JAMNALAL BAJAJ INSTITUTE
OF MANAGEMENT STUDIES

SOFTWARE QUALITY METRICS

JBIMS MIM SEM V – 2015-2018

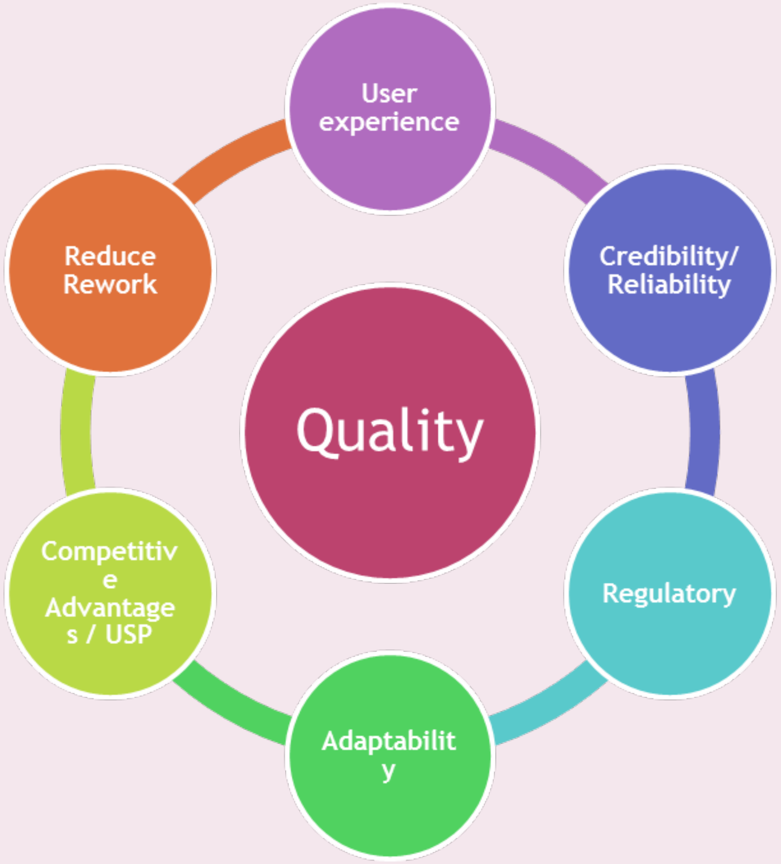
15-I-131 – MUFADDAL NULLWALA

CONTENTS

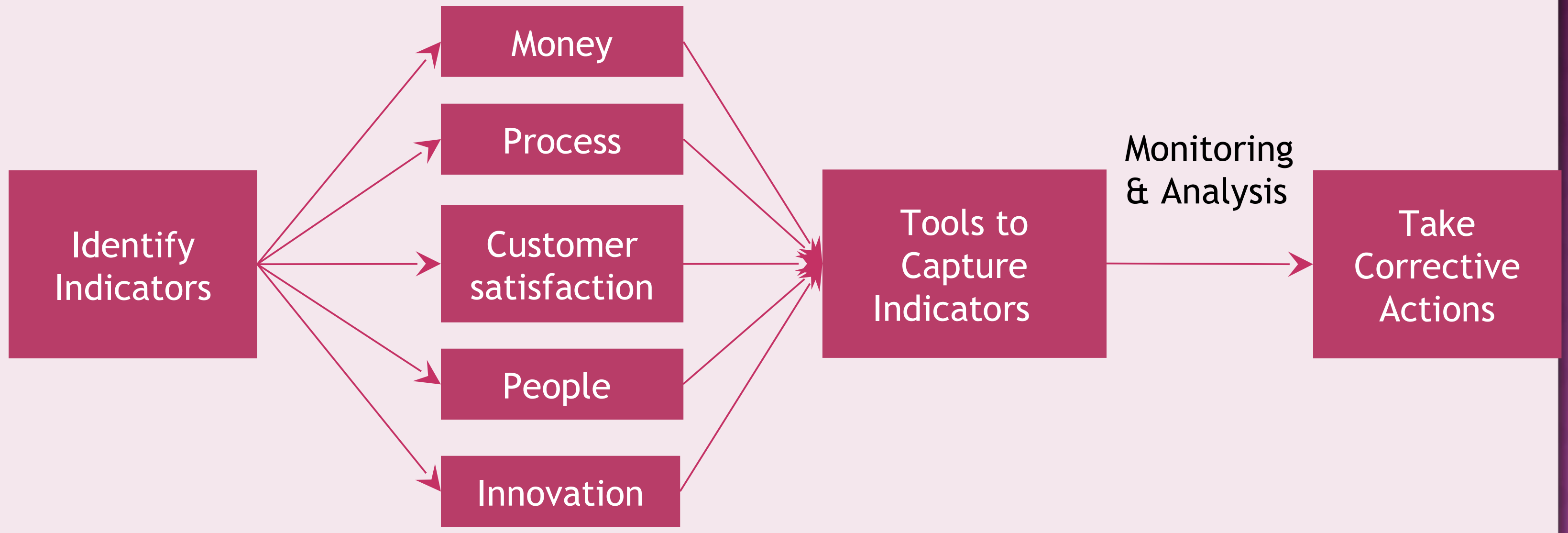
WHY DO WE NEED QUALITY MANAGEMENT ?

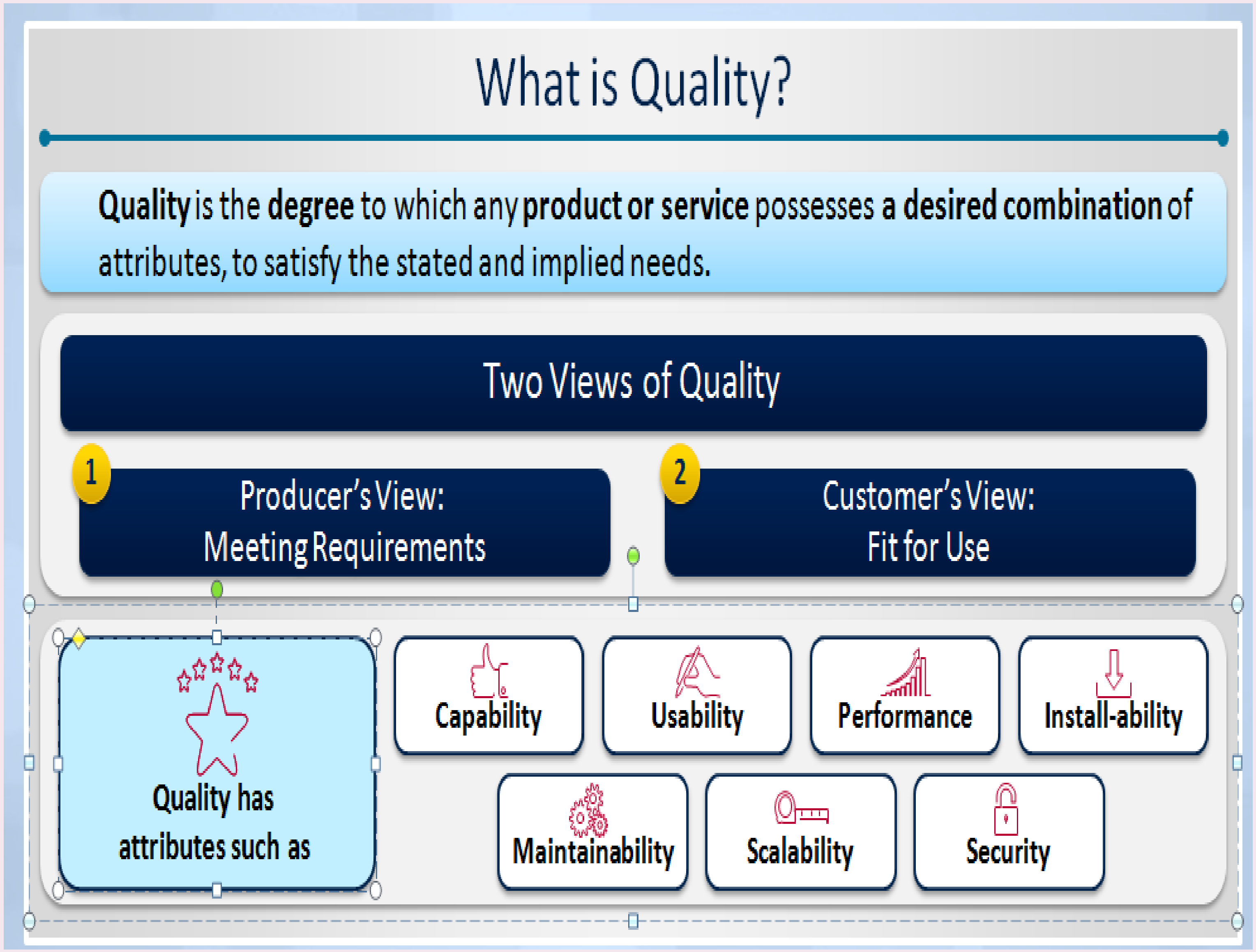
- Who Are we ?
- What we do ?
- Why makes us do that?

