Introduction to Software Engineering

Syllabus

Unit	Contents	Lectures
1.	Introduction to Software Engineering • Introduction to software • Qualities of good software • Introduction to software engineering • Components of software engineering • Software development models • Comparative analysis of process models • Software Development Life Cycle	5

What is Computer?

- Computer is defined as a programmable electronic device, devised for performing and controlling operations that can be expressed either in logical or numerical form.

Components of Computer System

- Hardware
- Software
- Firmware
- Liveware(Humanware)

Software

- Software is the set of instructions to acquire inputs and to manipulate them to produce the desired output in terms of functions and performance as determined by the user of the software.
- It also includes a set of documents such as the **software manual**, meant for users to understand the software system.
- A software is described by its capabilities.
- The capabilities relate to the functions it executes, the features it provides and the facilities it offers.

Software …

- The software is developed keeping in mind certain hardware and operating system considerations known as platform.

Components of Software

- Software is bundle product
  - The Software Code(Source) Intellectual Property
  - The Executables
  - The Design Documents [Intellectual Property]
  - The Operation and System Manual
  - Installation and Implementation Manual

Classes of Software

- Software is classified into two classes.
- They are
  - Generic
  - Customized

Generic Software

- Generic software is designed for a broad customer market whose requirements are very common, fairly stable and well understood by the software engineer.
- A generic product is managed by the developer.
- The requirements and specifications in a generic product are controlled by the developer.

Customized Software

- Customized software are those that are developed for a customer where domain, environment and requirements being unique to that customer cannot be satisfied by the generic products.
- A customized product is managed by the customer.
- The requirements and specifications in a customized product are controlled by the customer and influenced by the practices in the industry.

Qualities of Good Software

- Software can have attributes which together will decide whether it is good or bad.
- The definition of good software changes with the person who evaluates it.
- The software is required by the customer, used by multiple end users in the organization and developed by the software engineer.
- The user considers the usability and reliability of the software as the most important criteria.

Qualities of Good Software …

- The software engineer looks for the maintainability and efficiency of the software.
- Hence bare minimum requirement for a software to be good are
  - Maintainability
  - Dependability
  - Efficiency
  - Usability

Introduction to Software Engineering

- Software and its development is becoming day by day a complex task and having several dimension which require careful consideration.
- If these considerations are not addressed properly, the software product is
  - Unreliable and is of doubtful quality
  - Delivered late to the customer
  - Non maintainable under changing circumstances
  - Inefficient in resource management
  - Very expensive

Reasons of Fallout

- Lack of understanding of customer requirement by both the customer and the developer.
- Non use of structured methodology to gather customer requirements and its analysis to arrive at the Software Requirement Specification (SRS).
- Poor estimation of resource, effort and cost.
- Non use of developmental and testing methodology to ensure the quality of the software.

Reasons of Fallout …

- Hence we can conclude that software development is a costly and risk proposition if the development process does not follow …
  - Systematic
  - Scientific Approach

Why Software Developing is Software Engineering?

- An Engineering Approach calls for
  - Requirement / Need Analysis
  - Conceptualization of the Product with Specification
  - Product Design
  - Process of Manufacturing / Development i.e. Coding of Solution
  - Testing
  - Quality Assurance
  - Installation and Implementation

Why Software Developing is Software Engineering?

- Software Engineering is defined as a discipline that addresses following aspects of software development.
  - Economic aspects are
    - Cost
    - Benefit
    - Return on Investment

Why Software Developing is Software Engineering?

- Design aspects are
  - Ease of Development
  - Ensuring Delivery of Customer Requirements
- Maintenance aspects are
  - Ease of Effecting Changes
  - Modifications

Why Software Developing is Software Engineering?

- Implementation aspects are
  - Ease of Installation
  - Demonstration and Implementation of Software by the customer and the user

Why Software Developing is Software Engineering?

- Software Development activity is considered as engineering discipline because it is
  - Systematic
  - Scientific
  - Methodical
  - Uses standard models and algorithms in design and development

Scope of Software Engineering

- Requirements study and definition for determining software scope
- Requirements analysis to generate **S**oftware **R**equirement **S**pecifications
- **S**oftware **D**esign to deliver various components and specifications
- **S**oftware **D**evelopment
- Integration
- Testing
- Implementation

Scope of Software Engineering

- Maintenance of software
  - Corrective
  - Enhancement
  - Adaptive

Definition of Software Engineering

- IEEE(Institute of Electrical and Electronic Engineers) defines Software Engineering as the application of
  - Systematic
  - Disciplined
  - Quantifiable Approach
  - Development Operations
  - Implementation of the software
  - Deployment Operations
  - Maintenance of the software

Definition of Software Engineering

- Software Engineering focuses on **quality** and **develops processes** to **deliver the quality** using **tools and methodologies**.
- Software Engineering deals with **(development) processes** to **ensure delivery of the software** through
  - Management Control of Development Process
  - Production of Requirement Analysis Models
  - Data Models
  - Process Models
  - Reports

Definition of Software Engineering

- Software Documentation
- Software Engineering Technology is applied for software development through engineering methods.

Activities Involved in Software Engineering

- Software Engineering executes following set of mandatory activities for good software development.
- They are
  - Requirement Analysis and Definition
  - Software scoping and determination of its boundaries
  - Factorizing software in different components and configurations for development, testing and integration
  - Planning of Software Development Process, scheduling, executing, monitoring and control of software development

Activities Involved in Software Engineering

- Testing at all stages and phases for quality assurance as required by the customer
- Documenting the software for its users
- Software Scoping and Determination of its Boundaries

- Factorizing Software in different components and configurations for
  - Development
  - Testing
  - Integration

Activities Involved in Software Engineering

- Planning, Scheduling, Executing, Monitoring and Control of Software Development
- Testing at all stages and phases for quality assurance as required by the customer
- Documenting the software for its users
- Implementation through demonstration, installation and executions at the customer site
- Change management in pre and post implementation phase

Activities Involved in Software Engineering

- The managerial activities that contribute to the efficiency and effectiveness of the software are
- Resource and effort estimation and its management
- Risk assessment and its management
- Development process management to remain within cost, time and budget
- Project management for achieving software goals

Prerequisite Skills

- In order to perform the above activities, the software engineer should have following basic skills
- Analysis and modeling abilities
- Conceptualize a software solution and support it by design
- Software development skills using tools, techniques and technologies
- Strategies skills to evolve development, testing and implementation strategies

Prerequisite Skills …

- In addition to the above basic skills the software engineering effort is productive when the software engineer has
  - Domain knowledge of the software system
  - Project Management Skills
  - Higher level communication skills and leadership qualities
  - Essential technology and language skills

Prerequisite Skills …

- Software Engineering is a methodology based on scientific and engineering principles which when applied, helps to achieve software solution as specified by the customer.
- It facilitates with effectiveness the development of software, with the quality duly validated in the development process.
- The production of good software faces following challenges
- Integrating the new software in legacy systems which are stabilized and run on an old platform and technology.

