Software Estimation

Syllabus

Unit	Contents	Lectures
2.	Software Estimation • Size • Effort • Cost	**10**

Syllabus …

Unit	Contents	Lectures
2.	Software Metrics • Metrics Database • Process Metrics • Documentation • Line of Code • Review • Quality Metrics • Software size estimation • Function Point Analysis	

Syllabus …

Unit	Contents	Lectures
2.	• Marks II • Function Point Analysis Estimation of Effort and Schedule • Impact of Risk Estimation on Effort and Time • Impact of Schedule and Manpower Constraints • Delphi Cost Estimation Technique • Expert Judgement • COCOMO Model • Application Suite Model	

Syllabus …

Unit	Contents	Lectures
2.	● COCOMO II Model Software Cost Estimation ● Personal Cost ● Hardware Cost ● Software Cost ● Training Cost ● Marketing Cost ● Outsourcing Cost etc.	

What is Measure?

- **Measure** provides a quantitative unit of the object in which one is interested.
- Measure provides a **quantitative indication** of the extent, amount, diamension, capacity, or size of some attribute of a product or process.

Types of Measure

- Direct Measure
- Indirect Measure

Direct Measure

- The Direct Measures are
    - Cost
    - Effort
    - Lines of Code
    - Response Time
    - Resource Usage i.e. memory, disc capacity

Indirect Measure

- The Indirect Measures are
  - Functionality
  - Quality
  - Complexity
  - Efficiency
  - Dependability
  - Maintainability
  - Ease of use
  - Quality of documentation

Measurement

- Measurement is the activity of determining the measure.
- The act of determining a measure is known as measurement.

Metrics

- Metrics is a quantitative measure built out of different measures that represent the entity of interest.
- IEEE has defined metrics as “ A quantitative measure of the degree to which a system, component, or process possesses a given attribute”.
- A metric or combination of metrics that provides insight into the software process, a software project, or the product itself.

Metrics

- Metrics is a quantitative measure built out of different measures that represent the entity of interest.
- IEEE has defined metrics as " A quantitative measure of the degree to which a system, component, or process possesses a given attribute".
- A metric or combination of metrics that provides insight into the software process, a software project, or the product itself.

Activities of Measurement Process

- Formulation → The derivation (i.e., identification) of software measures and metrics appropriate for the representation of the software that is being considered
- Collection → The mechanism used to accumulate data required to derive the formulated metrics
- Analysis → The computation of metrics and the application of mathematical tools
- Interpretation → The evaluation of metrics in an effort to gain insight into the quality of the representation

Activities of Measurement Process

- Feedback → Recommendations derived from the interpretation of product metrics and passed on to the software development team

Characterizing and Validating Metrics

- A metric should have desirable mathematical properties
    - It should have a meaningful range (e.g., zero to ten)
    - It should not be set on a rational scale if it is composed of components measured on an ordinal scale
- If a metric represents a software characteristic that increases when positive traits occur or decreases when undesirable traits are encountered, the value of the metric should increase or decrease in the same manner

Characterizing and Validating Metrics …

- Each metric should be validated empirically in a wide variety of contexts before being published or used to make decisions
  - It should measure the factor of interest independently of other factors
  - It should scale up to large systems
  - It should work in a variety of programming languages and system domains.

Designing Metrics: Collection and Analysis Guidelines

- Whenever possible, data collection and analysis should be automated
- Valid statistical techniques should be applied to establish relationships between internal product attributes and external quality characteristics
- Interpretative guidelines and recommendations should be established for each metric.

Designing Metrics: Collection and Analysis Guidelines

- Whenever possible, data collection and analysis should be automated
- Valid statistical techniques should be applied to establish relationships between internal product attributes and external quality characteristics
- Interpretative guidelines and recommendations should be established for each metric.

Goal Oriented Software Measurement

- Goal/Question/Metric (GQM) paradigm
- GQM technique identifies meaningful metrics for any part of the software process
- GQM emphasizes the need to
  - Establish an explicit measurement goal that is specific to the process activity or product characteristic that is to be assessed
  - Define a set of questions that must be answered in order to achieve the goal
  - Identify well-formulated metrics that help to answer these questions

Goal Oriented Software Measurement …

- GQM utilizes a goal definition template to define each measurement goal
- Example use of goal definition template
  - Analyze the software architecture for the purpose of evaluating architecture components.
  - Do this with respect to the ability to make software more extensible from the viewpoint of the software engineers, who are performing the work in the context of product enhancement over the next three years.

Goal Oriented Software Measurement …

- GQM utilizes a goal definition template to define each measurement goal
- Example use of goal definition template
  - Analyze the software architecture for the purpose of evaluating architecture components.
  - Do this with respect to the ability to make software more extensible from the viewpoint of the software engineers, who are performing the work in the context of product enhancement over the next three years.

Goal Oriented Software Measurement …

- Example questions for this goal definition
  - Are architectural components characterized in a manner that compartmentalizes function and related data?
  - Is the complexity of each component within bounds that will facilitate modification and extension?

What is Software Metrics?

- Software metrics is built out of several measures and provides an insight into various aspects of software like software processes, software product etc.
- The metrics data are collected over a period on several software are useful to build standards for planning and estimation of resource, cost, effort, software size and time for software development.
- Software metrics can be used to indicate the basic attributes of software.

What is Software Metrics?

Size	Line of Code, Function Point Count
Quality	Reliability, Accuracy and Dependability and are measured through errors, failures and number of runs between failures
Execution Time	Process time of execution of a critical process etc.

Software Metrics

Day	Time	Body Temp.	Time Taken	Season
MON	6:00	97.0 F	12.00 Sec	Spring
TUE	6:00	96.8 F	12.05 Sec	Spring
WED	5:00	97.5 F	11.90 Sec	Spring
THU	5:30	97.0 F	11.95 Sec	Spring
FRI	6:00	97.4 F	12.01 Sec	Spring
SAT	5:00	96.9 F	11.92 Sec	Spring
SUN	8:00	98.0 F	12.20 Sec	Spring

Software Metrics

- Software Engineering is a stable and quantitative engineering discipline.
- The stability arises from wide range of matrices evolved by the software engineers to measure various aspects of the software.
- The advantage of metrics is that we can measure different aspects of software that need evaluation on an ongoing basis for estimation in quantitative terms.
- The terms Measure, Measurement and Metrics are different aspects.

Software Metrics …

- Measure provides a quantitative unit of the object in which one is interested. Provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process.
- Measurement is the activity of determining the measure.
- The act of determining a measure is termed as measurement.
- Metrics is a quantitative measure built out of different measures that represent the entity of interest. (IEEE)

Software Metrics …

- A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

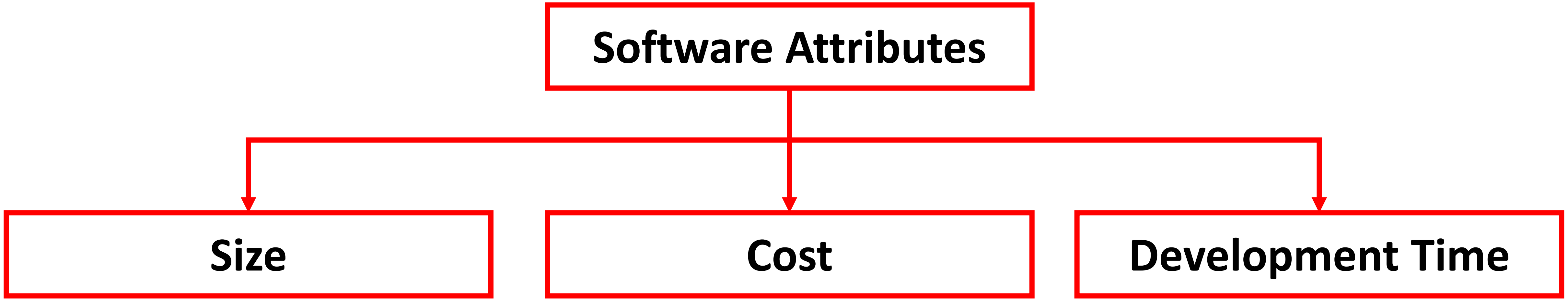
Software Attributes

- A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Types of Software Attributes

