Number of external interfaces		X	5	7	10
Count total					

- The count table can be computed with the help of above given table.
- Now the software complexity can be computed by answering following questions. These are **complexity adjustment values.**
  1. Does the system need reliable backup and recovery ?
  2. Are data communications required ?
  3. Are there distributed processing functions ?
  4. Is performance of the system 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 which is designed being 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 ?
- Rate each of the above factors according to the following scale :
- Function Points (FP) = Count total x (0.65 + (0.01 x Sum (F_i))



- Once the functional point is calculated then we can compute various measures as follows
  - Productivity = FP/person-month
  - Quality = Number of faults/FP
  - Cost = $/FP
  - Documentation = Pages of documentation/FP.

Advantages

1. This method is independent of programming languages.
2. It is based on the data which can be obtained in early stage of project .

Disadvantages

1) This method is more suitable for business systems and can be developed for that domain.
2) Many aspects of this method are not validated.
3) The functional point has no significant meaning. It is just a numerical value.

Comparison between size oriented and function oriented metrics

Sr. No.	Size oriented metrics	Function oriented metrics
1.	Size oriented software metrics is by considering the size of the software that has been produced.	Function oriented metrics use a measure of functionality delivered by the software.
2.	For a size oriented metric the software organization maintains simple records in tabular form. The typical table entries are : Project name, LOC, Effort, Pages of documents, errors, defects, total number of people working on project.	Most widely used function oriented metric is the Function Point (FP) computation of the function point is based on characteristics of software's information domain and complexity.

Exercise 8.2.1 *Study of requirement specification for ABC project has produced following results : Need for 7 inputs, 10 outputs, 6 inquiries, 17 files and 4 external interfaces. Input and external interface function point attributes are of average complexity and all other function points attributes are of low complexity.*

Determine adjusted function points assuming complexity adjustment value is 32.

Solution : Given that :

7 inputs

10 Outputs

6 inquiries

17 files

4 external interfaces

Average complexity for inputs and external interfaces. Low complexity for remaining parameters.

Adjusted function point value $\sum (F_i) = 32$

Let us calculate count total value.

Measurement parameters	Count	Weighting factor				
		X	Simple	Average	complex	
Number of user inputs	7	X		4		28
Number of user outputs	10	X	4			40
Number of user inquiries	6	X	3			18
Number of files	17	X	7			119
Number of external interfaces	4	X		7		28
Count total						233

$$\text{Function point} = \text{Count total} \times [0.65 + 0.01 \times \Sigma(F_i)]$$

$$= 233 \times [0.65 + 0.01 \times 32]$$

$$= 233 \times [0.65 + 0.32]$$

$$= 233 \times 0.97$$

$$FP = 226.01$$

Hence adjusted function point is **226.01.**

8.2.3 Attributes of Effective Software Metrics

The effective software metrics should have following attributes.

1. **Simple and computable** - The derivation of metric should be easy to compute and should not be a time consuming activity.
2. **Empirically and intuitively persuasive** - It should be immediate and can be derived based on observations.
3. **Consistent and objective** - The metric should produce unambiguous results. Anybody should get the same result by using these metrics when same set of information is used.
4. **Consistent in its use of units and dimensions** - The mathematical units and dimensions used for the metric should be consistent. And there should not be intermixing of units.
5. **Programming language independent** - The metric should be based on analysis model, design model and program structure. It should be independent of programing languages, syntax or semantic of any programing language.

4. Change sizing

This approach is used when existing software has to be modified as per the requirement of the project. The size of the software is then estimated by the number and type of reuse, addition of code, change made in the code, deletion of code.

- The result of each sizing approaches must be combined statistically to create **three-point estimate** which is also known as **expected-value estimate.**

8.4.2 Problem based Estimation

- The problem based estimation is conducted using LOC based estimation, FP based estimation, process based estimation and use cased based estimation.
- LOC and FP based data are used in two ways during software estimation -
  1. These are useful to estimate the **size** each element of software.
  2. The baseline metrics are collected from past project and LOC and FP data is used in conjunction with estimation variable to develop cost and effort values for the project. (LOC) and (FP) estimation are different estimation techniques. Yet, both have number of characteristics in common.
- With a bounded statement of software scope a project planning process begins and by using the statement of scope the software problem is decomposed into the functions that can be estimated individually.
- (LOC) or (FP)is then estimated for each function.
- **Baseline productively metrics** are then applied to the appropriate estimation variable and cost or effort for the function is derived.
- **Function estimates** are combined to obtain an overall estimate for the entire project.
- Using historical data the project planner **expected value** by considering following variables -
  1. Optimistic 2. Most likely 3. Pessimistic

For example, following formula

$$S = [S_{opt} + 4 * S_m + S_{pess}]/6$$

considers for "most likely" estimate where S is the estimation size variable, S_{opt} represents the optimistic estimate, S_m represents the most likely estimate and S_{pess} represents the pessimistic estimate values.