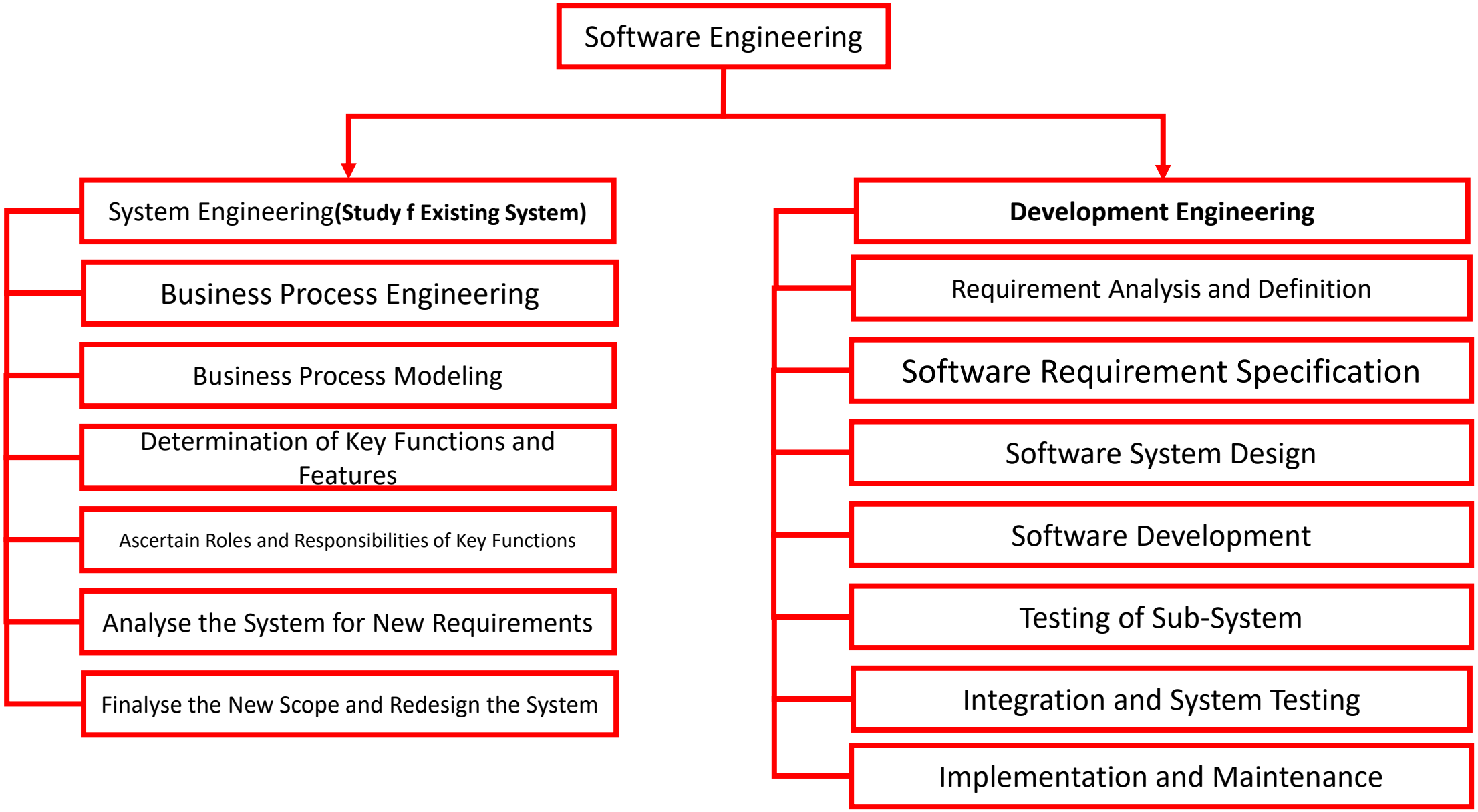
Prerequisite Skills …

- Integrating without disturbance to current business operations and not at extra cost.
- Ensuring that heterogeneous system developed over a period of time work together effectively with new software implementation
- Handling issues of data migration, system interfacing, inter operability, up gradation of technology and so on without loss of system integrity.
- Need to continuously upgrade that is a result of continuous improvement in technology, having a direct bearing on the cost and efficiency of the system and software.

Components of Software Engineering

- Software Engineering Approach has two components.
- They are
  - Systems Engineering Approach
  - Development Engineering Approach

Approaches of Software Engineering …



How Quality and Characteristics of Software is defined ?

- The software and its quality depends upon the system in which the software is installed.
- The characteristics of the system have a lot of bearing on the software scope, design and quality.
- A software engineer first understands the system in which the software is to be run.

System Engineering Methods

- The system has broad meaning.
- It includes
  - The Business Systems
  - Computer Software used in the business system
  - The understanding of the system is done through the System Study and Analysis.
  - The System Study and Analysis is done using System Engineering Methodology(SEM)

System Engineering Methods …

- The SEM steps are
  - Define the objective of the system
  - Define the boundaries of the system
  - Factorize the system into different components for understanding the system functions and features.
  - Understand the relationships between various components.
  - Define the relationship in terms of input, output and processes.
  - Understand and ascertain the role of hardware and software.