- Size
- Cost
- Development Time

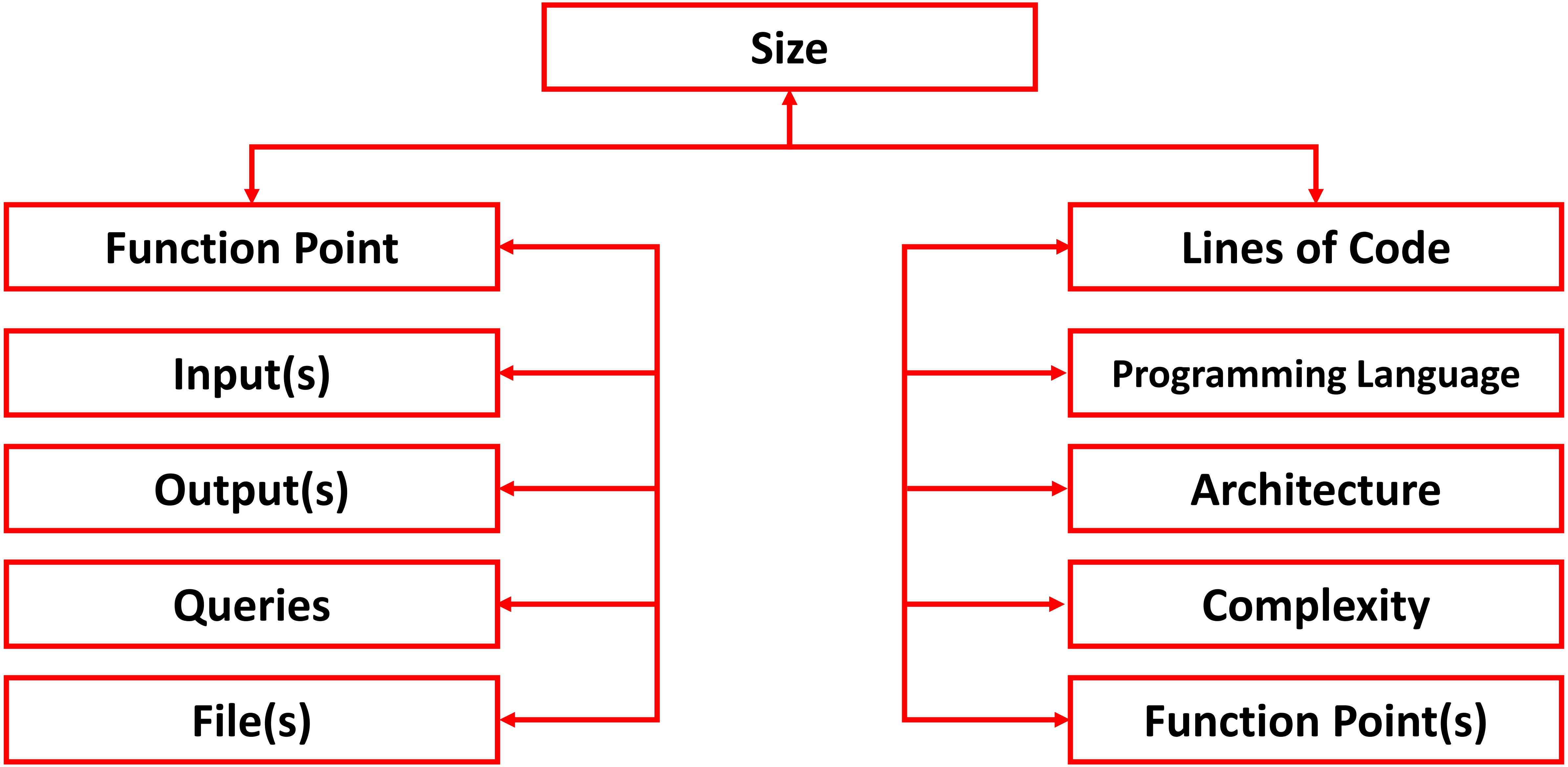
Software Attributes …



Size of Software

- Size of software depends on
  - Lines of Code
  - Function Point(s)

Software Metrics …



Size of Software …

- Line of Code depends on
  - Language
  - Architecture
  - Complexity
  - Function Point

Size of Software …

- Function Point depends on
  - Input
  - Output
  - Queries
  - Files

Project Size Categories

- Project size is a major factor that determines the level of management control, types of tools and techniques required for the software project.
- As per size, project is of following types: -
  - Trivial Project
  - Small Project
  - Medium Size Project
  - Large Project
  - Very Large Project
  - Extremely large System

Trivial Project

- Number of Programmer →One programmer develops a trivial project.
- Duration → The programmer perhaps works part time for a few days or for a few weeks to develop the software.
- Software Size →The programs are not having more than 500 source code lines packaged in 10 – 20 routines.
- Such programs are developed exclusively for personal use and are usually discarded after few months. There is a little need of formal analysis, elaborate design, documentation, extensive test planning or supporting documents for trivial program.

Small Size Project

- Number of programmers →A small project employs more than one programmers.
- Duration → Programmers usually take 1 – 6 months to develop the software.
- Software Size → Software contains 1000- 2000 lines of source codes in perhaps 25-50 sub routines.
- Small projects include scientific app used by engineer to solve numerical problems, small commercial app written by data processing personal to perform straight forward data manipulation and resource generation.

Small Size Project …

- A small project requires a little interaction with the programmer or between programmer and customer.
- Degree of formality would be much less than it is required on the large projects.

Medium Size Project

- Number of programmers → 2 – 3 programmers.
- Duration → 1 – 2 years to complete the project.
- Software Size → 10000 – 50000 lines of source codes packaged in 250-1000 routines.
- Some of them are management information system, inventory control system etc.
- Development of medium size project requires interaction among programmers and the communication with customer.
- A certain degree of formality is required in planning of document and project reviews.

Large Size Project

- Number of programmers → 5 – 20
- Duration → 2 – 3 Years.
- Software Size → 50000 – 100000 lines of source code packaged in several sub systems.
- A large project often has significant interaction with other programs and software system. For example compilers, database packages, real time control system etc.
- Communication among developer and programmer becomes more severe.

Large Size Project …

- A large project requires more than one programming team which often involves more than one level of management.
- The size and complexity of large project makes it difficult to foresee or even eventualities during planning and analysis.
- Systematic process, standardized document and common reviews are essential throughout the project.

Very Large Size Project

- Number of programmers → 100 – 1000
- Duration → 4 – 5 Years.
- Software Size → 1,000,000 lines of source codes.
- A very large system generally consists of several major sub systems.
- Each of which form a large system.
- The examples are operating system, database system, systematic processes, standardized document and common reviews are essential throughout the large project.

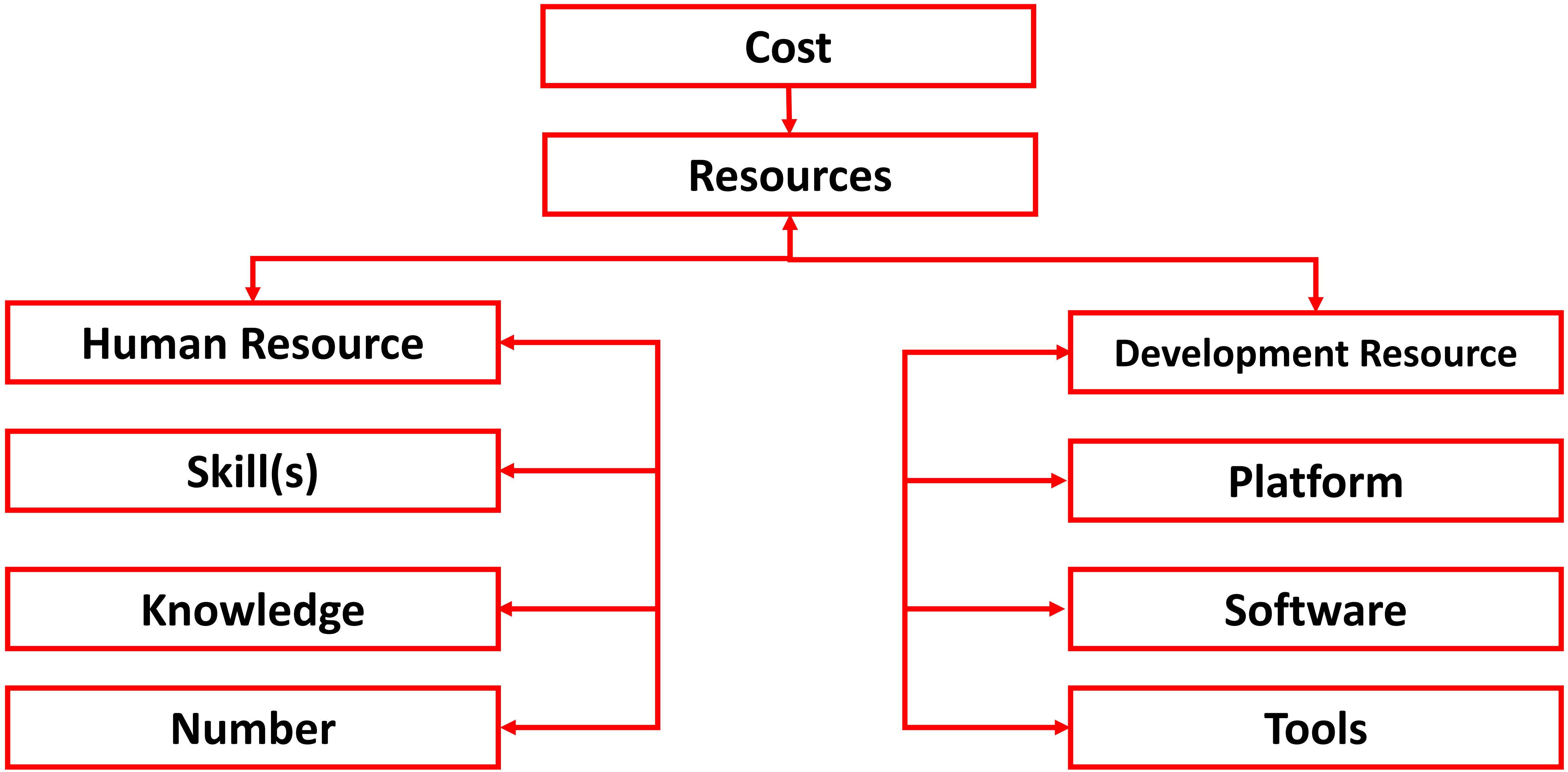
Extremely Large Size Project

- Number of programmers → 2000 – 5000
- Duration → Upto 10 Years.
- Software size → 10,000,000 – 100,000,000 lines of source code.
- Extremely large system consists of several large sub system and often involved real time processing, telecommunication, multi-tasking and distributive processing.
- These system often have extremely high reliability requirement and often involve life and processes.
- The examples are air traffic control, ballistic missile defense system etc.