WHY QUALITY ?


Quality
User experience
Credibility/ Reliability
Regulatory
Adaptabilit y
Competitiv e Advantage s / USP
Reduce Rework

PROCESS OF QUALITY MANAGEMENT





What is Quality?

Quality is the **degree** to which any **product or service** possesses **a desired combination** of attributes, to satisfy the stated and implied needs.

Two Views of Quality

1. Producer's View: Meeting Requirements
2. Customer's View: Fit for Use

Quality has attributes such as

- Capability
- Usability
- Performance
- Install-ability
- Maintainability
- Scalability
- Security

SOFTWARE QUALITY METRICS

- The subset of metrics that focus on quality
- Software quality metrics can be divided into:
  - End-product quality metrics
  - In-process quality metrics
- The essence of software quality engineering is to investigate the relationships among in-process metric, project characteristics , and end-product quality, and, based on the findings, engineer improvements in quality to both the process and the product.

GENERAL USE OF SOFTWARE METRICS

- Software metrics are used to obtain objective reproducible measurements that can be useful for quality assurance, performance, debugging, management, and estimating costs.

- Finding defects in code (post release and prior to release), predicting defective code, predicting project success, and predicting project risk .

TYPES OF SOFTWARE METRICS

- Product metrics – e.g., size, complexity, design features, performance, quality level
- Process metrics – e.g., effectiveness of defect removal, response time of the fix process
- Project metrics – e.g., number of software developers, cost, schedule, productivity

THREE GROUPS OF SOFTWARE QUALITY METRICS

- Product quality
- In-process quality
- Maintenance quality
- Product Quality Matrices

PRODUCT QUALITY METRICS

- Intrinsic product quality
  - Mean time to failure
  - Defect density
- Customer related
  - Customer problems
  - Customer satisfaction

INTRINSIC PRODUCT QUALITY

- Intrinsic product quality is usually measured by:
  - the number of “bugs” (functional defects) in the software (defect density), or
  - how long the software can run before “crashing” (MTTF – mean time to failure)

- The two metrics are correlated but different

DIFFERENCE BETWEEN ERRORS, DEFECTS, FAULTS, AND FAILURES (IEEE/ANSI)

- An error is a human mistake that results in incorrect software.
- The resulting fault is an accidental condition that causes a unit of the system to fail to function as required.
- A defect is an anomaly in a product.
- A failure occurs when a functional unit of a software-related system can no longer perform its required function or cannot perform it within specified limits

THE DEFECT DENSITY METRIC

- This metric is the number of defects over the opportunities for error (OPE) during some specified time frame.
- We can use the number of unique causes of observed failures (failures are just defects materialized) to approximate the number of defects.
- The size of the software in either lines of code or function points is used to approximate OPE.

LINES OF CODE

- Lines of Code or LOC is a quantitative measurement in computer programming for files that contains code from a computer programming language, in text form.
- The number of lines indicates the size of a given file and gives some indication of the work involved.
- It is used as the Unit of Sizing of the Software

METRICS TO COUNT LOC

Metric	Supported as	Description
Physical lines	LINES	This metric counts the physical lines, but excludes classic VB form definitions and attributes.
Physical lines of code	*(not supported)*	This type of a metric counts the lines but excludes empty lines and comments. This is sometimes referred to as the *source lines of code* (sLOC) metric.
Logical lines	LLINES	A *logical line* covers one or more physical lines. Two or more physical lines can be joined as one logical line with the line continuation sequence " _". The LLINES metric counts a joined line just once regardless of how many physical lines there are in it.
Logical lines of code	LLOC	A logical line of code is one that contains actual source code. An empty line or a comment line is not counted in LLOC.
Statements	STMT	This is not a line count, but a statement count. Visual Basic programs typically contain one statement per line of code. However, it's possible to put several statements on one line by using the colon ":" or writing single-line If..Then statements.

LINES - PHYSICAL LINES

LINES = Number of lines

This is the simplest line count, LINES counts every line, be it a code, a comment or an empty line.

Maximum procedure length

Max 66 linesLINES <= 66. The procedure fits on one page when printed.

Max 150 linesLINES <= 150. A recommendation for Java.

Max 200 linesLINES <= 200. The procedure fits on 3 pages.

Maximum file length

Max 1000 linesLINES <= 1000. This file size accommodates 15 one-page procedures or 100 short 10-line procedures.

Max 2000 linesLINES <= 2000. A recommendation for Java.

LLOC - LOGICAL LINES OF CODE

LLOC = Number of logical lines of code

LLOC counts all logical lines except the following:

- Full comment lines
- Whitespace lines
- Lines excluded by compiler conditional directives

Maximum acceptable LLOC

Procedure LLOC <= 50
Class LLOC <= 1500
File LLOC <= 2000

Minimum acceptable LLOC

Procedure LLOC >= 3
Class LLOC >= 3
File LLOC >= 1

FUNCTION POINTS

Function points are a unit measure for software much like an hour is to measuring time, miles are to measuring distance or Celsius is to measuring temperature.

Function Point Analysis, systems are divided into five large classes and general system characteristics:

- External Inputs
- External Outputs
- External Inquires

Transactions

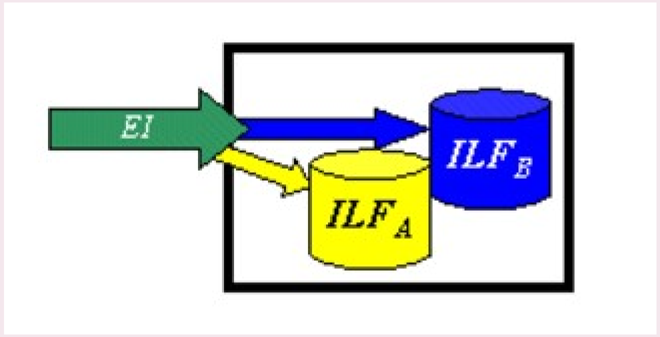
- Logical Files
- External Interface Files

Logical Information

COMPONENTS OF FUNCTION POINTS

- ***External Inputs (EI):***

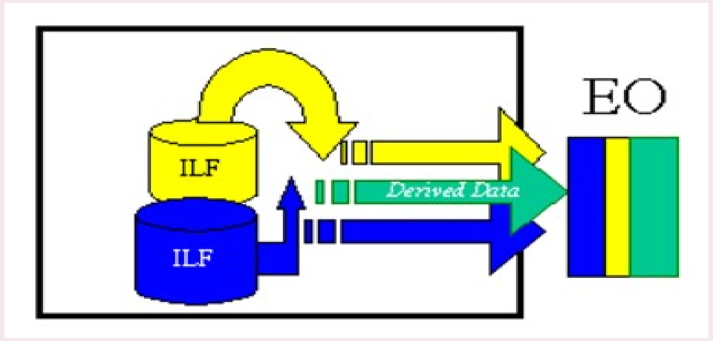
  It is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files. The data can be either control information or business information. The graphic represents a simple EI that updates 2 ILF's (FTR's).