System Engineering Methods …

- Understand and ascertain the role of hardware and software
- Understand the role of the databases and any other software products used in the system.
- Identify key operational and functional requirements of the system to be addressed.
- Model the system for analysis and development through modeling software.
- Discuss the systems with the customer, the users and the stakeholders affected by the system

Goals of Development Engineering Methodology

- Development Engineering Methodology has the goal of translating the system requirements as software system goal and proceeds to achieve it through following steps.
  - Requirement definition and specifications
  - Design solution to deliver the requirement
  - **Determine the architecture** for delivery of the solution
  - Software development planning
  - Software testing by components
  - Integration of system components

Goals of Development Engineering Methodology

- Integration testing for confirmation and conformance of delivery of requirements
- Determination of implementation strategy
- Implementation
- Change management process
- Maintenance of installed product

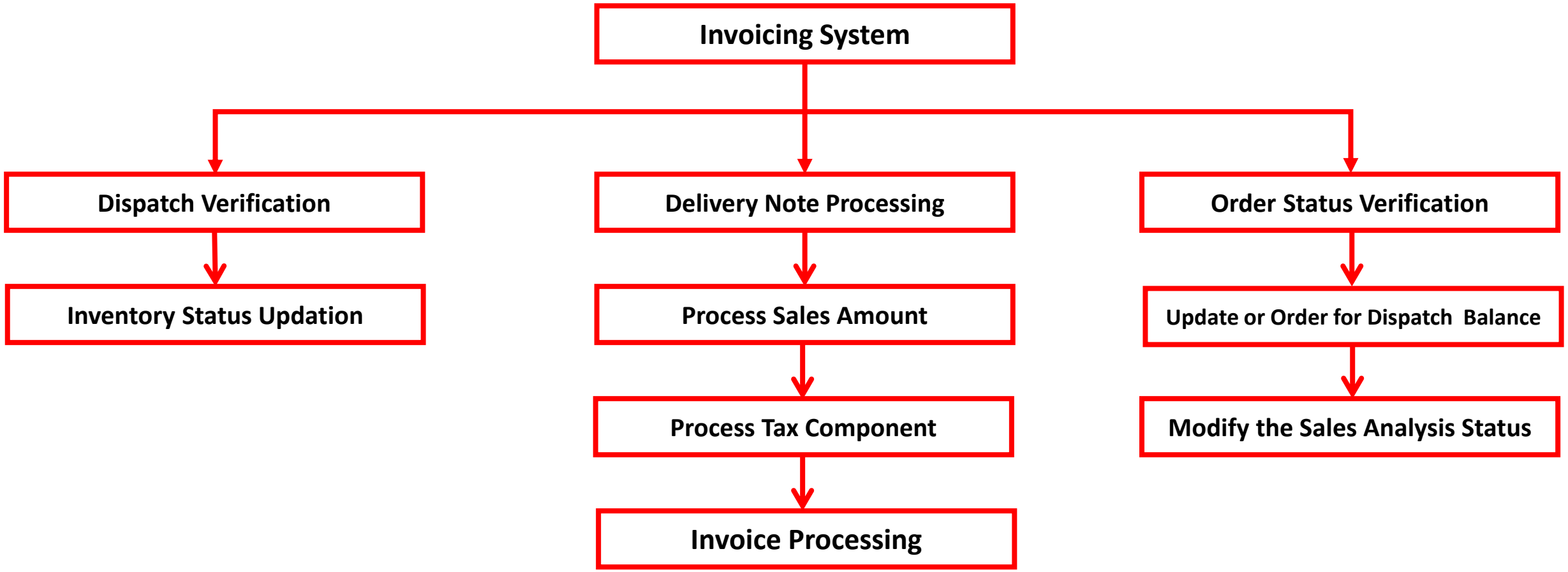
Approaches of Software Engineering

- Software Engineering is practiced through two approaches. They are
  - Structured Systems Analysis and Design(SSAD)
  - Object Oriented Systems Analysis and Design(OOSAD)
- The SSAD approach was introduced by Yourdon and DeMarco.
- In SSAD, the system and its requirements are decomposed in structured manner.

Approaches of Software Engineering

- Considering the **Case of Invoicing (Billing) System**
  - It will be decomposed into sub systems by functions
  - Software development is carried out using the sub system structure, tested, integrated and implemented.
  - The skill of the software engineer lies in decomposing the system in a structured way that allows for understanding and developing a software with the required characteristics.
  - In SAD, the focus is on functions and data structure designed for those functions.
  - Skill lies in the decomposition of system.

Approaches of Software Engineering



Approaches of Software Engineering

- The OOSAD development approach was proposed by Boach, Coad, Yourdon, Rumbaugh and Jackobson.
- In OOSAD system, an analysis of domain and organization business is done and object of models independent of system under consideration is developed.
- The object could represent a function, process or a document evolved for the organization.
- Each object has the attributes to describe, methods to perform and relationships to other objects.

Approaches of Software Engineering

- The OOSAD principle is that when an object is developed ,it is available for use in current, proposed and futuristic systems.
- In OOSAD, objects and processing methods (systems) are decoupled from the data. Skill lies in the modeling the organization and business in the objects.

Approaches of Software Engineering

Object	Invoice	Order
Attributes	ID NO. Address A/C No. Amount	ID NO. Address A/C No. Amount
Methods	Compute Value of Goods Compute Disocunt Compute Ad. Charge Compute Invoice Amount	Compute Order Value of Goods Compute Disocunt Compute Ad. Charge Compute Net Order Value

Approaches of Software Engineering

- Like system engineering, development engineering also has two approaches.
- Each approach has a process unique to itself based on certain terms and conditions.
- The development engineering follows following models
  - Waterfall Model
  - Evolutionary(Iterative) Model