Cost of Software

- Cost of Software depends on resource utilization.
- Resources are of two types
  - Human Resource
  - Development Resource

Software Metrics …



Cost of Software

- Cost of Human Resource depends on
  - Skills
  - Knowledge
  - Number

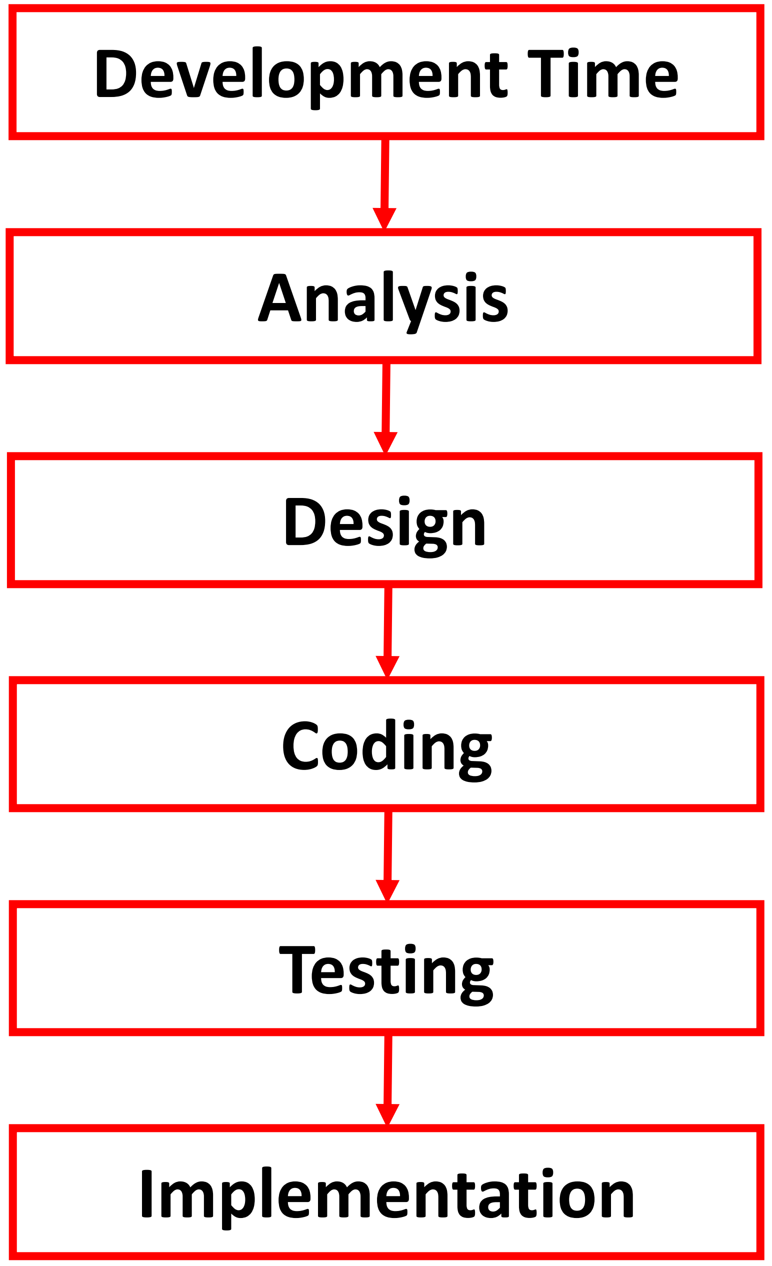
Cost of Software

- Cost of Development Resource depends on
  - Platform
  - Software
  - Tools

Development Time

- Development Time depends on time taken in
  - Analysis
  - Design
  - Coding
  - Testing
  - Implementation

Development Time …



Attributes of Software Metrics

- The beginning of constructing effective software metrics is done by selecting appropriate measures and the process management.
- It is very difficult to arrive at one single value or attribute to evaluate the software.
- The stress is on to build up a variety of measures and the concerned personnel will frame metrics based on these measures.
- Once the appropriate measures is selected, process of management is required to those measures.

Attributes of Software Metrics …

- The process of management has following attributes
  - Appropriate choice of process management for software
  - Appropriate methods of data collection
  - Appropriate mathematical tools for compilation of measure
  - Resultant measurement that provides an insight into the relevant aspect of the software.

Attributes of Software Metrics …

- The above stated attributes can be measured if the measurement process works on the following principles
  - The objectives of measurement are clear
  - Data collection, analysis and computation are automated to eliminate bias.
  - Valid statistical techniques are used for data collection and computation of measures.
  - Simple to learn
  - Easy to compute
  - Unambiguous

Attributes of Software Metrics …

- Clear in objective terms for application
- Independent of technology, architecture or programming languages.
- Simple and computable
- It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time
- Empirically and intuitively persuasive
- The metric should satisfy the engineer's intuitive notions about the product attribute under consideration

Attributes of Software Metrics …

- Consistent and objective
- The metric should always yield results that are unambiguous.
- Consistent in the use of units and dimensions
- The mathematical computation of the metric should use measures that do not lead to bizarre combinations of units
- Programming language independent
- Metrics should be based on the analysis model, the design model, or the structure of the program itself
- An effective mechanism for high-quality feedback
- The metric should lead to a higher-quality end product

Software Metrics Database

- A disciplined SE approach using engineering principles assesses the following entities:
  - Method
  - Process
  - Environment
  - People
  - Technology
  - Software Product

Software Metrics Database …

- Software organizations build metrics for main entities such as software product, development process by collecting data through direct and indirect measures from successful internal projects.
- The measures are then used to build relevant metrics as indicators or guidelines for assessment.
- Metrics are built for three main entities: -
  - Process
  - Product
  - Project

Software Metrics Database …

- Process Metrics
  - Process metrics are used for process improvement.
- Product Metrics
  - Product metrics are used for product improvement and to distinguishing factors.
- Project Metrics
  - Project metrics are used for software project planning and control.

Software Metrics Database …

- The common entities for which a metrics database is developed are
    - Process
    - Project(Product)
    - Quality
- The metrics database is used to assess and monitor the progress of the software development, identify problem areas that may become critical, manipulate activities for effective resource usage and support team effort to meet the project deadline.

Type of Metrics

- Base Line Metrics
- Process Metrics
- Project Metrics
- Analysis Model Metrics
- Design Model Metrics
- Source Code Metrics
- Testing Metrics
- Maintenance Metrics

Baseline Metrics

- An organization develops a baseline of metrics as a reference for planning and assessment of new software.
- Baselines are created using data as a measure built from several software products and projects developed earlier.
- The data used for baselines should come from a valid source, from similar software environments and should be accurate and reliable.
- With the growth in projects and products, the baseline metrics improve due to additional data from new projects.

Baseline Metrics …

- The development of baselines metrics is a continuous process in a matured development organization.
- Baseline metrics are used as a starting point for technical guidance, building business proposals, planning and resource estimation and subsequently to control the development of the new software.

Process Metrics

- Software process and project metrics are quantitative measure.
- They are management tool.
- They offer insight into effectiveness of the software process and the projects that are conducted using the process as a framework.
- Basic quality and productivity data are collected.
- These data are analyzed, compared against past averages and assessed.
- The goal is to determine whether quality and productivity improvements have occurred.

Process Metrics …

- The data can also be used to pinpoint the problem areas.
- Remedies can be then developed and the software process can be improved.
- The process of software development is influenced by
  - The Customer
  - Business
  - Development environment
- These have significant impact on process performance.

Process Metrics …

- The customer environment stems from the customer need and characteristics of those in the customer organization who will use the software.
- The customer environment depends upon the customer and the user.
- Software requirements will be clearly spelt out and agreed to by all users and stakeholders.

Process Metrics …

- Ease of use i.e. user friendly is the commonly required attribute.
- If the business environment is complex as a result of business rules, types of business functions, nature and size of business, then this has significant impact onto process and performance of the organization.
- The development resource environment is considered good if the skills and knowledge required to deliver the software is available within the organization.
- If not then the process as well as organizational performance will be adverse.

Process Metrics …

- The customer and business environment are beyond the control of the organization.
- Although some control can be exercised on development process.
- The process and organizational performance is also affected by nature of software product, quality of people and technology.
- Due to complexity of such factors affecting the process, there are no direct measures available to build the metrics.
- Indirect measures like errors, defects and bugs repaired during testing, demonstration etc.

Process Metrics …

- It is possible to compute efforts taken to develop the product, time taken for development etc.
- Indirect measures measuring the performance of critical activities like requirement analysis, planning, designing etc.

Process Metrics …

- Uses of Measurement
  - Can be applied to the software process with the intent of improving it on a continuous basis.
  - Can be used throughout a software project to assist in estimation.
  - Can be used to help assess the quality of software work products and to assist in tactical decision making as a project proceeds.

Process Metrics …

- Reasons to Measure
    - To characterize in order to
        - Gain an understanding of processes, products, resources and environments.
        - Establish baselines for comparisons with future assessments.
        - To evaluate
        - Determine status with respect to plans.

Process Metrics …

- To predict in order to
  - Gain understanding of relationships among processes and products.
  - Build models of these relationships.
  - To improve in order to
  - Identify roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance.