COMPONENTS OF FUNCTION POINTS …CONTD’

- ***External Outputs (EO):***

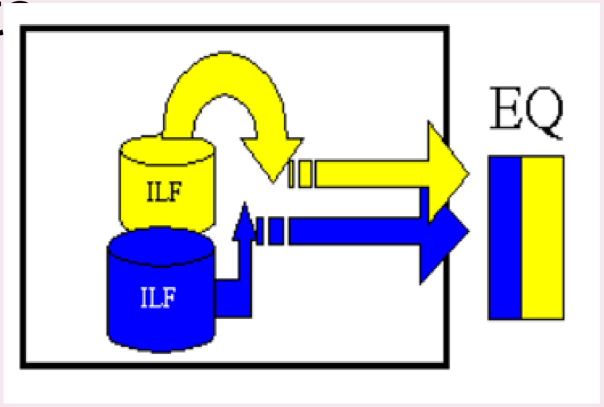
  Is an elementary process in which derived data passes across the boundary from inside to outside. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications. These reports and files are created from one or more internal logical files and external interface file. The following graphic represents on EO with 2 FTR's there is derived information (green) that has been derived from the ILF's



COMPONENTS OF FUNCTION POINTS …CONTD’

- ***External Inquiry (EQ)***

  An elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. The input process does not update any Internal Logical Files, and the output side does not contain derived data. The graphic below represents an EQ with two ILF's and no derived dat

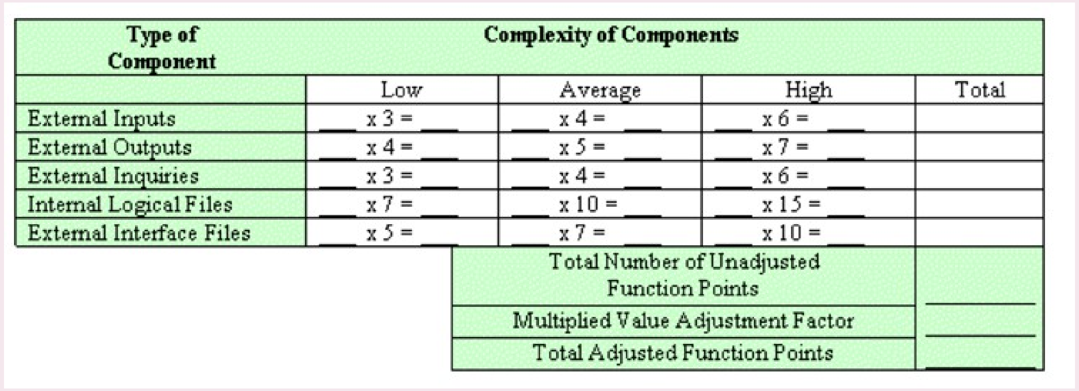


COMPONENTS OF FUNCTION POINTS …CONTD’

- ***Internal Logical Files (ILF’s)*** - a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs.
- ***External Interface Files (EIF’s)*** - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The external interface file is an internal logical file for another application.

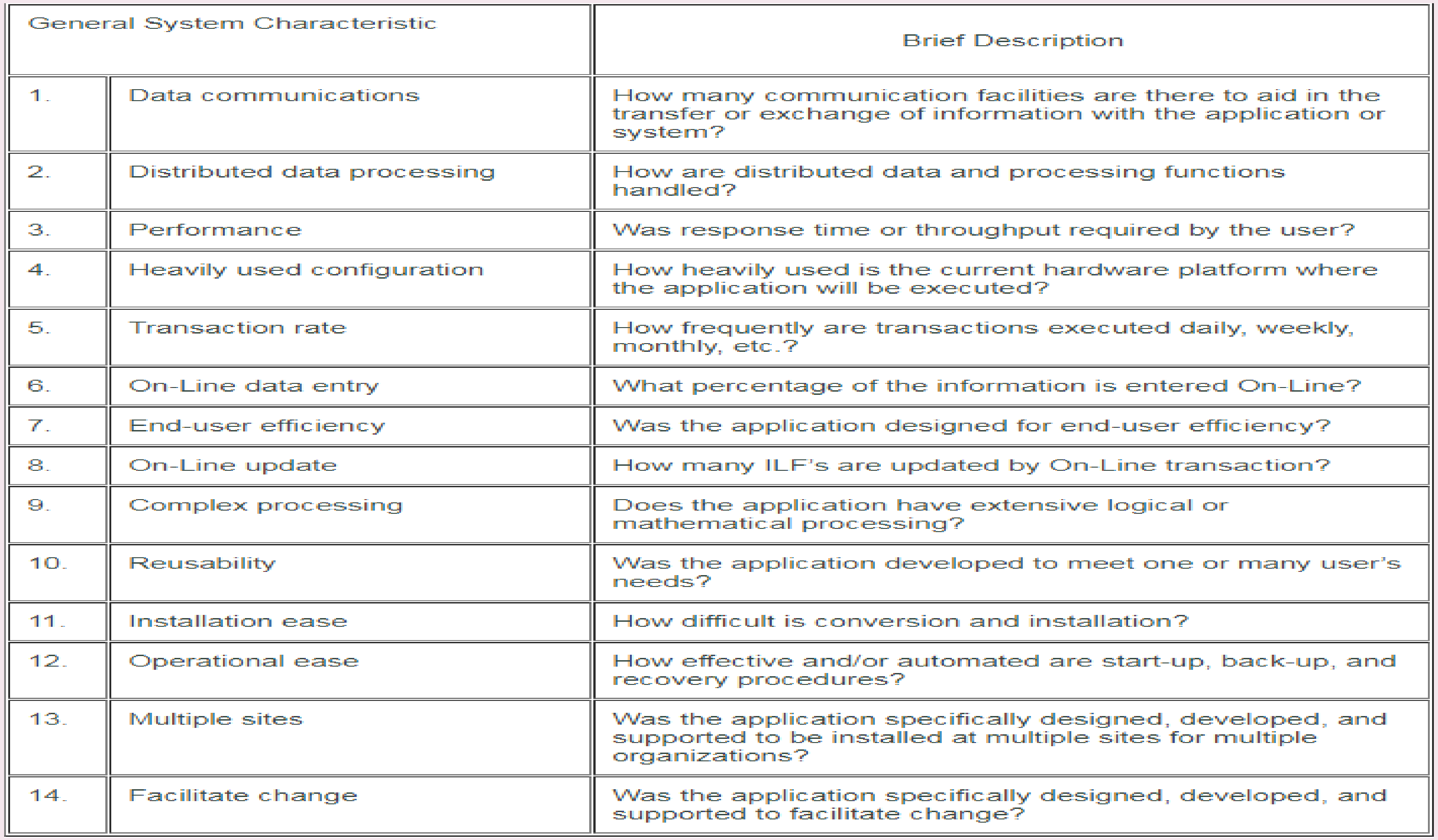
FUNCTION POINTS….CONTD’

- After the components have been classified as one of the five major components (EI’s, EO’s, EQ’s, ILF’s or EIF’s), a ranking of low, average or high is assigned.
- The counts for each level of complexity for each type of component can be entered into a table such as the following one.