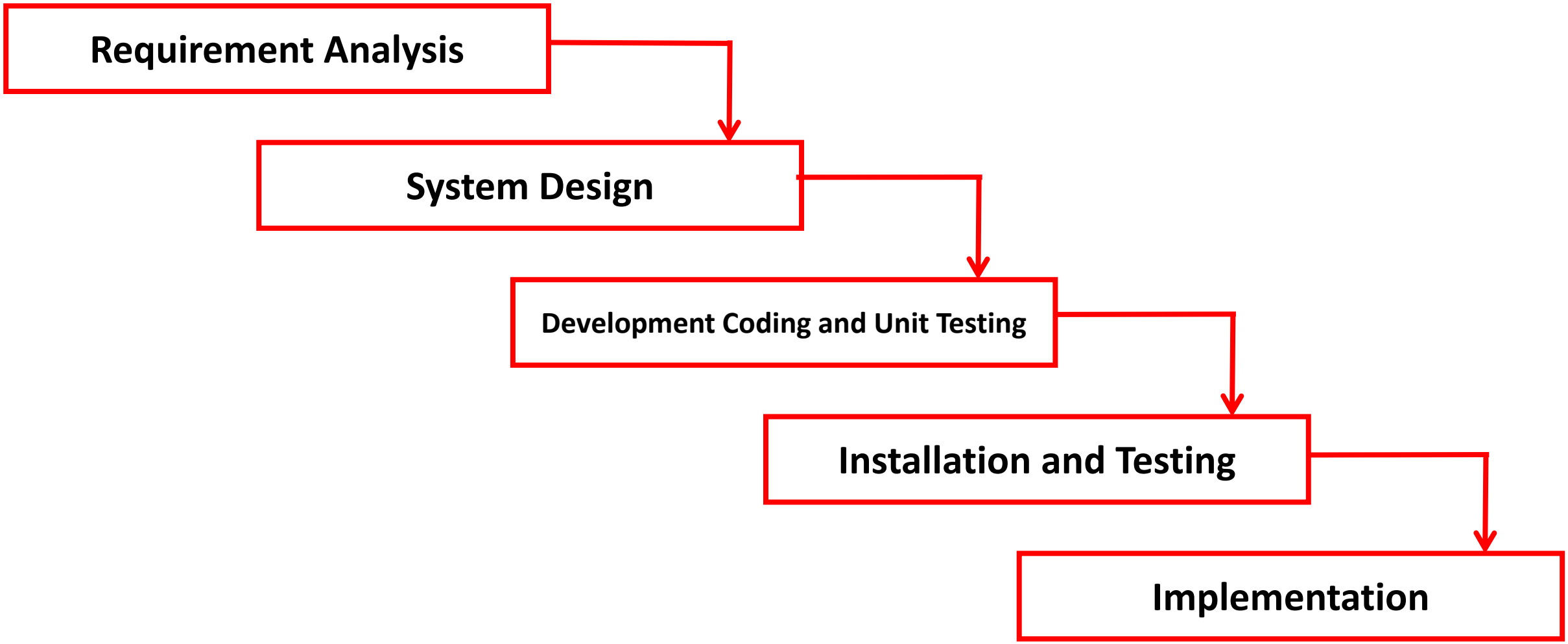
Waterfall Model

- The development engineering model generally followed is the “WATERFALL MODEL”, when following conditions prevail.
- Requirement analysis and definition is **stable** and unlikely to change in a significant way during development and in the post implementation period.
- Software system design is not likely to undergo a change due to changes in technology, platform and language considered in the system.
- The system is so well understood by users.

Waterfall Model …

- The Changes Are Not Expected During The Operations And Maintenance.
- It means, where the requirements are easy to ascertain and establish and are stable, the development is customer specific and changes are not foreseen in the near future, development of software is done using “WATERFALL MODEL”.

Waterfall Model …



Evolutionary Model

- When requirements are difficult to establish in clear terms, the “WATERFALL MODEL” becomes ineffective.
- Once software requirements are not clear, software system design and development become very costly operations.
- Such system scenarios are prevalent due to uncertain business conditions, user not being clear about how the systems should run and behave in different conditions and there being no unanimous views among all users about the scope, functions and features of the system.

Evolutionary Model …

- When the software engineer is confronted with above mentioned scenario, development of software is carried out through “EVOLUTION MODEL”, development engineering effort is made first to establish correct, precise requirement definitions and system scope, as agreed by all the users across the organization.
- This is achieved through the application of iterative processes to evolve a system most suited under the given circumstances.

Evolutionary Model …

- The process is termed as “ITERATIVE” as the software engineer goes through a repetitive process of
  - Requirement Analysis
  - Design
  - Testing through Prototype
  - Implementation
  - Assessment
  - Evaluation
- till all users and stakeholders are satisfied.

Evolutionary Model …

- The requirements and scope get enhanced and improved in each iterative development process, making the system better in every respect.
- In the iterative development process, the risk of the software being unacceptable to the users is very much reduced.
- The use of technology and processing architecture.

Evolutionary Model …

- The requirements and scope get enhanced and improved in each iterative development process, making the system better in every respect.
- In the iterative development process, the risk of the software being unacceptable to the users is very much reduced.
- The use of technology and processing architecture.

Capability Maturity Model

- A bench–mark for measuring the maturity of an organization's software process
- CMM defines 5 levels of process maturity based on certain Key Process Areas (KPA)

Capability Maturity Model …

CMM Level	Characteristics	KPAs
1(Initial)	Chaotic, unrepeatable, high risk of non- performance	Key areas not defined

Capability Maturity Model …

CMM Level	Characteristics	KPAs
2(Repeatable)	Methodical in umbrella activities and processes. Performance is repeated but not improved.	Requirement analysis project planning, sub-contract and out-sourcing SQA and low level configuration management Main umbrella activities are in use

Capability Maturity Model …

CMM Level	Characteristics	KPAs
3(Defined)	Improved performance in cost, schedule, quantity and risk management.	Organizational process in place. Integrated management. Organizational focus in training, HR management, quality initiatives, supporting software development

Capability Maturity Model …

CMM Level	Characteristics	KPAs
4(Managed)	learns from project experience and continuous performance improvement in all key areas of the project.	Process management with the goal of effectiveness and efficiency

Capability Maturity Model …

CMM Level	Characteristics	KPAs
5(Optimizing)	All round performance improvement through learning. High performance in all quality attributes and risk management through **RMMM**.	Process choice linked to project characteristics, change management, verification, testing and maintainability are the focal points. Goal is to optimize the use of the resources.