Development of Process Metrics

- Process metrics are collected across all projects and over long periods of time.
- They are used for making strategic decisions.
- The intent is to provide a set of process indicators that lead to long term software process improvement.
- The only way to know how/where to improve any process is to
  - Measure specific attributes of the process.
  - Develop a set of meaningful metrics based on these attributes.

Development of Process Metrics …

  - Use the metrics to provide indicators that will lead to a strategy for improvement.
- We measure the effectiveness of a process by deriving a set of metrics based on outcomes of the process such as
  - Errors uncovered before release of software.
  - Defects delivered to and reported by the end users.
  - Work products delivered.
  - Human effort expanded.
  - Conformance to the schedule.
  - Time and effort to complete each generic activity.

Etiquette of Process Metrics

- Use common sense and organizational sensitivity when interpreting metrics data.
- Provide regular feedback to the individuals and teams who collect measures and metrics.
- Don’t use metrics to evaluate individuals.
- Work with practitioners and teams to set clear goals and metrics that will be used achieve them.
- Never use metrics to threaten individuals or teams.

Etiquette of Process Metrics …

- Metrics data that indicate a problem should not be considered "negative"
    - Such data are merely an indicator for process improvement.
- Don't obsess on a single metric to the exclusion of other important metrics.

Source of Error and Their Distribution

Source / Cause	Reason of Error/ Effects	% age of Errors Anticipated
Requirement Analysis and Specification	Business rules and logic. Number of conditions and constraints.	25
Requirement Analysis and Specification	Data Management, Definition Structure and Application.	10

Source of Error and Their Distribution

Source / Cause	Reason of Error/ Effects	% age of Errors Anticipated
Requirement Analysis and Specification	Use of Standards	5
Design	Design specification	30
Code	Testing	15
User Interface	Design	5
Hardware Interface	Design	5
Software Interface	Design	5

Process Metrics …

- The choice of process model is very important if the organization has to control the quality and performance of the software.
- The selection of the appropriate process model for the given environment in which the solution is demanded is extremely critical.
- The organization's capability to handle each phase in the most efficient manner will control the incidence of error during the life cycle.
- Hence, process metrics differ from organization to organization, resulting in different software solution proposals differing from each other in the technical, commercial and management aspects.

Project Metrics

- It is used by the project manager to control and coordinate the project in terms of project cost, time and effort through management of skills, customer relations, technology of development and software solution design.
- It is also used to plan and execute life cycle activities.
- The most advantageous use of project metrics is in estimation of various aspects of new software project.

Project Metrics ...

- It provides data on estimation of time, effort, resource, activities errors etc through certain basic measures like function point, line of code, pages in the documentation, number of reviews i.e. requirement analysis, design, code etc.
- Project metrics enable a software project manager to
  - Assess the status of an ongoing project.
  - Track potential risks.
  - Uncover problem areas before their status becomes critical.
  - Adjust work for flow or tasks.

Project Metrics …

  - Evaluate the project team's ability to control quality of software work products.
- Many of the same metrics are used in both the process and project domain.
- Project metrics are used for making tactical decisions.
- They are used to adapt project workflow and technical activities.
- The first application of project metrics occurs during estimation
  - Metrics from past projects are used as a basis for estimating time and effort.

Project Metrics …

- As a project proceeds, the amount of time and effort expended are compared to original estimates.
- As technical work commences, other project metrics becomes important
  - Production rates are measured (represented in terms of models created, review hours, function points and delivered source lines of code).
  - Error uncovered during each generic framework activity i.e. communication, planning, modeling, construction, deployment etc. are measured.

Project Metrics …

- Project metrics are used to
  - Minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks.
  - Assess product quality on an ongoing basis and when necessary, to modify the technical approach to improve quality.
  - As quality improves, defects are minimized.
  - As defects go down, the amount of rework required during the project is also reduced.
  - As rework goes down, the overall project cost is reduced.

Analysis Model Metrics

- Functionality delivered
  - Provides an indirect measure of the functionality that is packaged within the software.
- System size
  - Measures the overall size of the system defined in terms of information available as part of the analysis model.
- Specification quality
  - Provides an indication of the specificity and completeness of a requirements specification.

Design Metrics

- Architectural Design Metrics
- Component Level Metrics
- Interface Design Metrics
- Hierarchical Architecture Metrics
- Object – Oriented Design Metrics
- Specific Class – Oriented Design Metrics

Architectural Design Metrics

- These metrics place emphasis on the architectural structure and effectiveness of modules or components within the architecture.
- They are “**black box**” in that they do not require any knowledge of the inner workings of a particular software component.
- Provide an indication of the quality of the architectural design.

Component Level Design Metrics

- Measure the complexity of software components and other characteristics that have a bearing on quality

Interface Design Metrics

- Focuses on usability.

Hierarchical Architectural Design Metrics

- Fan Out → the number of modules immediately subordinate to the module i, that is, the number of modules directly invoked by module i
- Structural Complexity → $S(i) = fout(i)$, where $fout(i)$ is the "fan out" of module i
- Data complexity → $D(i) = v(i)/[fout(i) + 1]$, where $v(i)$ is the number of input and output variables that are passed to and from module i
- System Complexity → $C(i) = S(i) + D(i)$

Hierarchical Architectural Design Metrics

- With the increase in the value of the above complexity, the overall architectural complexity of the system also increases and this leads to greater likelihood that the integration and testing effort will also increase
- Shape complexity → size = n + a, where n is the number of nodes and a is the number of arcs
    - Allows different program software architectures to be compared in a straightforward manner

Hierarchical Architectural Design Metrics

- Connectivity Density (i.e., the arc-to-node ratio) → r = a/n
  - May provide a simple indication of the coupling in the software architecture

Object Oriented Design Metrics

- Size
  - Population → A static count of all classes and methods
  - Volume → A dynamic count of all instantiated objects at a given time
  - Length → The depth of an inheritance tree
- Coupling → The number of collaborations between classes or the number of methods called between objects
- Cohesion → The cohesion of a class is the degree to which its set of properties is part of the problem or design domain.

Object Oriented Design Metrics …

- Primitiveness → The degree to which a method in a class is atomic (i.e., the method cannot be constructed out of a sequence of other methods provided by the class)

Specific Class Oriented Design Metrics

- Specific Class – Oriented Metrics
- Weighted methods per class
  - The normalized complexity of the methods in a class
  - Indicates the amount of effort to implement and test a class
- Depth of the Inheritance Tree

  → The maximum length from the derived class (the node) to the base class (the root).

  → Indicates the potential difficulties when attempting to predict the behavior of a class because of the number of inherited methods

Specific Class Oriented Design Metrics …

- Number of children (i.e., subclasses)
  - As the number of children of a class grows
  - Reuse increases
  - The abstraction represented by the parent class can be diluted by inappropriate children
  - The amount of testing required will increase
- Measure characteristics of classes and their communication and collaboration characteristics.

Specific Class Oriented Design Metrics …

- Coupling between object classes
  - → Measures the number of collaborations a class has with any other classes
  - → Higher coupling decreases the reusability of a class
  - → Higher coupling complicates modifications and testing
  - → Coupling should be kept as low as possible

Specific Class Oriented Design Metrics …

- Response for a class
  - → This is the set of methods that can potentially be executed in a class in response to a public method call from outside the class
  - → As the response value increases, the effort required for testing also increases as does the overall design complexity of the class

Specific Class Oriented Design Metrics …

- Lack of cohesion in methods

  → This measures the number of methods that access one or more of the same instance variables (i.e., attributes) of a class

  → If no methods access the same attribute, then the measure is zero

  → As the measure increases, methods become more coupled to one another via attributes, thereby increasing the complexity of the class design

Source Code Metrics

- Complexity Metrics → Measure the logical complexity of source code (can also be applied to component-level design)
- Length Metrics → Provide an indication of the size of the software

Testing Metrics

- Statement and Branch Coverage → Metrics Lead to the design of test cases that provide program coverage
- Defect – Related Metrics → Focus on defects (i.e., bugs) found, rather than on the tests themselves
- Testing Effectiveness Metrics → Provide a real-time indication of the effectiveness of tests that have been conducted
- In – Process Metrics → Process related metrics that can be determined as testing is conducted
- Information Domain Values
- Number of external inputs
- Each external input originates from a user or is transmitted from

Maintenance Metrics

- Software Maturity Index (SMI) → Provides an indication of the stability of a software product based on changes that occur for each release
- SMI = $[M_T - (Fa + Fc + Fd)]/M_T$ where
  - M_T = #modules in the current release
  - F_a = #modules in the current release that have been added
  - F_c = #modules in the current release that have been changed
  - F_d = #modules from the preceding release that were deleted in the current release

Maintenance Metrics …

- As the SMI (i.e., the fraction) approaches 1.0, the software product begins to stabilize
- The average time to produce a release of a software product can be correlated with the SMI.