Type of Component	Complexity of Components			
	Low	Average	High	Total
External Inputs	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	
External Outputs	___ x 4 = ___	___ x 5 = ___	___ x 7 = ___	
External Inquiries	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	
Internal Logical Files	___ x 7 = ___	___ x 10 = ___	___ x 15 = ___	
External Interface Files	___ x 5 = ___	___ x 7 = ___	___ x 10 = ___	
		Total Number of Unadjusted Function Points		___
		Multiplied Value Adjustment Factor		___
		Total Adjusted Function Points		___

- The value adjustment factor (VAF) is based on 14 general system characteristics (GSC's) that rate the general functionality of the application being counted. Each characteristic has associated descriptions that help determine the degrees of influence of the characteristics.



General System Characteristic		Brief Description
1.	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2.	Distributed data processing	How are distributed data and processing functions handled?
3.	Performance	Was response time or throughput required by the user?
4.	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5.	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6.	On-Line data entry	What percentage of the information is entered On-Line?
7.	End-user efficiency	Was the application designed for end-user efficiency?
8.	On-Line update	How many ILF's are updated by On-Line transaction?
9.	Complex processing	Does the application have extensive logical or mathematical processing?
10.	Reusability	Was the application developed to meet one or many user's needs?
11.	Installation ease	How difficult is conversion and installation?
12.	Operational ease	How effective and/or automated are start-up, back-up, and recovery procedures?
13.	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14.	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

- Once all the 14 GSC's have been answered, they should be tabulated using the IFPUG Value Adjustment Equation (VAF)

  VAF = 0.65 + [(Ci) / 100]

  Ci = degree of influence for each General System Characteristic

0.65 may vary as per requirements

- The final Function Point Count is obtained by multiplying the VAF times the Unadjusted Function Point (UAF).

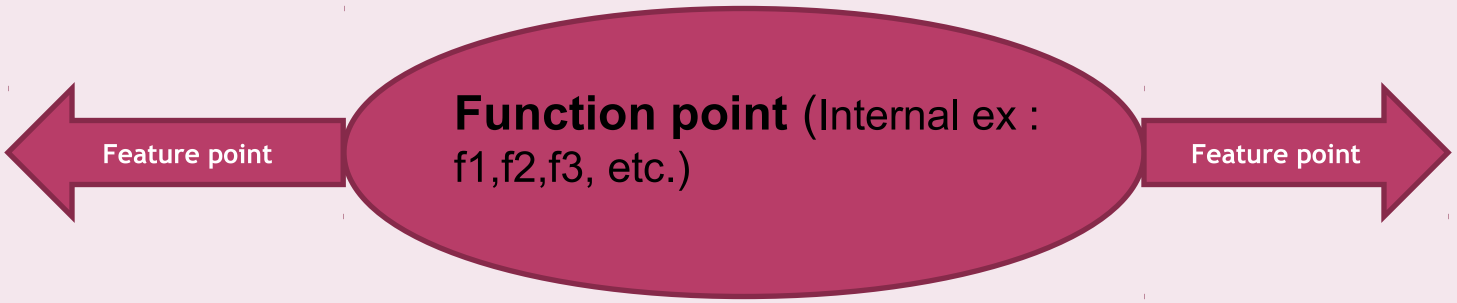
  FP = UAF * VAF

FEATURE POINT METRICS

Feature point metrics is software estimation technique where identification of different features of the software and estimate cost based on features.

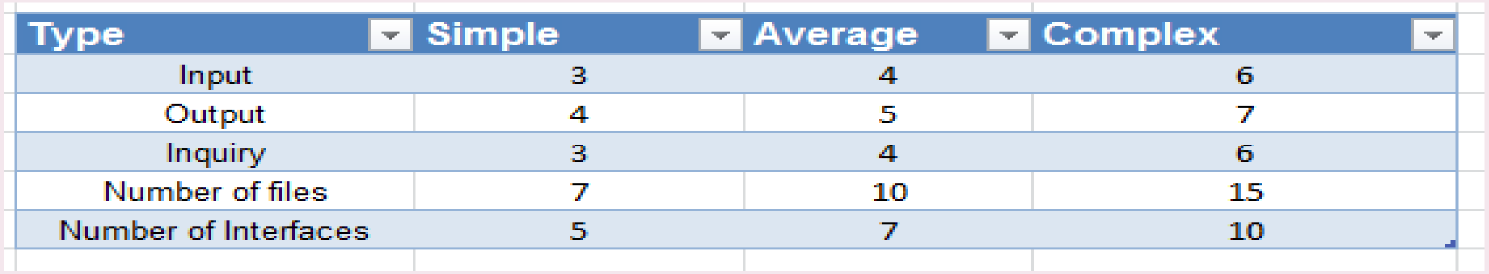
- Software for engineering and embedded systems/applications.
- Software where application complexity is high.
- For example SAP ERP package have different features for purchasing process :
  - Creation of Purchase Order/Automatic availability check of stock for warehouse/plant
  - Creation of Purchase requisition/Conversion of purchase requisition into purchase order
  - Purchase order release strategy using purchase organization structure.
  - Transfer order creation for stock movement with LIFO(last in first out)/FIFO(first in first out) strategies.

- **Feature point (external)**



FEATURE POINT….

- It includes a new software characteristics : “Algorithms”.



Type	Simple	Average	Complex
Input	3	4	6
Output	4	5	7
Inquiry	3	4	6
Number of files	7	10	15
Number of Interfaces	5	7	10

- Steps to get feature point of software as below :
  - Count feature points
  - Continue the feature point count by counting the number of algorithms
  - Weigh complexity.
  - Evaluate environmental factors
  - Calculate the complexity adjustment factor.
  - Multiply the raw feature point count by the complexity adjustment factor

FEATURE POINT (CONT…)

- Another feature point metrics developed by Boeing : “Integrate Data Dimension of Software with functional and control dimensions” known as **3D function Point**

- Boeing 3D 787 Dreamliner Live Flight Tracker

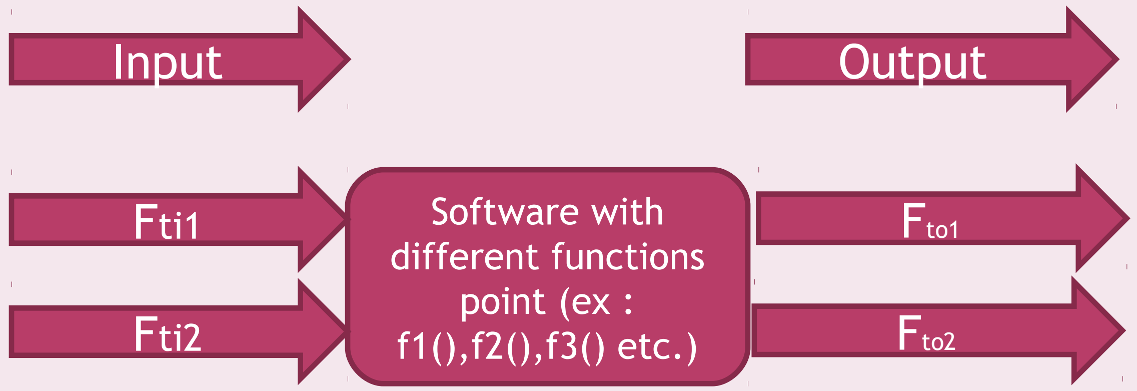
FEATURE POINT (CONT...)