Capability Maturity Model …

- Level 5 – Optimizing (< 1%)
- Level 4 – Managed (< 5%)
- Level 3 – Defined (< 10%)
- Level 2 – Repeatable (~ 15%)
- Level 1 – Initial (~ 70%)

Capability Maturity Model …

- Level 5
  - Optimizing (< 1%)
  - Process change management
  - Technology change management
  - Defect prevention

Capability Maturity Model …

- Level 4
  - Managed (< 5%)
  - software quality management
  - quantitative process management

Capability Maturity Model …

- Level 3
  - Defined (< 10%)
  - Peer reviews
  - Intergroup coordination
  - Software product engineering
  - Integrated software management
  - Training program
  - Organization process definition
  - Organization process focus

Capability Maturity Model …

- Level 2
  - Repeatable (~ 15%)
  - Software configuration management
  - Software quality assurance
  - Software project tracking and oversight
  - Software project planning
  - Requirements management
- Level 1
  - Initial (~ 70%)

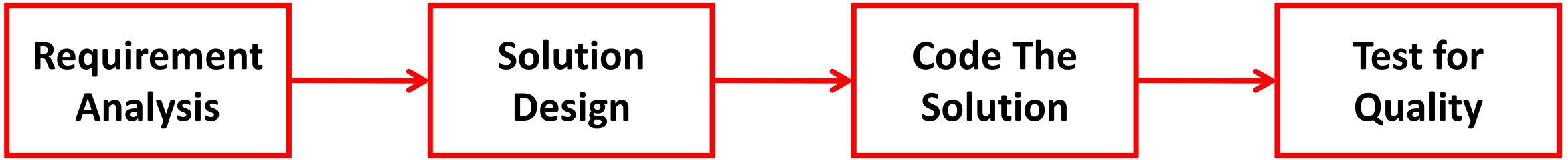
Software Process Model

- Irrespective of which level of CMM the organization has, the software engineer has following five software process models.
  - LSM (Linear Sequential Model)
  - PRM (Prototype Model)
  - RAD (Rapid Application Development Model)
  - INM (Incremental Model)
  - BSM (Boehm Spiral Model)

Linear Sequential Model

- The LSM is also called as classic life cycle model.
- It is similar to waterfall model.
- In this model, the flow of activities moves ahead and if each activity is carried out properly, the chances of reverting to an earlier activity or activities are very remote.
- In a system environment where there is clarity and unanimity on the requirements and specifications.

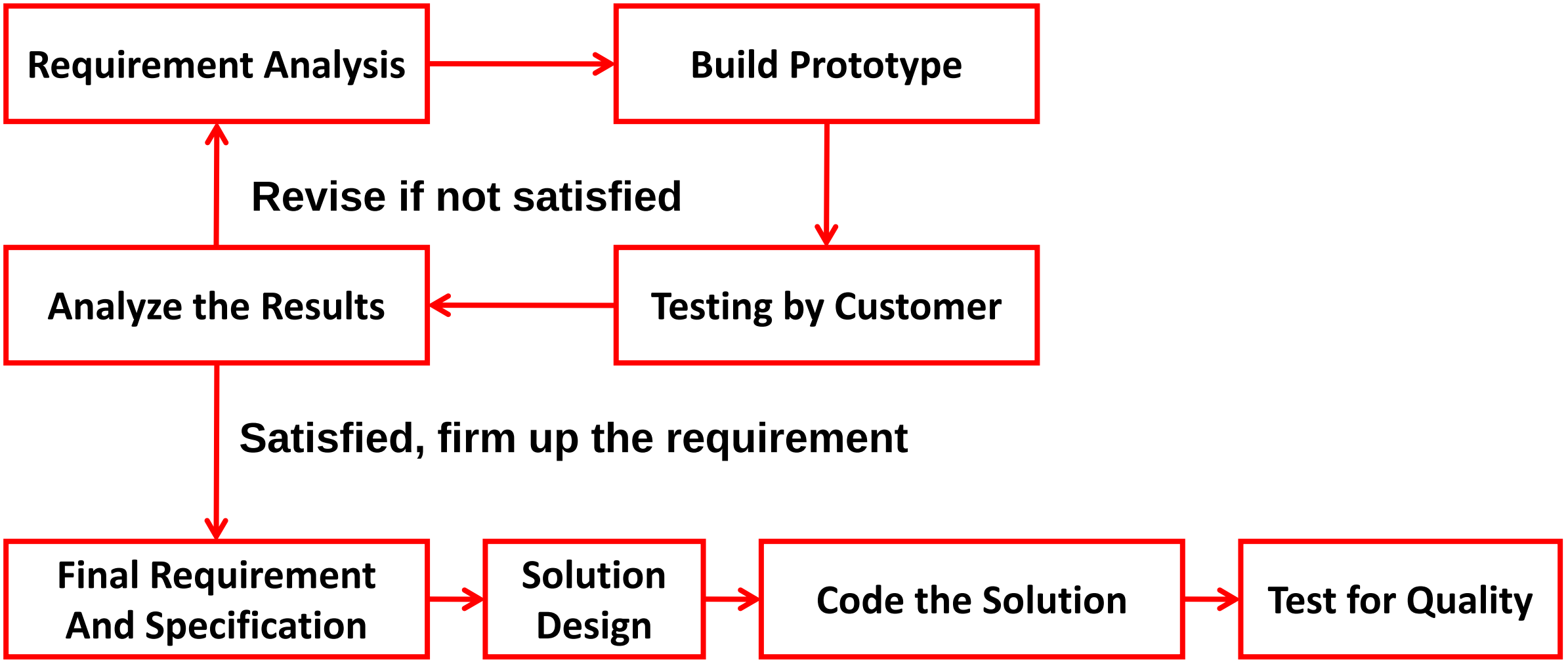
Linear Sequential Model …



Prototype Model

- This model is used where the users, customers and stakeholders are not sure about their requirement. Some times the software engineer is not sure what kind of design solution would be most suitable and appropriate.
- In such cases, the prototype model is built to ascertain the requirement specifications, solution and appropriate technology.