Attributes of (Software) Metrics

- The beginning of constructing effective software metrics is done by selecting appropriate measures and the process management.
- It is very difficult to arrive at one single value or attribute to evaluate the software.
- So, the stress is on to build up a variety of measures and the concerned personnel will frame metrics based on these measures.
- Once the appropriate measures is selected, process of management is required to those measures.
- The process of management has following attributes

Attributes of (Software) Metrics …

- →Appropriate choice of process management for software
- →Appropriate methods of data collection
- →Appropriate mathematical tools for compilation of measure
- →Resultant measurement that provides an insight into the relevant aspect of the software

Attributes of (Software) Metrics …

- The stated attributes can be measured if the measurement process works on the following principles
    - The objectives of measurement are clear
    - Data collection, analysis and computation are automated to eliminate bias.
    - Valid statistical techniques are used for data collection and computation of measures.
    - Simple to learn
    - Easy to compute
    - Unambiguous

Attributes of (Software) Metrics …

- Clear in objective terms for application
- Independent of technology, architecture or programming languages
- Simple and computable
- It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time
- Empirically and intuitively persuasive
- The metric should satisfy the engineer’s intuitive notions about the product attribute under consideration

Attributes of (Software) Metrics …

- Consistent and objective
- The metric should always yield results that are unambiguous
- Consistent in the use of units and dimensions
- The mathematical computation of the metric should use measures that do not lead to bizarre combinations of units
- Programming language independent
- Metrics should be based on the analysis model, the design model, or the structure of the program itself
- An effective mechanism for high – quality feedback
- The metric should lead to a higher-quality end product

Metrics Database

- A disciplined SE approach using engineering principles assesses the following entities: -
  - Method
  - Process
  - Environment
  - People
  - Technology
  - Software Product

Metrics Database …

- Software organizations build metrics for main entities such as software product, development process by collecting data through direct and indirect measures from successful internal projects.
- The measures are then used to build relevant metrics as indicators or guidelines for assessment.
- Metrics are built for three main entities:
  - Process → Process metrics are used for process improvement.
  - Product → Product metrics are used for product improvement and to distinguishing factors.

Metrics Database …

- Project $\rightarrow$ Project metrics are used for software project planning and control.

- The common entities for which a metrics database is developed are
  - Process
  - Project(i.e. Product)
  - Quality

Metrics Database …

- The metrics database is used to assess and monitor the progress of the software development, identify problem areas that may become critical, manipulate activities for effective resource usage and support team effort to meet the project deadline.

Estimation of Size of Software

- Process Metrics
- Documentation
- Line of Code
- Reviews
- Quality Metrics

Estimation of Effort and Schedule

- Delphi Cost Estimation Technique

Impact of Risk Estimation on Effort and Time

- Delphi Cost Estimation Technique

Impact of Schedule and Manpower Constraint

- Delphi Cost Estimation Technique

Expert Judgement

- It is top down management technique.
- Expert judgement relies on the judgement, background and business sense of one or more key people in the organization.
- An expert might arrive at the cost estimate in the following manner.
    - The system to be developed is a process control system similar to the one that was developed one year ago at a cost of 10 lakhs, we made a respectable profit with that project.
    - The new project has 25% more activity and hence we consume more time and cost to develop and so on.

Disadvantages of Expert Judgement

- The biggest quality on expert judgement name the experience can also be liability.
- The expert may be confident that the project is similar with the previous one but may have overlooked some factor that make the new project significantly different or the expert making the judgement or the estimate may not have the experience with the project similar to the present one.
- To overcome this, group of experts some time prepare a consensus estimates.

Disadvantages of Expert Judgement …

- They try to minimize the individual oversights and lack of familiarity with the particular project.
- This effort neutralizes the present biases and desire to gain control through an overly optimistic estimate.

Delphi Cost Estimation Technique

- Developed at RAND Corporation in 1951.
- Developed by Norman Dalkey and Olaf Helmer.
- Structured communication technique or method
- Developed as a systematic, interactive forecasting method which
- Relies on a panel of experts
- The technique can also be adapted for use in one to one i.e. face-to-face meetings
- Used for business forecasting

Delphi Cost Estimation Technique …

- Delphi is based on the principle that forecasts (or decisions) from a structured group of individuals / experts are more **accurate** than those from unstructured groups.
- The experts answer **questionnaires** in two or more rounds.
- After each round, a **facilitator** provides an anonymous summary of the expert's forecasts from the previous round as well as the reasons they provided for their judgments.
- Experts are encouraged to revise their earlier answers in light of the replies of other members of their panel.

Delphi Cost Estimation Technique …

- During this process the range of the answers will decrease and the group will converge towards the "**correct**" answer.
- The process is stopped after a predefined stop criterion
  - Number of rounds
  - Achievement of consensus
  - Stability of results
- The mean or median scores of the final rounds determine the results

Important Features of Delphi Cost Estimation Technique

- Anonymity (Hiding the identity) of the participants
- Structured flow of information
- Regular feedback
- Role of the facilitator

Anonymous Participants

- All participants remain anonymous.
- Their identity is not revealed, even after the completion of the final report. This prevents
  - The Authority
  - Personality
  - Reputation of participants

from dominating others in the process.

Anonymous Participants …

- Frees participants from their
  - Personal Biases
  - Allows free expression of opinions
  - Encourages open critique
  - Facilitates admission of errors when revising earlier judgments

Structured Flow of Information

- The initial contributions from the experts are collected in the form of answers to questionnaires and their comments to these answers.
- The panel director controls the interactions among the participants by processing the information and filtering out irrelevant content.
- This avoids the negative effects of face-to-face panel discussions and solves the usual problems of group dynamics.

Regularity in Feedback

- The Delphi Method allows participants to comment on the responses of others, the progress of the panel as a whole, and to revise their own forecasts and opinions in real time.

Role Played by Facilitator

- The person coordinating the Delphi method is usually known as a facilitator or Leader, and facilitates the responses of their panel of experts, who are selected for a reason, usually that they hold knowledge on an opinion or view.
- The facilitator sends out questionnaires, surveys etc. and if the panel of experts accept, they follow instructions and present their views.
- Responses are collected and analyzed, then common and conflicting viewpoints are identified.

Role Played by Facilitator …

- If consensus is not reached, the process continues through thesis and antithesis, to gradually work towards synthesis, and building consensus.

Implementing Delphi Estimation Technique

- Choosing a Facilitator
- Identifying Your Experts
- Defining the Problem i.e. questionaire
- First Round of Evaluation of Questions
- Second Round of Evaluation of Questions
- Third Round of Evaluation of Questions
- …
- Act on the findings

Implementing Delphi Estimation Technique

- Choosing a Facilitator. The initial step of the Delphi Method is to picking your facilitator. It is suggested to do...
- Identifying Your Experts. The success of your Delphi method depends upon a board of specialists that are...
- Defining the Problem. Next step revolves around defining what issues you are looking to comprehend via Delphi Method.
- Ask first round of Questions. In the first round, you need to pose general questions to pick up an expansive...

COCOMO – I Model

- Constructive Cost Model.
- Propose by Barry Boehm in 1981.
- Good for software engineering started using OOD (Object Oriented Development), component technology, reusable strategy and automated tool for code generation, testing etc.
- COCOMO model uses exponential components in estimation as the size of the software increases, the development team size increases. Increasing system development overhead namely communication, configuration management, integration etc.

COCOMO – I Model …

- COCOMO considers software product attributes, development team attributes and project management attributes and weigh them suitably to improve the estimation.
- The latest version of COCOMO is COCOMO-II, released in 1997. The model used 161 data points and uses Bayesian Statistical Analysis of Empirical Data of completed project and expert opinion.
- Regression model based on LOC i.e. number of Lines of Code.
- Procedural Cost Estimate Model for Software Projects.

COCOMO – I Model …

- Used as the process of reliably predicting the various parameters associated with making a project like
  - Size
  - Effort
  - Cost

COCOMO – I Model …

  - Time
  - Quality
- COCOMO is one of the most generally used software estimation models
- COCOMO predicts the efforts and schedule of a software product based on the size of the software

Types of Projects in COCOMO – I Model

- Organic
- Semidetached
- Embedded

Organic Projects in COCOMO – I Model

- The project deals with developing a well-understood application program
- The size of the development team is reasonably small
- Team members are experienced in developing similar types of projects
- Examples are
  - Business System
  - Inventory Management System
  - Data Processing System

Semidetached Projects in COCOMO – I Model

- The development consists of a mixture of experienced and inexperienced staff.
- Team members may have limited experience in related systems but may be unfamiliar with some aspects of the order being developed.
- Example are
  - Developing a new Operating System (OS)
  - Database Management System (DBMS)
  - Complex Inventory Management System

Embedded Projects in COCOMO – I Model

- The software being developed is strongly coupled to complex hardware
- Stringent regulations on the operational method exist
- Examples are
  - ATM
  - Air Traffic Control System

Projects in COCOMO – I Model

- Boehm provides a different set of expression to predict effort (in a unit of person month)and development time from the size of estimation in KLOC(Kilo Line of code) efforts estimation takes into account the productivity loss due to
  - Holidays
  - Weekly off
  - Refreshment breaks etc.