Hybrid system such as :

- Stock Control system with Heavy communication
- Cooling system control process
- Update Control Group
- Read only Control Group
- External Control Data

FEATURE POINT (CONT…)

- Real time software typically contains large number of single occurrence groups of data



Input
Output
Fti1
Fti2
Software with different functions point (ex : f1(),f2(),f3() etc.)
F_{to1}
F_{to2}

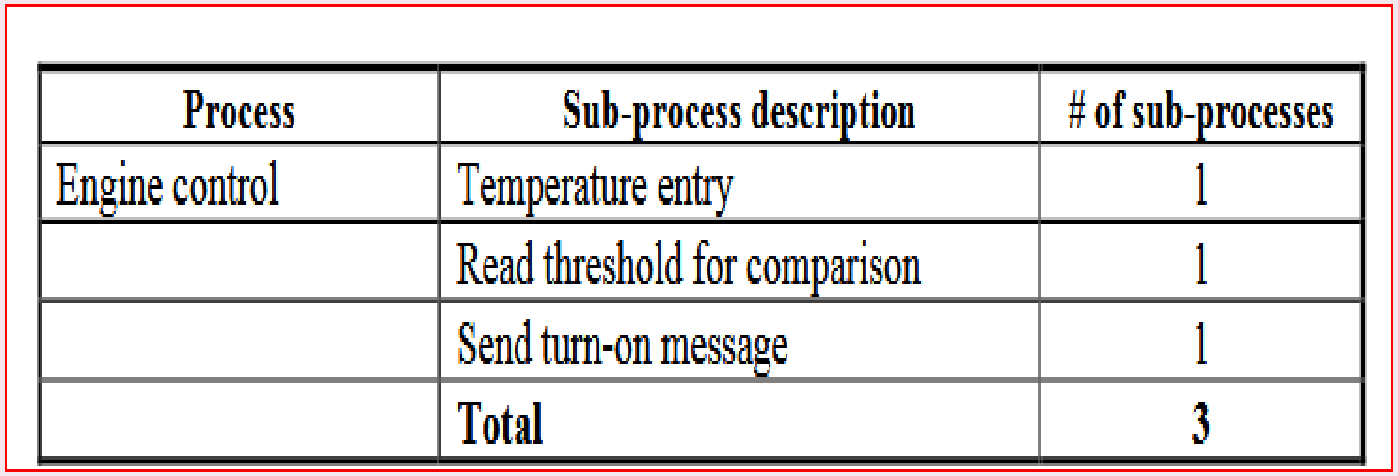
Where Fti : Input feature point
Fto : Output feature point
f1(),f2() : function point of software
Estimation = $\sum_{Fi}^{Fn} Fi\, Fo\, dfi\, dfo$

SOFTWARE FOR PROCESS CONTROL

- An engine temperature control process (process with a few sub- processes)

  Steps follows as below :

  - Output : Turn on Cooling system when required



Process	Sub-process description	# of sub-processes
Engine control	Temperature entry	1
	Read threshold for comparison	1
	Send turn-on message	1
	Total	**3**

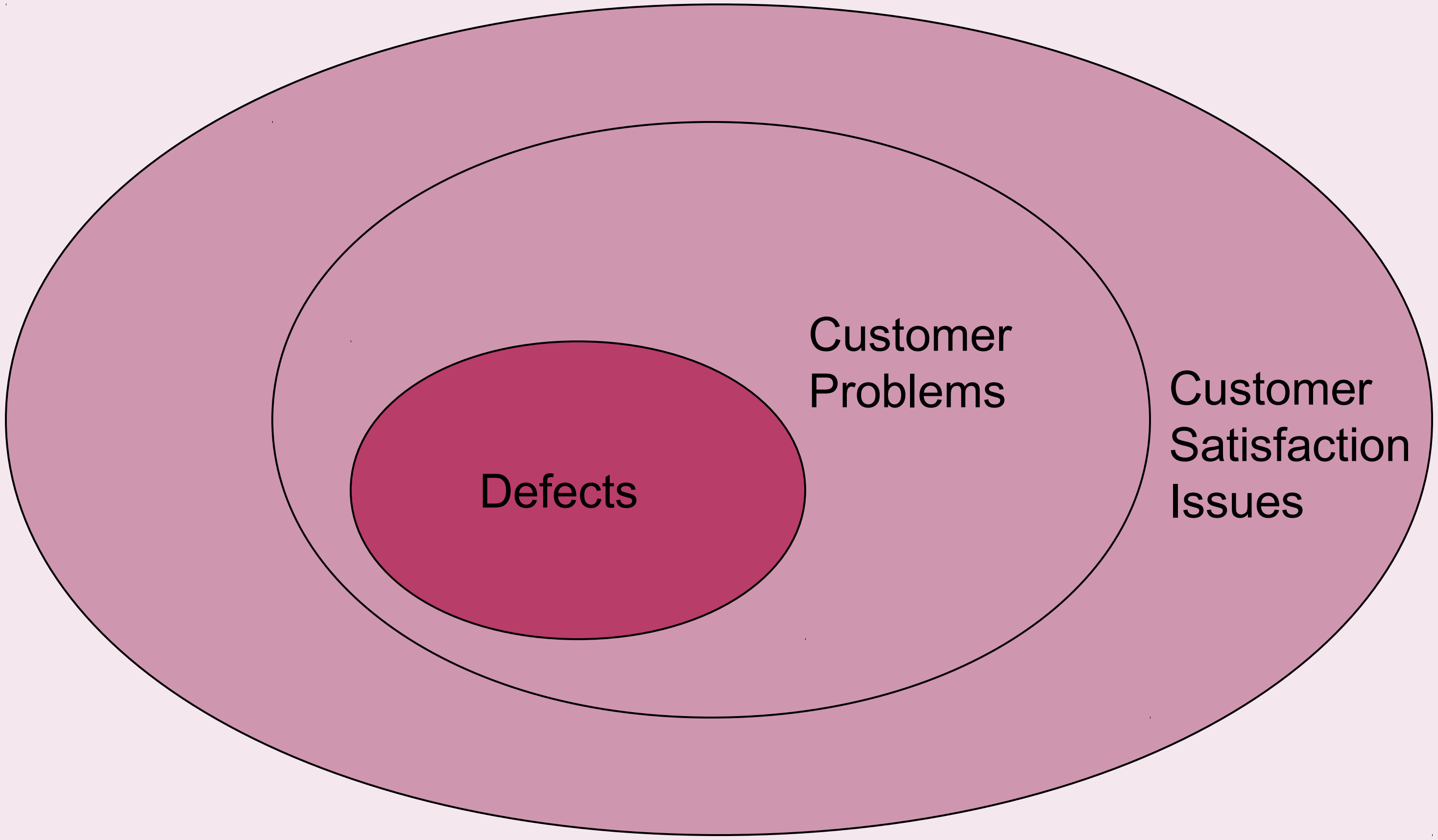
CAPABILITY OF FEATURE POINT

- Feature point metrics are language or platform independent.

- Easily computed from the SRS(Software requirements specification) during project planning.

- It gives an idea of “Effort” and “Time” for software project estimation.

CUSTOMER SATISFACTION METRICS



CUSTOMER SATISFACTION METRICS (CONT'D)

- Customer satisfaction is often measured by customer survey data via the five-point scale:
  - Very satisfied
  - Satisfied
  - Neutral
  - Dissatisfied
  - Very dissatisfied

IBM PARAMETERS OF CUSTOMER SATISFACTION

- CUPRIMDSO
  - Capability (functionality)
  - Usability
  - Performance
  - Reliability
  - Installability
  - Maintainability
  - Documentation
  - Service
  - Overall

HP PARAMETERS OF CUSTOMER SATISFACTION

- FURPS
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Service

EXAMPLES METRICS FOR CUSTOMER SATISFACTION

- Percent of completely satisfied customers
- Percent of satisfied customers (satisfied and completely satisfied)
- Percent of dissatisfied customers (dissatisfied and completely dissatisfied)
- Percent of non-satisfied customers (neutral, dissatisfied, and completely dissatisfied)

IN-PROCESS QUALITY METRICS

- Defect density during machine testing
- Defect arrival pattern during machine testing
- Phase-based defect removal pattern
- Defect removal effectiveness

DEFECT DENSITY DURING MACHINE TESTING

- Defect rate during formal machine testing (testing after code is integrated into the system library) is usually positively correlated with the defect rate in the field.