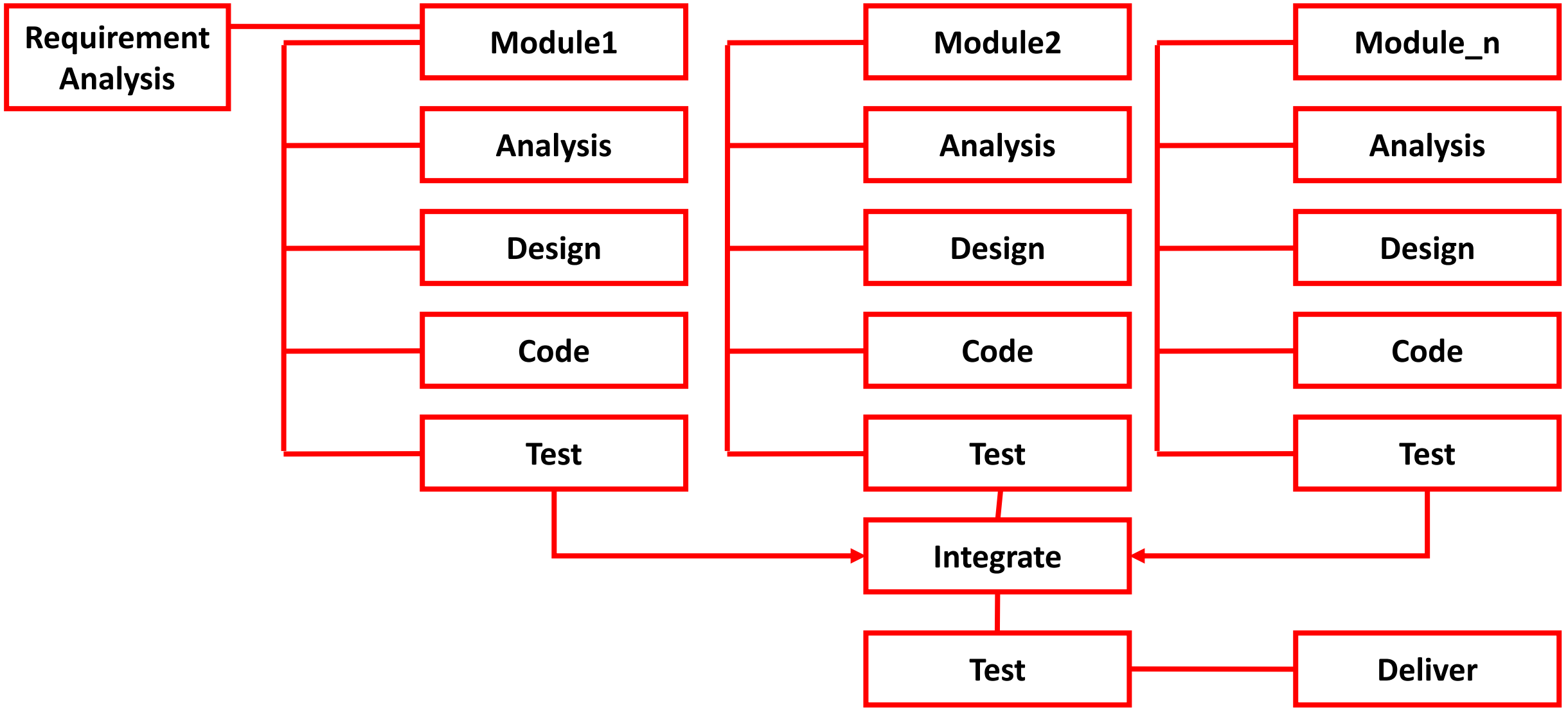
Prototype Model …



Rapid Application Development Model

- The RAD model is proposed when requirements and solutions can be modularized as independent system or software component, each of which can be developed by different teams.
- RAD is also recommended when system components have already been developed by the organization in the context of other software systems and these can be used with minor or no change.
- Due to the reusability of the system components and the possibility the requirement into smaller components that can then be assigned to different teams, it is called RAD.

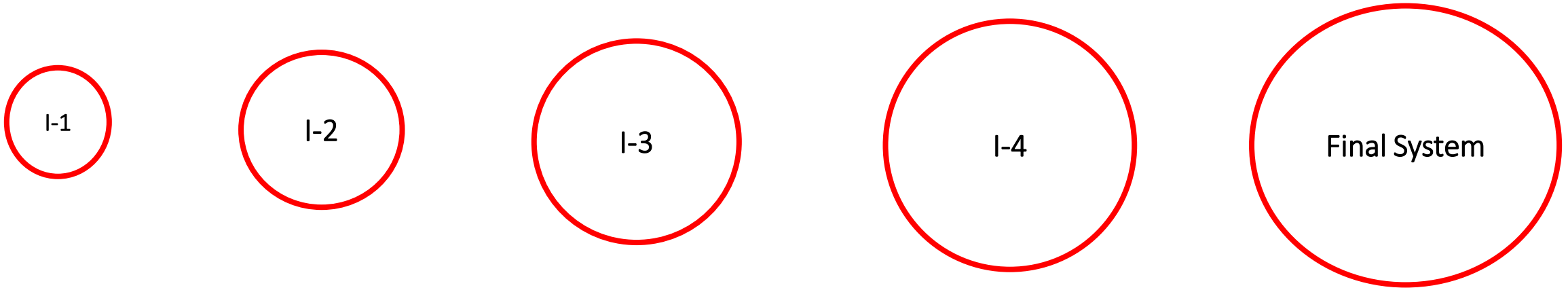
Rapid Application Development Model …

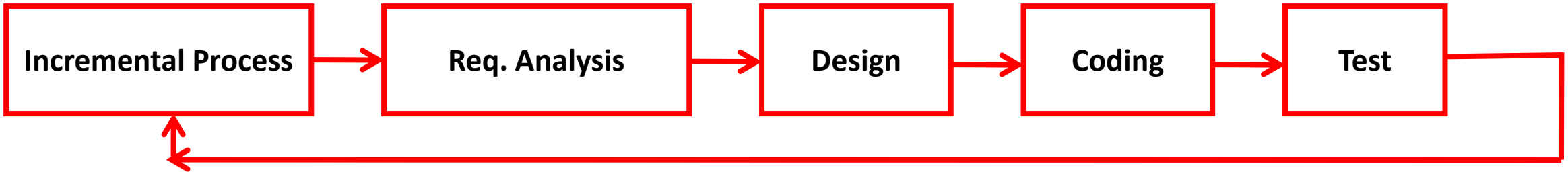


Incremental Model

- Some times it has been observed that the nature of the requirement is complex and its visualization in precise terms is difficult. In this situation, experimentation has to be done through trial and error to finalize the requirement analysis and specifications.
- Requirements needs to be built up in parts to avoid the risk of developing an incorrect and improper solution and hence the approach is evolutionary. The solution is evolved in parts and incremental model.