Cost Factors in COCOMO – I Model

- The cost factors are divided into four categories. They are
  - Product
  - Hardware
  - Personnel
  - Project

Attributes of Cost Factors in COCOMO – I Model

- Product b_1
  - Required software reliability extent
  - Size of the application database
  - The complexity of the product
- Hardware i.e. b_2
  - Run-time performance constraints
  - Memory constraints
  - The volatility of the virtual machine environment
  - Required turnabout time

Attributes of Cost Factors in COCOMO – I Model …

- Personnel i.e. a_1
  - Analyst capability
  - Software engineering capability
  - Applications experience
  - Virtual machine experience
  - Programming language experience
- Project i.e. a_2
  - Use of software tools
  - Application of software engineering methods
  - Required development schedule

Attributes of Cost Factors in COCOMO – I Model …

Product Attributes						
	Very Low	Low	Normal	High	Very High	Extra High
RELY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65

Attributes of Cost Factors in COCOMO – I Model …

Hardware Attributes						
	Very Low	Low	Normal	High	Very High	Extra High
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT/ VOL		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	

Attributes of Cost Factors in COCOMO – I Model …

Personal Attributes						
	Very Low	Low	Normal	High	Very High	Extra High
ACAP	1.46	1.19	1.00	0.86	0.71	
AEXP	1.29	1.13	1.00	0.91	0.82	
PCAP	1.42	1.17	1.00	0.86	0.70	
VEXP	1.21	1.10	1.00	0.90		
LEXP	1.14	1.07	1.00	0.95		

Attributes of Cost Factors in COCOMO – I Model …

Project Attributes						
	Very Low	Low	Normal	High	Very High	Extra High
MODP	1.24	1.10	1.00	0.91	0.82	
TOOL	1.24	1.10	1.00	0.91	0.83	
SECD	1.24	1.10	1.00	1.04	1.10	

COCOMO – I Model

- Boehm says that the , software cost estimation should be done through three stages
  - Basic Model
  - Intermediate Model
  - Detailed Model

Basic Model

- The basic COCOMO model provide an accurate size of the project parameters.
- The expressions for the basic COCOMO estimation model are
  - **Effort**=a_1*(**KLOC**) *a_2 Man Month Effort (or Person Month)
  - T_{dev}=b_1*(Effort)*b_2 **Months**
- Where
  - KLOC is the estimated size of the software product indicate in Kilo Lines of Code,
  - a_1,a_2,b_1,b_2 are constants for each group of software products

Basic Model …

- T_{dev} is the estimated time to develop the software, expressed in months
- Effort is the total effort required to develop the software product, expressed in Man Month Effort or Person Month (PM).

Basic Model: Estimation of Development Effort

- The expression for the estimation of the effort based on the code size are
  - Organic
    - Effort = **2.4***(KLOC)*1.05 MME or PM
  - Semi-detached
    - Effort = 3.0*(KLOC)*1.12 MME or PM
  - Embedded
    - Effort = 3.6*(KLOC)*1.20 MME or PM

Basic Model: Estimation of Development Time

- The expression for the estimation of the development time based on the effort are
  - Organic
    - T_{dev} = 2.5*(Effort)*0.38 Months
  - Semi-detached
    - T_{dev} = 2.5*(Effort)*0.35 Months
  - Embedded
    - T_{dev} = 2.5*(Effort)*0.32 Months

Basic Model: Estimation of Development Time

- Suppose a project was estimated to be 500 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded.
  - Organic
    - Effort = **2.4***(500)*1.05 MME or PM = 1260
  - Organic
    - T_{dev} = 2.5*(1260)*0.38 Months= 1197

Working of COCOMO – I Model

- Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
- Determine a set of 15 multiplying factors from various attributes of the project.
- Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

COCOMO – II Model

- The COCOMO-II Model uses 3 estimation models to estimate effort and cost models are as follows:-
- Application Suite Model
- Early Designed Level Model
- Post Architectural Model

Application Suite Model

- The model is used where software can be decomposed into several components and each components can be described in object point.
- The objects are screen, report and 3GL components which are easy to identify and count when the software system is split into sub-system components.
- Object point should be noted that are not object classes.

Application Suite Model …

- The candidates for the object points are screen, report and the number of 3GL modules to supplement the 4GL code.
- The objects are given object points depending on the level of complexities.
- The advantage of using object point is that they can be estimated from high level design of the software and they are only number of screen, report and 3GL modules.
- The complexity level is the judgement of the Software Engineering.

Application Suite Model …

Object Points			
Objects	**Simple**	**Complex**	**Very Complex**
● Screen	1	2	3
● Report	2	5	8
● 3GL Module	4	10	-

Application Suite Model …

- The object point counts need modification by the way of reduction and the software may use reusable components and library.
- Therefore revised object point (ROP) is given as ROP = OBJECT POINT X (100 – (% of Reuse))/100.
- For this ROP, the effort in man month is computed using the productivity constant based on the software development team's experience and capacity.

Application Suite Model …

Developers level of Experience and Maturity	**Very low**	**Low**	**Normal**	**High**	**Very High**
Productivity constant(NOP per month)	4	7	13	25	50

Application Suite Model …

- Man Month Effort (MME)=ROP/Productivity Constant.
- Example: - Let Object Point=40, Reuse(%)=10, then find ROP=?
- ROP=Object Point x (100 – (reuse (%)))/100 = 40*(100 – 10) / 100 = 40 * 90 / 100 = 36.
- Now, if the development team experience and maturity level is normal, the productivity constant is 13 and hence MME = ROP / Productivity Constant = 36 / 13 = 3 Man Months.
- The model is used to estimate the effort at the prototype level where the requirements are not clear.

Early Design Level Model

- Early Design Level Model

COCOMO – II Model

- The COCOMO-II Model uses the base equation **$MME=A*(SIZE)^B$**, Where
  - **MME** is the **M**an **M**onth **E**ffort
  - **A** is the Constant representing nominal productivity
  - **B** is the Factor integrating economy/dis-economy of scale(Also known as Scale Factor)
  - SIZE = KLOC (Thousand Lines of Codes).
- If B=1, the software does not have any impact on MME.
- If B<1, the MME have positive impact.
- If b>1, the MME have negative impact.

COCOMO – II Model …

- This model uses five factors for arriving at economics and dis-economics of scale which together concludes the B-factor.
  - Precedentness i.e. PREC
  - Flexibility i.e. FLEX
  - Risk Resolution i.e. RESL
  - Team Cohesion i.e. TEAM
  - Process Maturity i.e. PMAT

COCOMO – II Model …

Factor Code	Factor Name	Very Low	Low	Normal	High
PREC	Precedentness	6.20	4.96	3.72	2.48
FLEX	Flexibility	5.07	4.05	3.04	2.03
RESL	Risk Resolution	7.07	5.65	4.24	2.83
TEAM	Team Cohesion	5.48	4.38	3.29	2.19
PMAT	Process Maturity	7.80	6.24	4.68	3.12

COCOMO – II Model …

- **PERC i.e. Precedentness**
  - Understanding and experience of developing similar software.
  - If the degree of learning benefit which can be given to new software is very low, then rating is 6.20.
- **FLEX i.e. Flexibility**
  - Flexibility measured based on the degree of freedom and comfort level the developer has, based on the level of conformance required to either pre – established or customer-laid down standards, specifications, tools and schedules.

COCOMO – II Model …

- **RESL i.e. Risk Resolution**
  - If the organization and the software development team has considerable experience in risk management and is in a position to develop an RMMM plan for the project, then the level of risk resolution is very high and the rating value is 2.83.
- **TEAM i.e. Team Cohesion**
  - If the capacity of the organization to provide a development team whose members will work towards common objectives in cohesion is nominal, then the rating value is 3.29.

COCOMO – II Model …

- **PMAT i.e. Process Maturity**
  - This is the SEI-CMM level used for describing and organization’s development maturity.
  - If the CMM level is very low i.e. ‘1’, then the rating value is 7.80.

COCOMO – II Model …

- Let us assume that in a given situation, the organization’s level on these five factors is very low, then B = 0.91 + 0.01 * (6.20 + 5.07 + 7.07 + 5.48 + 7.80) = 0.91 + 0.01* 31.62 = 0.91 + 0.3162 =1.2262
- A=13, size based on FPA is 10 KLOC
- Then MME=13*(10)$^{1.2262}$ = 220 Man Month
- In case of the organization’s level on these five factors is high, then
- B = 0.91 + 0.01*(2.48 + 2.03 + 2.83 + 2.19 + 3.12) = 0.91 + 0.01*12.65 = 0.91 + 0.1265 = 1.0365 = 1.04
- Then MME=13*(10)$^{1.04}$ =142 Man Months

Post Architectural Model

- There are some other factors which are relevant as they have larger impact on MME.
- If we consider these factors, then we calculate MME which is modified value.
- There are 16 factors which are significant.
- They are
    - RELY
    - DATA
    - CPLX

Post Architectural Model …

- RUSE
- DOCU
- TIME
- STOR
- PVOL
- ACAP
- PCAP
- PCON
- AEXP

Post Architectural Model …

- PEXP
- LTEX
- TOOL
- SITE

- COCOMO-II equation for MME (Modified) = MME * (product of ratings of 16 factors).

Post Architectural Model …

Factor	Factor Name	Levels and Factors			
		Very Low	Low	Nominal	High
Product Factor					
RELY	Software Reliability	0.82	0.92	1.00	1.10
DATA	Database Size	0.80	0.90	1.00	1.14
CPLX	Software Complexity	0.73	0.87	1.00	1.17
RUSE	Required Reusability	0.85	0.95	1.00	1.07
DOCU	Documentation	0.81	0.91	1.00	1.11

Post Architectural Model …

Factor	Factor Name	Levels and Factors			
		Very Low	Low	Nominal	High
Platform Factors					
TIME	Time Constraint on Execution	NRA	NRA	1.00	1.11
STOR	Main Storage Constraint	NRA	NRA	1.00	1.05
PVOL	Platform Volatility	NRA	NRA	1.00	1.15

Post Architectural Model …

Factor	Factor Name	Levels and Factors			
		Very Low	Low	Nominal	High
Personnel Factor					
ACAP	Analyst Capability	1.42	1.19	1.00	0.85
PCAP	Programmer Capability	134	1.15	1.00	0.88
AEXP	Analyst Experience	1.22	1.10	1.00	0.88
PEXP	Programmer Experience	1.19	1.09	1.00	0.91
LTEX	Language and Tools Experience	1.20	1.09	1.00	0.91
PCON	Personnel Continuity	1.29	1.12	1.00	0.90

Post Architectural Model …

Factor	Factor Name	Levels and Factors			
		Very Low	Low	Nominal	High
Project Factor					
TOOL	Use of Software Tool	1.17	1.09	1.00	0.90
SITE	Site Environment	1.22	1.09	1.00	093

Post Architectural Model …

- **RELY i.e. Software Reliability**
  - Failure does not cause any inconvenience, rating = very low.
  - Failure is fatal, rating is high.
- **DATA i.e. Database Size**
  - Database size is measured as D/P, where D: database in bytes, P: lines of codes.
  - If D/P<10, Rating is very low, if D/P>1000, rating is high.