- The simple metric of defects per KLOC or function point is a good indicator of quality while the product is being tested.

DEFECT DENSITY DURING MACHINE TESTING (CONT'D)

- Scenarios for judging release quality:
  - If the defect rate during testing is the same or lower than that of the previous release, then ask: Does the testing for the current release deteriorate?
    - If the answer is no, the quality perspective is positive.
    - If the answer is yes, you need to do extra testing.

DEFECT DENSITY DURING MACHINE TESTING (CONT'D)

- Scenarios for judging release quality (cont'd):
  - If the defect rate during testing is substantially higher than that of the previous release, then ask: Did we plan for and actually improve testing effectiveness?
    - If the answer is no, the quality perspective is negative.
    - If the answer is yes, then the quality perspective is the same or positive.

DEFECT ARRIVAL PATTERN DURING MACHINE TESTING

- The pattern of defect arrivals gives more information than defect density during testing.
- The objective is to look for defect arrivals that stabilize at a very low level, or times between failures that are far apart before ending the testing effort and releasing the software.

THREE METRICS FOR DEFECT ARRIVAL DURING TESTING

- The defect arrivals during the testing phase by time interval (e.g., week). These are raw arrivals, not all of which are valid.
- The pattern of valid defect arrivals – when problem determination is done on the reported problems. This is the true defect pattern.
- The pattern of defect backlog over time. This is needed because development organizations cannot investigate and fix all reported problems immediately. This metric is a workload statement as well as a quality statement.

PHASE-BASED DEFECT REMOVAL PATTERN

- This is similar to test defect density metric.
- It requires tracking defects in all phases of the development cycle.
- The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.

DEFECT REMOVAL EFFECTIVENESS

- DRE = (Defects removed during a development phase <divided by> Defects latent in the product) x 100%
- The denominator can only be approximated.
- It is usually estimated as:

    Defects removed during the phase +

    Defects found later

Defect Removal Effectiveness (cont'd)

- When done for the front end of the process (before code integration), it is called *early defect removal effectiveness*.
- When done for a specific phase, it is called *phase effectiveness*.

METRICS FOR SOFTWARE MAINTENANCE

- The goal during maintenance is to fix the defects as soon as possible with excellent fix quality
- The following metrics are important:
  - Fix backlog and backlog management index
  - Fix response time and fix responsiveness
  - Percent delinquent fixes
  - Fix quality

TOOLS USED FOR QUALITY METRICS/MEASUREMENTS

- Cause and Effect Diagrams
- Flow Charts
- Checksheets
- Histograms
- Pareto Charts
- Control Charts
- Scatter Diagrams

CAUSE & EFFECT DIAGRAMS-FISHBONE DIAGRAM

Purpose:

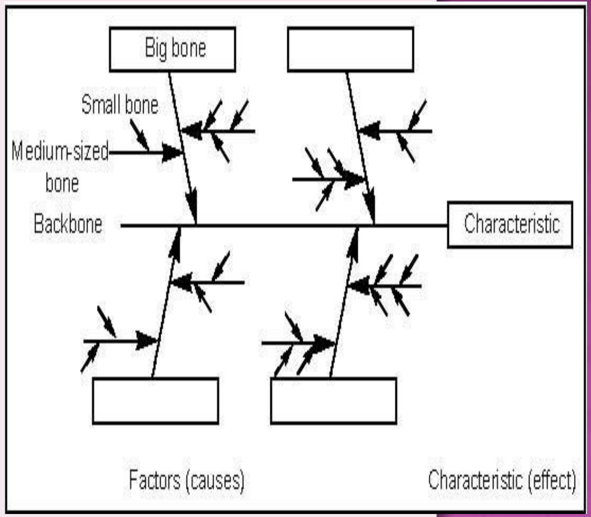
Graphical representation of the trail leading to the root cause of a problem

How is it done?

- Decide which quality characteristic, outcome or effect you want to examine (may use Pareto chart)
- Backbone –draw straight line
- Ribs – categories
- Medium size bones –secondary causes
- Small bones – root causes

Benefits:

- Breaks problems down into bite-size pieces to find root cause
- Fosters team work
- Common understanding of factors causing the problem
- Road map to verify picture of the process
- Follows brainstorming relationship



FLOW CHARTS- LINEAR/TOPDOWN

Purpose:

Visual illustration of the sequence of operations required to complete a task

- ✓ Schematic drawing of the process to measure or improve.
- ✓ Starting point for process improvement
- ✓ Potential weakness in the process are made visual.
- ✓ Picture of process as it *should* be.

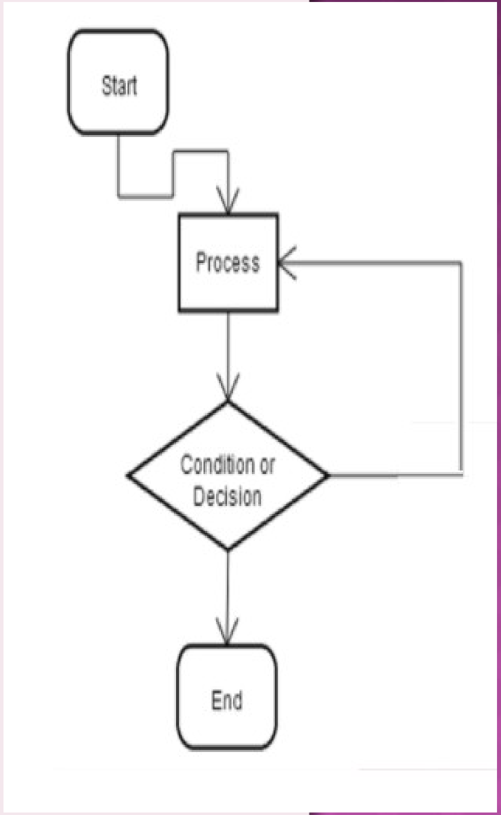
How is it done?

Topdown:

- List major steps
- Write them across top of the chart
- List sub-steps under each in order they occur
- **Linear:**
- Write the process step inside each symbol
- Connect the Symbols with arrows showing the direction of flow

Benefits:

- Identify process improvements
- Understand the process
- Shows duplicated effort and other non-value-added steps
- Clarify working relationships between people and organizations
- Target specific steps in the process for improvement.



CHECKSHEETS

Purpose:

- Tool for collecting and organizing measured or counted data
- Data collected can be used as input data for other quality tools

How is it done?

- Decide what event or problem will be observed. Develop operational definitions.
- Decide when data will be collected and for how long.
- Design the form. Set it up so that data can be recorded simply by making check marks.
- Label all spaces on the form.
- Each time the targeted event or problem occurs, record data on the check sheet.