Incremental Model …





Spiral Model

- It was proposed by Boehm.
- It combines the properties of LSM,RAD and INM.
- The spiral model is recommended where the requirements and solution call for full fledged, large and complex system combined with features and facilities from scratch.
- It is used when the technology and skills are new and the user is not able to state the requirements clearly.

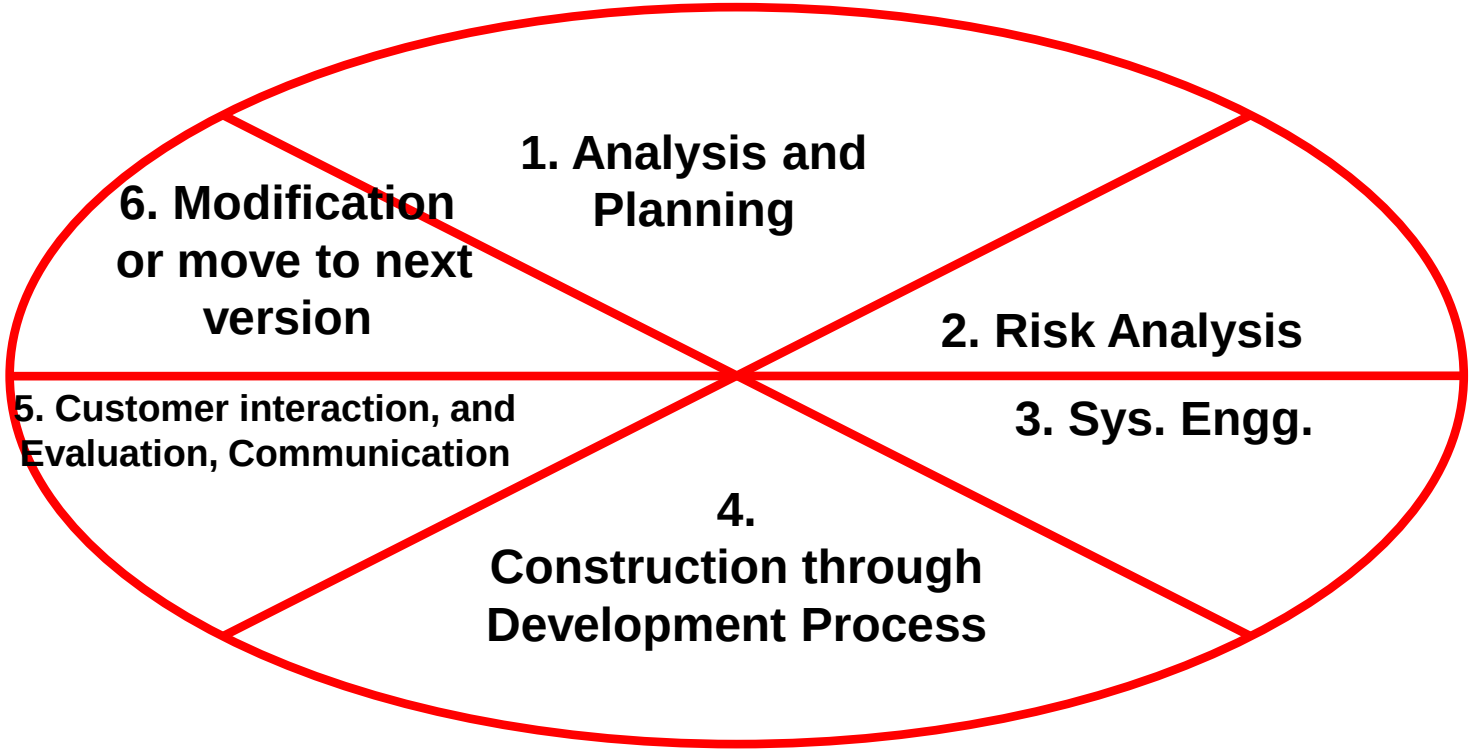
Spiral Model …

- When the requirement is not clear and the proposed solution has multi user, multi function, multi features and multi location application to be used on multiple platform where seamless integration, interfacing , data migration and replication are the major issues.
- Boehm's model proposes a process model and above which core process, the software engineer in continuous interaction with the user/customer/stakeholder through a well defined model of Analysis, Design, Demonstration, Evaluation, Revision and Deliver the development activity is done.

Spiral Model …

- The entire scope is divided into versions of the system and all versions are in place when the complete system solution is delivered.
- The main focus of the Boehm's model is risk analysis, communication with the customer and finding system parameters through customer/user/stakeholder participation in the entire process.

Spiral Model …



Comparative Analysis of Process Models

Model	Nature of Requirement	Customer Interaction	Risk Level	Technology	Domain Knowledge	Focus
LSM	Simple, universal	Initial, once or twice	Zero	Proven and universally understood	Common	Solution
PRM	Simple, needs testing and conformation	Couple of times till prototype is approved	Very low	Proven but needs testing	Technology	Prototype and solution
RAD	Simple , breakable for distributed development and ease of integration	Initially couple of times to ascertain the scope	Low	Team management, proven technology	Generally known	Development methodology and architecture