8.4.3 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

Estimate the project in based on LOC

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
2D Graphics Analysis(2DGA)	5600
3D Geometric Analysis function(3DGA)	6450
Database Management(DBM)	3100
Computer Graphics Display Facility(CGDF)	4740
Peripheral Control Function(PCF)	2250
Design Analysis Modules (DAM)	7980
Total Estimation in LOC	**32620**

- Expected LOC for 3D Geometric analysis function based on three point estimation is -
  - Optimistic estimation 4700
  - Most likely estimation 6000
  - Pessimistic estimation 10000

$$S = [S_{opt} + (4 * S_m) + S_{pess}]/6$$

Expected value = [4700 + (4 * 6000) + 10000] /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

8.4.4 Example of FP based Estimation

FP focuses on information domain values rather than software functions. Thus we create a function point calculation table for ABC project, discussed in section 8.4.3.

INFO DOMAIN VALUE	Opt.	most likely	pessimistic	esti. value	weight factor	FP
NO.OF INPUTS	25	28	32	28.1	4	112
NO. OF OUTPUTS	14	17	20	17	5	85
NO. OF INQUIRIES	17	23	30	23.1	5	116
NO. OF FILES	5	5	7	5.33	10	53
NO.OF EXTERNAL INTERFACES	2	2	3	2	7	15
COUNT TOTAL						381

- For this example we assume average complexity weighting factor.
- Each of the complexity weighting factor is estimated and the complexity adjustment factor is computed using the complexity factor table below. (Based on the 14 questions)

Sr. No.	FACTOR	VALUE (Fi).
1.	Back-up and recovery ?	4
2.	Data communication ?	2
3.	Distributed processing ?	0
4.	Performance critical ?	4
5.	Existing operational environment ?	3
6.	On-line data entry ?	4
7.	Input transactions over multiple screens ?	5
8.	Online updates ?	3
9.	Information domain values complex ?	5
10.	Internal processing complex ?	5
11.	Code designed for reuse ?	4
12.	Conversion / installation in design ?	3
13.	Multiple installations ?	5
14.	Application designed for change ?	5
		$\sum$ (Fi) → 52

The estimated number of adjusted FP is derived using the following formula : -

FP ESTIMATED = (FP COUNT TOTAL * [COMPLEXITY ADJUSTMENT FACTOR]

$$\text{FP ESTIMATED} = \text{COUNT TOTAL} * [0.65 + (0.01 * \sum (Fi))]$$

Complexity adjustment factor = [0.65 + (0.01 * 52)] = 1.17

- **FP ESTIMATED = (381 * 1.17) = 446** (Function point count adjusted with complexity adjustment factor)

- A review of historical data indicates -
  1. Average productivity is 6.5 FP/Person month
  2. Average labor cost is $6000 per month

Calculations for cost per function point, total estimated project cost and total effort

1. The cost per function point = (6000 / 6.5) = $923
2. Total estimated project cost = (446 * 923) = $411658
3. Total estimated effort = (446 / 6.5) = 69 Person-month.

8.4.5 Process based Estimation

- This is the most commonly used estimation technique for estimating the project based on the processes used.
- In this technique, the process is decomposed into relatively size set of tasks and the effort required to accomplish each task is estimated.
- The estimation begins with identification of software functions obtained from the project scope. Then a series of software process activities must be performed for each function. The function and software process activities are presented in a tabular form. Then planner estimates the efforts for each software function.

8.4.6 Example of Process based Estimation

Consider the same project described in section 8.4.3. We can estimate it using process based estimation technique by creating a table as shown below. In this technique for each of the business function software engineering tasks such as analysis, design, coding and testing is carried out. The estimated efforts for each of these functions are identified and their sum is obtained. In the following table, the sum of all the rows and columns is taken and the effort for each business function is estimated in percentage.