Benefits:

- Collect data in a systematic and organized manner
- To determine source of problem
- To facilitate classification of data

	Defect Type	Module1	Module2	Module3	Total
Number of Defects	A	|||	||||	||	9
	B	|	|||	|	5
	C	||||		||	6
Total		8	7	5	

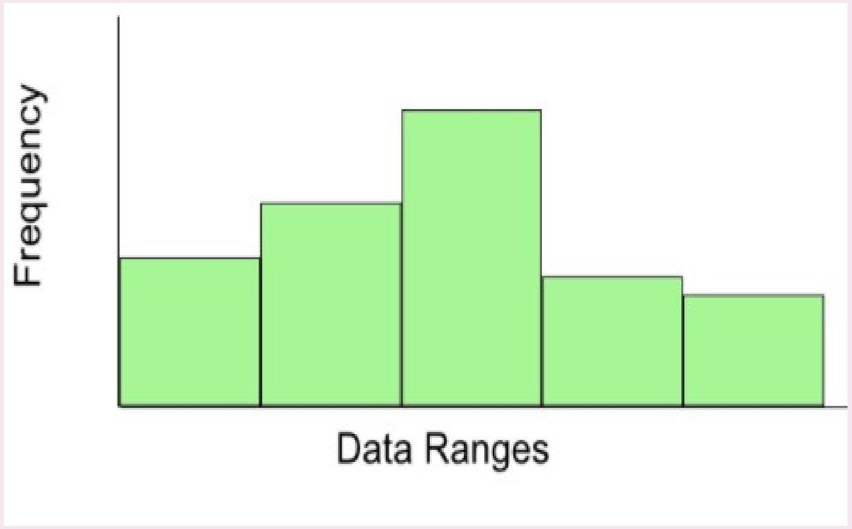
HISTOGRAMS

Purpose:

To determine the spread or variation of a set of data points in a graphical form

How is it done?

- Collect data, 50-100 data point
- Determine the range of the data
- Calculate the size of the class interval
- Divide data points into classes
- Determine the class boundary
- Count # of data points in each class
- Draw the histogram



Benefits:

- Allows you to understand at a glance the variation that exists in a process
- The shape of the histogram will show process behavior
- Often, it will tell you to dig deeper for otherwise unseen causes of variation.
- The shape and size of the dispersion will help identify otherwise hidden sources of variation
- Used to determine the capability of a process
- Starting point for the improvement process

PARETO CHARTS

Purpose:

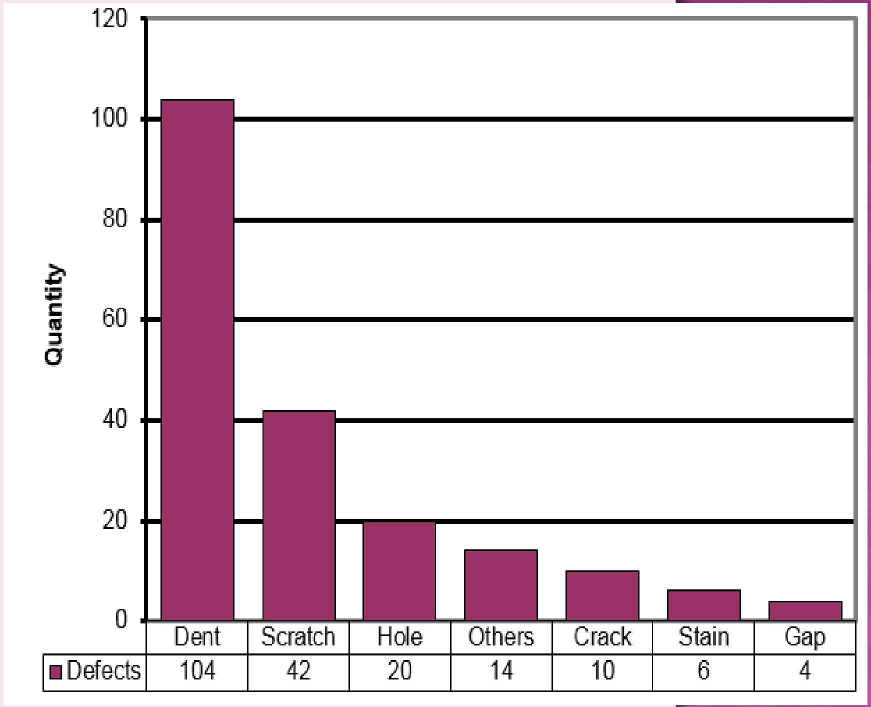
A Pareto chart is a bar graph and depicts which situations are more significant. Prioritize problems.

How is it done?

- Create a preliminary list of problem classifications.
- Tally the occurrences in each problem classification.
- Arrange each classification in order from highest to lowest
- Construct the bar chart

Benefits:

- Pareto analysis helps graphically display results so the significant few problems emerge from the general background
- It tells you what to work on first



CONTROL CHARTS

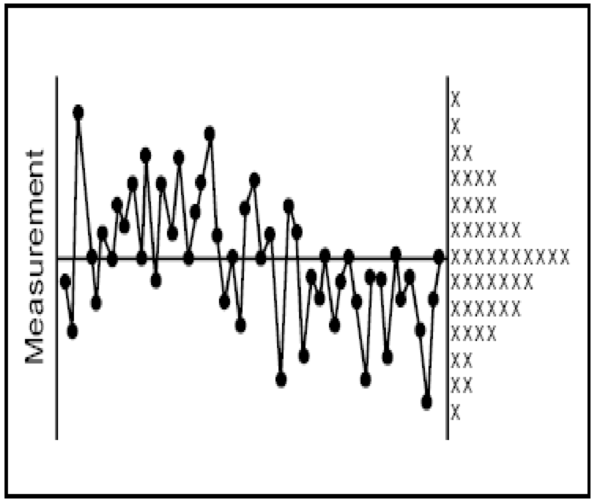
Purpose:

The primary purpose of a control chart is to predict expected product outcome. The control chart is a graph used to study how a process changes over time.

How is it done?

Control Chart Decision Tree:

- Determine Sample size (n)
- Variable or Attribute Data
  - Variable is measured on a continuous scale
  - Attribute is occurrences in n observations
- Determine if sample size is constant or changing



Unusual variation can hide in a frequency plot

Benefits:

- Predict process out of control and out of specification limits
- Distinguish between specific, identifiable causes of variation
- Can be used for statistical process control

SCATTER DIAGRAMS

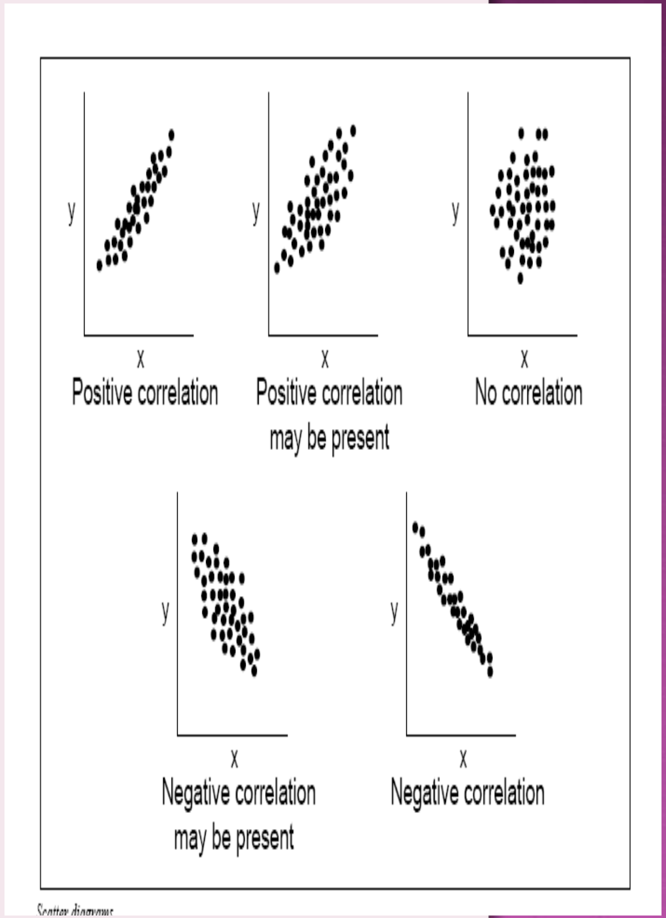
Purpose:

The scatter diagram graphs pairs of numerical data, with one variable on each axis, to look for a relationship between them.