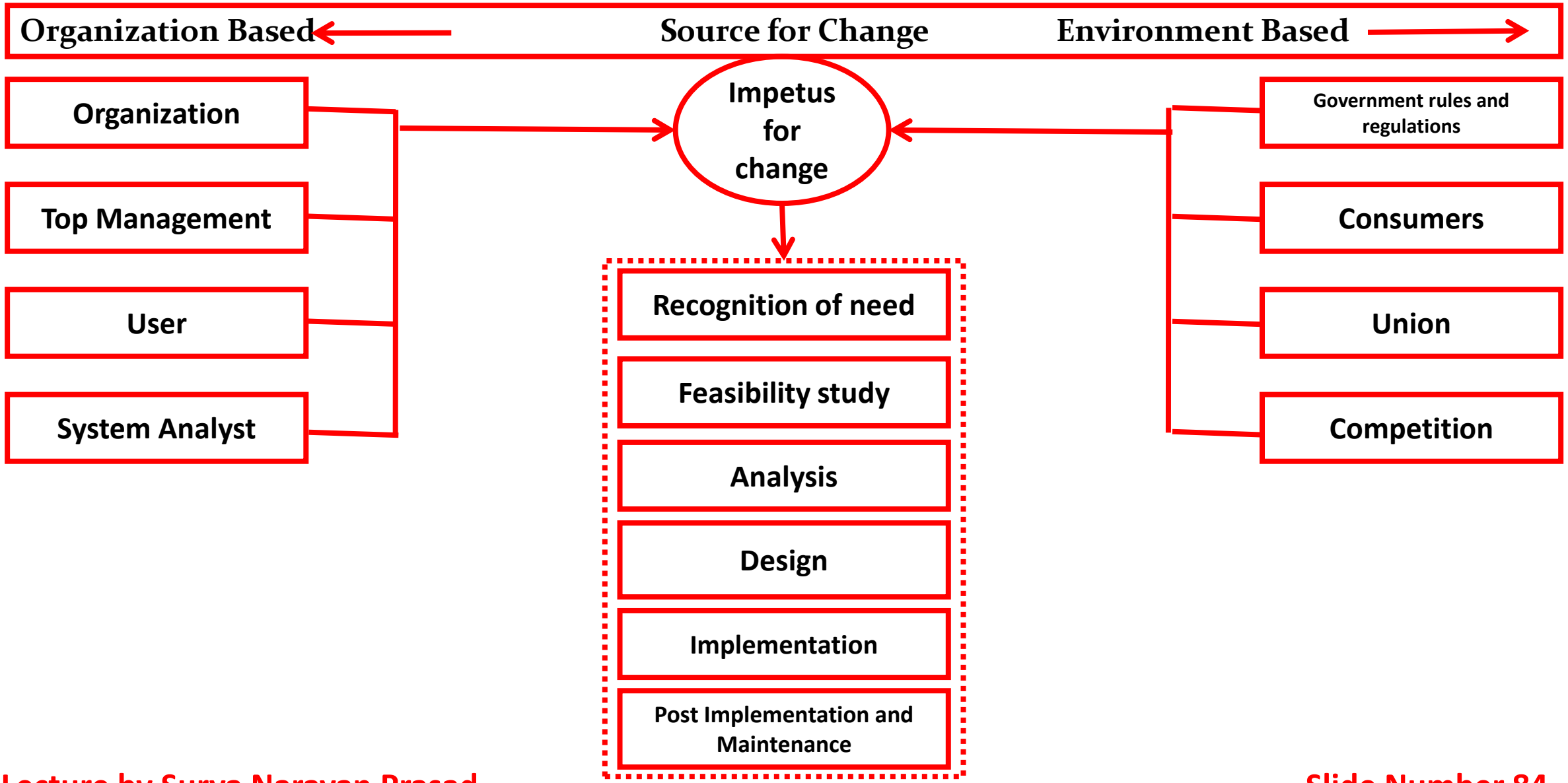
Comparative Analysis of Process Models …

Model	Nature of Requirement	Customer Interaction	Risk Level	Technology	Domain Knowledge	Focus
INM	Calls progressive introduction of functions, features and implementation	Very frequent, continuous basis for stage wise confirmation	Medium	Configuration management	High	Solution implementation strategy
BSM	Complex, large customer specifies, domain influences key processes and technology to be tested	Continues to move together	High	Integration, handling of multiple technologies and solutions	Very high	Tools, technology, solution, implementation, customer participation

Software Development Life Cycle

- A framework that describes the activities performed at each stage of a software development project.

Software Development Life Cycle …



Software Development Life Cycle …

Stage	Key Question	Result
Recognition of Need Preliminary Survey/ Initial Investigation	What is the problem or opportunity?	Statement of scope and objectives. Performance criteria
Feasibility Study Evaluation of existing system and procedures. Analysis of alternative candidate system. Cost estimation	What are the user's demonstrable needs? Is the problem worth solving? How can the problem be redefined?	Technical/ Behavioral feasibility. Cost/Benefit analysis. System scope and objectives. Statement of new scope and objectives
Analysis Detailed evaluation of present system Data collection	What must be done to solve the problem? What are the facts?	Logical model of system i.e. data dictionary, data flow diagram

Software Development Life Cycle …

Stage	Key Question	Result
Design General Design Specifications Detailed Design Specifications Output Input Files Procedures Program Construction Testing Unit Testing Combined Module Testing User Acceptance Testing	In general, how must the problem be solved? Specifically , how must the problem be solved? What is the system (processing) flow? Does the user approve the system? How well do individual program/modules test out? How ready are programs for acceptance test?	Design of alternative solutions. Final cost benefit analysis. Hardware specifications. Cost estimation. Implementation specifications. Implementation schedule. Approval of the system by the user. Program test plans. Security, audit and operating procedures. Actual hardware use. Formal system test.

Software Development Life Cycle …

Stage	Key Question	Result
Implementation User Training File/System conversion	What is the actual operation? Are user manuals ready? Are there delays in loading files?	Training Programme. User friendly documentation.
Post Implementation and Maintenance Evaluation Maintenance Enhancement	Is the key system running? Should the system be modified?	User requirements met. User standards met. satisfied user.

Software Development Life Cycle …

Stage	Key Question	Result
Implementation User Training File/System conversion	What is the actual operation? Are user manuals ready? Are there delays in loading files?	Training Programme. User friendly documentation.
Post Implementation and Maintenance Evaluation Maintenance Enhancement	Is the key system running? Should the system be modified?	User requirements met. User standards met. satisfied user.

Software Development Life Cycle …

Stage	Key Question	Result
Implementation User Training File/System conversion	What is the actual operation? Are user manuals ready? Are there delays in loading files?	Training Programme. User friendly documentation.
Post Implementation and Maintenance Evaluation Maintenance Enhancement	Is the key system running? Should the system be modified?	User requirements met. User standards met. satisfied user.