Post Architectural Model …

- **CPLX i.e. Software Complexity**
  - Few control options, simple, few calculations, simple data management, then rating is very low.
  - If all these are high, then rating is high.
- **RUSE i.e. Required Reusability**
  - No requirements of reusability i.e. customized software, then the rating is very low.
  - If the reusability is substantial, then the rating is high.

Post Architectural Model …

- **DOCU i.e. Documentation**
  - Documentation need is standard but low, then rating is very low.
  - If documentation need is very high both pre and post development, then rating is high.
- **TIME i.e. Time Constraint on Execution**
  - No constraint, then rating is very low.
  - Available time is almost equal to the execution time, then rating is high.

Post Architectural Model …

- **STOR i.e. Main Storage Constraint**
  - No constraint due to available very high storage, then rating is very low.
  - Available storage is almost equal to required storage, then rating is high.
- **PVOL i.e. Platform Volatility**
  - If platform is stable, then rating is very low.
  - If platform is unstable and may change rapidly, then the rating is high.

Post Architectural Model …

- **ACAP i.e. Analyst Capability**
  - In experience and lack of knowledge, then the rating is very low.
  - If analyst scores high on experience then the rating is high.
- **PCAP i.e. Programmer Capability**
  - In experience and lack of knowledge, then the rating is very low.
  - If analyst scores high on experience then the rating is high.

Post Architectural Model …

- **PCON i.e. Personnel Continuity**
  - Very low turnover, then the rating is high. 50% or more would leave, then the rating is very low.
- **AEXP i.e. Analyst Experience**
  - Minimum Experience, then the rating is very low.
  - More than adequate experience then the rating is high.

Post Architectural Model …

- **PEXP i.e. Programmer Experience**
  - Minimum Experience, then the rating is very low.
  - More than adequate experience then the rating is high.
- **LTEX i.e. Language and Tools Experience**
  - Minimum Experience, then the rating is very low.
  - More than adequate experience then the rating is high.
- **TOOL i.e. Use of Software Tools**
  - No use or occasional use, then the rating is very low.
  - Sustained use of variety of tools, then the rating is high.

Post Architectural Model …

- **SITE i.e. Site Environment**
    - Single site, single location, not more than one or two sponsors, then the rating is very low.
    - Multiple sites, multiple partners, number of locations but supported by good communication infrastructure, then the rating is high.
    - If problem of communication are not there, rating is very low.

Post Architectural Model …

- Now considering the case where all the ratings are very low, then
    - Product of all 16 ratings = 0.82* 0.80* 0.73* 0.85* 0.81* 1.42* 1.34*1.22*1.19*1.20*1.29*1.17*1.22=2.01
    - MME=220*2.01=442 Man Months
- Now considering the case where all the ratings are high, then
    - Product of all 16 ratings = 1.10* 1.14*1.17*1.07*1.11*1.11* 1.05*1.15*0.85*0.88*0.88*0.91*0.91*0.90*0.90*0.93=0.98
    - MME=142*0.98=139 Man Months

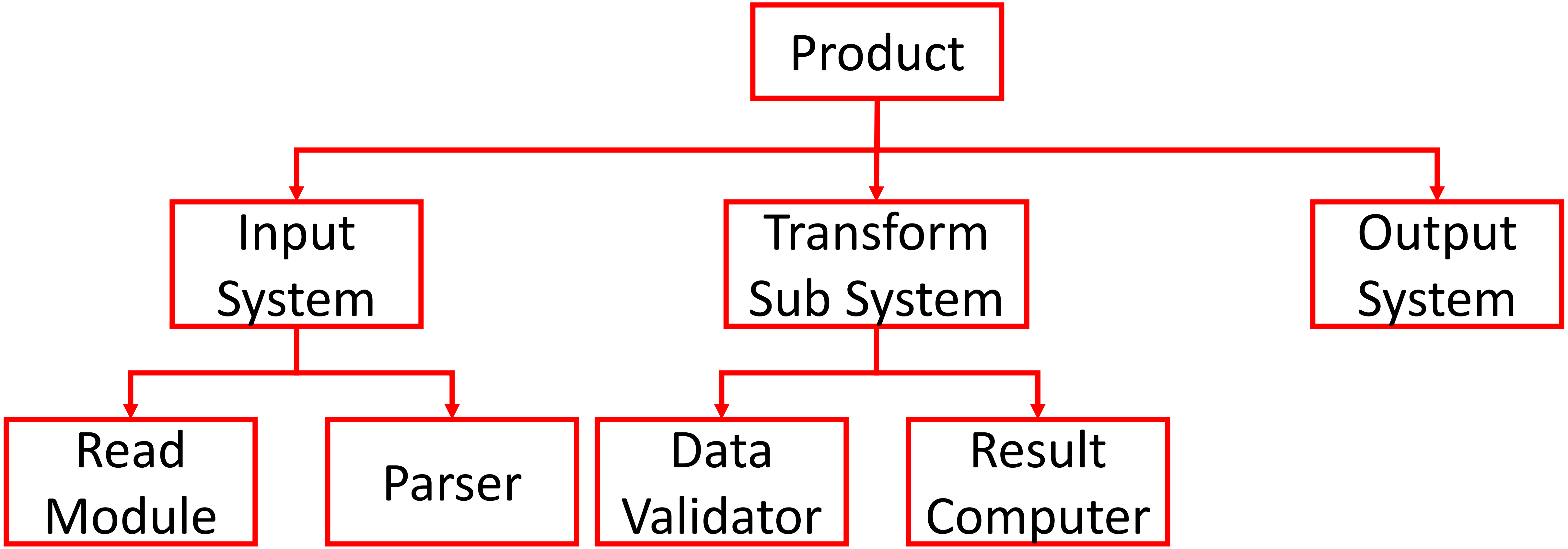
Work Breakdown Structure

- Expert judgement and group consensus are tow down techniques.
- The work breakdown structure is bottom up approach.
- A work breakdown structure is a hierarchical char that accounts for an individual part of the system.
- A work breakdown structure chart indicates product hierarchy and process hierarchy.

Product Hierarchy

- It identifies the product component and indicates the manner in which the components are interconnected.
- A work breakdown chart of process hierarchy identifies the work activity and relationship among those activities using the techniques.
- Costs are estimated by assigning cost to each individual component in the chart and then adding the cost.