How is it done?

- Decide which paired factors you want to examine. Both factors must be measurable on some incremental linear scale.
- Collect 30 to 100 paired data points.
- Find the highest and lowest value for both variables.
- Draw the vertical (y) and horizontal (x) axes of a graph.
- Plot the data

The shape that the cluster of dots takes will tell you something about the relationship between the two variables that you tested.



Benefits:

- Helps identify and test probable causes.
- By knowing which elements of your process are related and how they are related, you will know what to control or what to vary to affect a quality characteristic.

PROJECT

Combination of interrelated activities

Executed in logical sequence

Accomplishment of a desired objective

HISTORY OF PERT/CPM

PERT

Developed by the US Navy for the planning and control of the **Polaris missile program**

The emphasis was on completing the program in the shortest possible time.

CPM

Developed by **Du Pont** to solve project scheduling problems

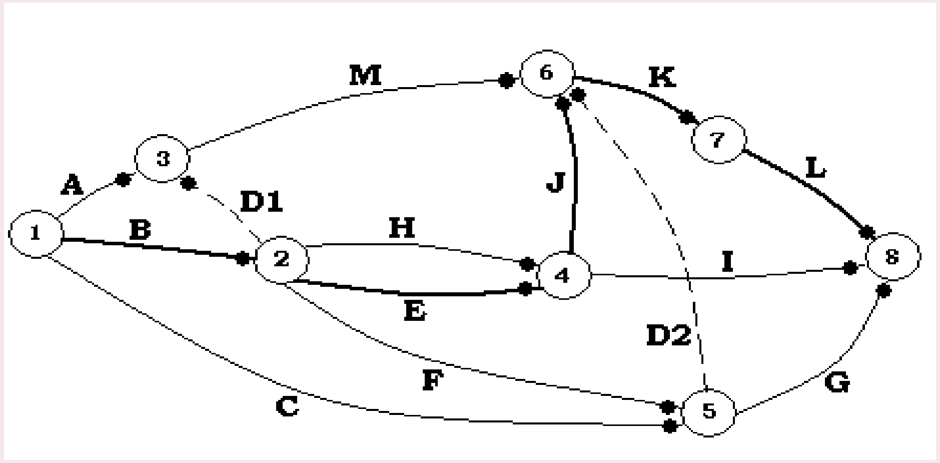
The emphasis was on the trade-off between the cost of the project and its overall completion time

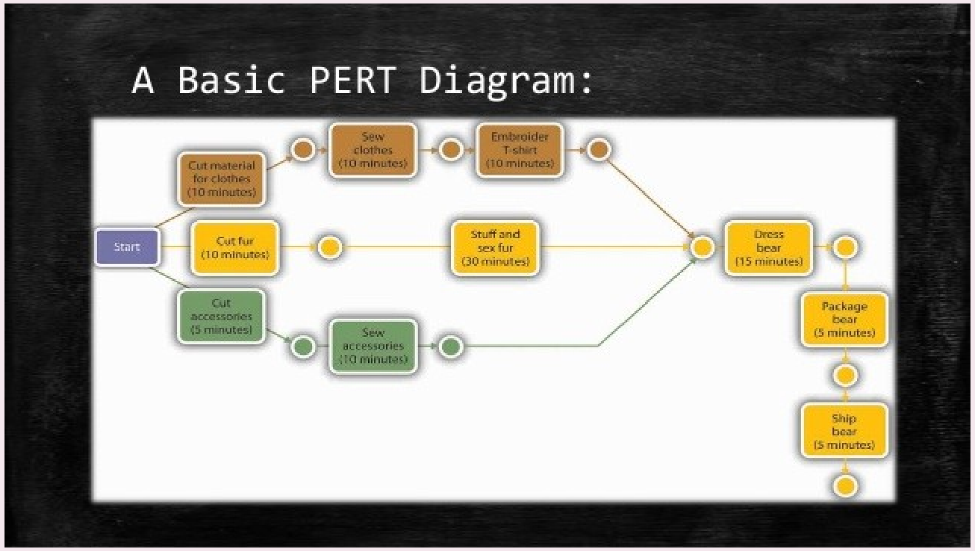
WHY PERT/CPM

- Prediction of deliverables
- Planning resource requirements
- Controlling resource allocation
- Internal program review
- External program review
- Performance evaluation
- Uniform wide acceptance

PERT/CPM EXAMPLE STRUCTURE

A basic CPM Diagram




A Basic PERT Diagram:
Start
Cut material for clothes (10 minutes)
Sew clothes (10 minutes)
Embroider T-shirt (10 minutes)
Cut fur (10 minutes)
Stuff and sex fur (30 minutes)
Dress bear (15 minutes)
Cut accessories (5 minutes)
Sew accessories (10 minutes)
Package bear (5 minutes)
Ship bear (5 minutes)

STEPS IN PERT/CPM

1. PLANNING
2. SCHEDULING
3. ALLOCATION OF RESOURCES
4. CONTROLLING

FRAMEWORK FOR PERT AND CPM

- Define the Project. The Project should have only a single start activity and a single finish activity.
- Develop the relationships among the activities.
- Draw the "Network" connecting all the activities.
- Assign time and/or cost estimates to each activity
- Compute the critical path.
- Use the Network to help plan, schedule, monitor and control the project.

CPM - CRITICAL PATH METHOD

- Drafting the design of Programme or Project
- Evaluation of drafted Programme or Project
- Review of evaluated Programme or Project

CPM CALCULATION

- Draw the CPM network
- Analyze the paths through the network
- Determine the float for each activity
- Compute the activity’s float
- float = LS - ES = LF - EF
- Float is the maximum amount of time that this activity can be delay in its completion before it becomes a critical activity, i.e., delays completion of the project
- Find the critical path is that the sequence of activities and events where there is no “slack” i.e.. Zero slack
- Longest path through a network
- Find the project duration is minimum project completion time

PERT-PROJECT EVALUATION & REVIEW TECHNIQUES

- To analyze and represent the tasks involved in completing a given project
- Accommodates the variation in event completion time
- Event-oriented technique rather than start- and completion-oriented
- Commonly used in conjunction with the critical path method

PERT CALCULATION

- Draw the network.
- Analyze the paths through the network and find the critical path.
- The length of the critical path is the mean of the project duration probability distribution which is assumed to be normal
- The standard deviation of the project duration probability distribution is computed by adding the variances of the critical activities (all of the activities that make up the critical path) and taking the square root of that sum
- Probability computations can now be made using the normal distribution table.
- Determine probability that project is completed within specified time

$$Z = \frac{x - \mu}{\sigma}$$

where $\mu = t_p$ = project mean time

σ = project standard mean time

x = (proposed) specified time

DIFFERENCE BETWEEN PERT & CPM

PERT	CPM
Probabilistic Model	Deterministic Model
Non-repetitive Jobs like planning & scheduling of programmes	Repetitive Jobs like residential construction
Results calculated on basis of Events	Results calculated on basis of activities
Related with activities of uncertain time	Related with activities of Well Known time

ADVANTAGES/DISADVANTAGES

Advantages:

- Reduction in cost
- Minimization of Risk in complex activity
- Flexibility
- Optimization of resources
- Reduction in Uncertainty

Disadvantages:

- Networks charts tend to be large
- Lack of timeframe in charts ,leads to difficulty in showing status
- Skillful Personnel required for planning/implementation