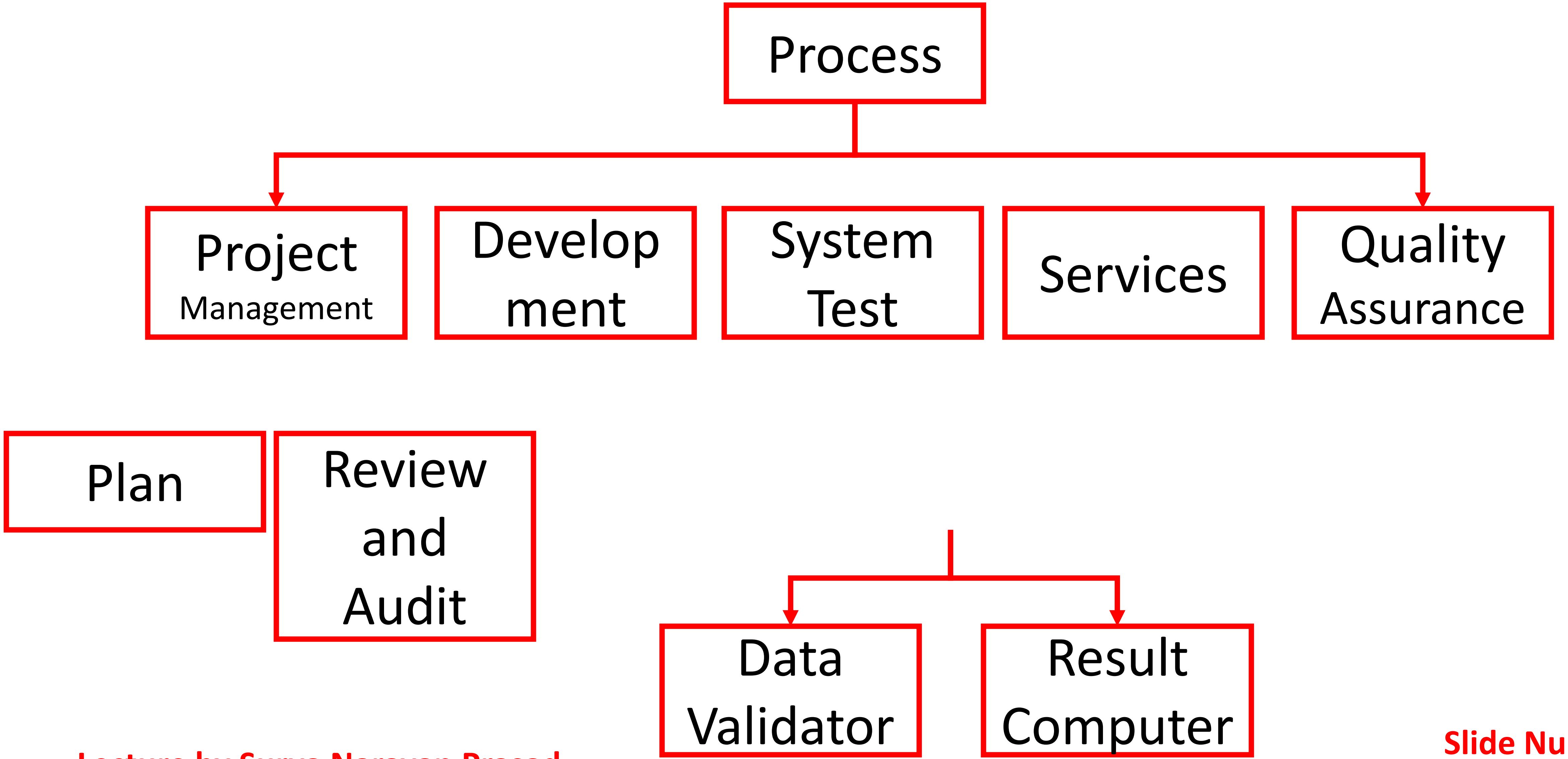
Product Hierarchy …



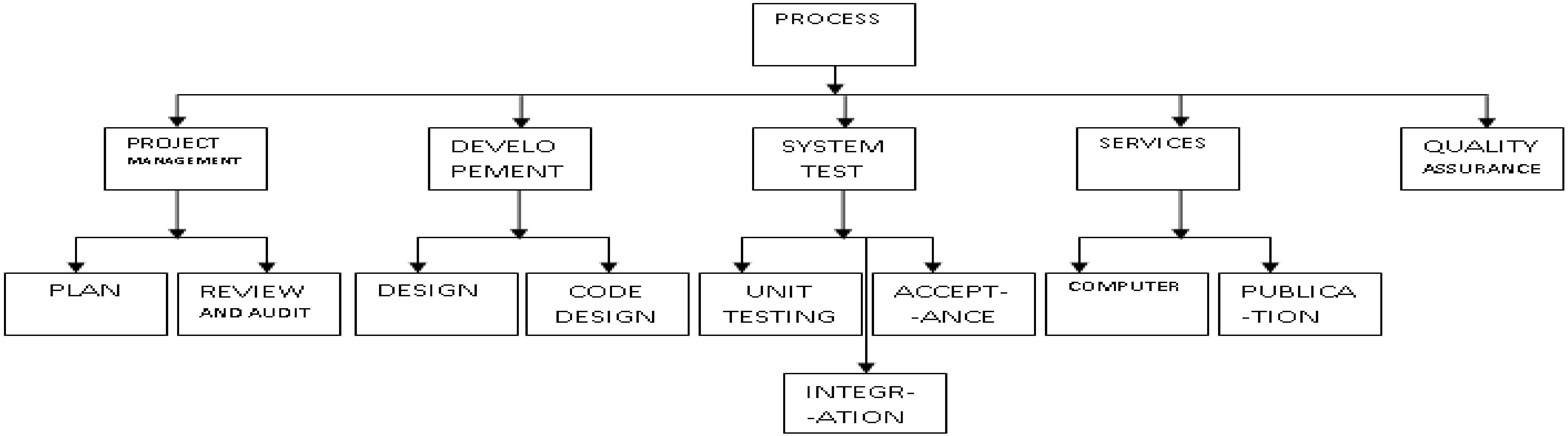
Process Hierarchy

- Processes can be broken down into components like project management, development, system test, services and quality assurance, which can be further broken down into sub components. Some planners uses both product and project hierarchy of work breakdown structure chart for cost estimation.
- The primary advantage of this technique is, it identifies the account for various process and product factors and in making explicit exact cost which are included in the estimates.

Process Hierarchy …



Process Hierarchy …



Cost Estimation

- Human Resource Cost
  - Skills
  - Knowledge
  - Numbers
- Development Resource Cost
  - Platform
  - Software
  - Tools

Cost Estimation …

- Personal Cost
- Hardware Cost
- Software Cost
- Training Cost
- Marketing Cost
- Outsourcing Cost

Planning the Development Process

- The first consideration is to define a product life cycle model.
- The software life cycle includes all activities required to define, develop, test, deliver, operate and maintain a software product.
- The Life cycle model includes
  - Phased Model
  - Cost Model
  - Prototype Model.

Phased Life Cycle Model

- In phased life cycle model segment, the software life cycle is a series of successive activities.
- Each phase requires well defined input information, neutralizes well defined processes and result in well defined product.
- Planning and requirement definition are two major activities of analysis phase.
- They include understanding the customer's problem by performing the feasibility study, developing a recommended study, developing the acceptance criteria and planning the development process.

Phased Life Cycle Model …

- Requirement definition is concerned with identifying the basic function of the software component. In a hardware / software people system, emphasis is placed on what the software will exactly do and what are its constraints. The product requirement definition is the specification that describe the processing environment, the required software function, performance constraint of the software i.e. size, speed, machine, configuration etc., exception handling, subsets and implementation priorities, probable changes and light modification and the acceptance criteria for the software.

Phased Life Cycle Model …

- In phase model, the software design follows analysis. Architectural design includes identifying the software components, decoupling and decomposing them into processing modules and conceptual data structure and specifying the interconnection among the components. Detail design involves adaptation of existing code, modification of standard algorithm, invention of new algorithms, design of data representation and packaging of the new software products.

Phased Life Cycle Model …

- The implementation phase involves transaction of design specification into source code and debugging, documentation and unit testing of source code. Errors discovered during the implementation test includes the errors in the data interfaces between routines, logical errors in the algorithm, errors in the data structure layout and failures to account for various processing cases.

Phased Life Cycle Model …

- In addition, source code may contain requirement errors that indicate the failure to capture the customer needs, in the requirement document, design error that reflect failure to translate requirement into correct design specification and an implementation errors that reflect the failure to correctly translate design specification into source code. One of the primary goal of phased approval to software development is to eliminate requirement and design errors from an evolving software product before implementation begins.

System Testing

- System Testing involve two types of testing. They are
  - Integration
  - Acceptance

Documentation

- Documentation is an important component of software.
- It can be a paper or electronic document. It could be part of software and available on line.
- It could be delivered separately and available offline.
- A complex software needs normally following documentation: -
  - System Manual
  - User Manual
  - System Maintenance Manual
  - Operations Manual

System Manual

- It provides information about system scope, design, architecture, system flow, technology details, interfaces used in the system etc.
- It is also backed by the requirement analysis and modeling, specification, important functions and features that the system provides.
- The source code is a part of system manual, although in general it is not given to the customer.
- Only executables of the entire system is delivered, demonstrated and implemented.

User Manual

- System user manual is an instruction for the users of the system.
- It provides screen by screen, interface by interface, file by file usage instructions and its impact elsewhere.
- It is also initially used for training as well as guiding users of the system.

System Maintenance Manual

- It deals with the routine system maintenance.
- It is used by the system administrator and coordinator to cater the problems like user problems, system problems, maintenance of files, databases, backups, security issues, system logs etc.

Operations Manual

- It deals with system operations as it functions.
- It provides guidelines to users to understand the implications of any action for the system.
- It provides transparency and insight as to how the system operates or responds to the action taken by the user.

Planning Software Project

- Defining the Problem
- Planning is a primary stage of the development.
- Lack of planning is the cause of schedule slippage, cost over-runs, poor quality and a high maintenance cost for the software.
- Careful planning is required for both development process and the work product in order to avoid these problem.
- Develop a definitive statement of the problem to be solved, which includes the description of the present situation, problem constraints and the statement of the goal to be achieved.

Planning Software Project …

- The problem statement should we trace in customer terminology.
- Justify a computerized strategy solution for the problem. Not the customer problem from the customer point of view. For example Inventory Problem, Payroll Problem etc. and not as the problem of sorting algorithm and relational database.
- Identify the functions to be provided by and the constraints on the hardware system, the software sub system and the people sub system.

Planning Software Project …

- The interaction among sub system must be established and development and operational constraints must be determined by each subsequent sub system.
- The function to be performed by each major sub system must be identified.

Goals and Requirement for Planning Software Project

- Determine system level goals and requirement for the development process and the work product given a consize statement of the problem and an indication of the constraint that exist for a solution, preliminary goals and requirement can be formulated.
- Goals are target to be achieved.
- Goals are applied to both the development process and the work product.
- Goals can either be qualitative or quantitative.

Goals and Requirement for Planning Software Project …

- High level goals and requirement can often be explained in terms of quality attribute that the system should process.
- The quality attributes are as follows:-
  - Portability
  - Reliability
  - Efficiency
  - Accuracy
  - Errors
  - Robustness
  - Correctness

Qualitative Goal Process

- The development process should enhance the professional skills of the quality and assurance personnel.

Quantitative Goal Process

- The system should be delivered on time.
- The system should reduce the cost of the transaction.