Software Engineering
Unit - 1
(Lecture Notes)

Prepared by

Jay Nanavati, Assistant Professor, SEMCOM

Topics

- Program vs. Software
- Software
- Software Engineering (Definition & Objective)
- Phases in Software Development

Program vs. Software

Program	Software
▪ Small in size (LOC)	Large in size (KLOC)
▪ Limited Functionality	Wide range of functionality
▪ Simple or less complex	Complex
▪ Used by the author of the program	Used by people other than developers
▪ Little or no documentation	Sufficient documentation
▪ Bugs can be tolerated	Bugs can not be tolerated
▪ Testing is often not done.	Testing is always done.
▪ Portability, reliability & usability are not so important.	Portability, reliability & usability are also important.

Software

- IEEE Definition

  “A software is a collection of

  1. Programs
  2. Procedures
  3. Rules
  4. Documentation and
  5. Data “

Software Engineering

- IEEE Definition

  “Software engineering is
  the systematic approach to the
  Development,
  Operation,
  Maintenance and
  Retirement of
  Software.”

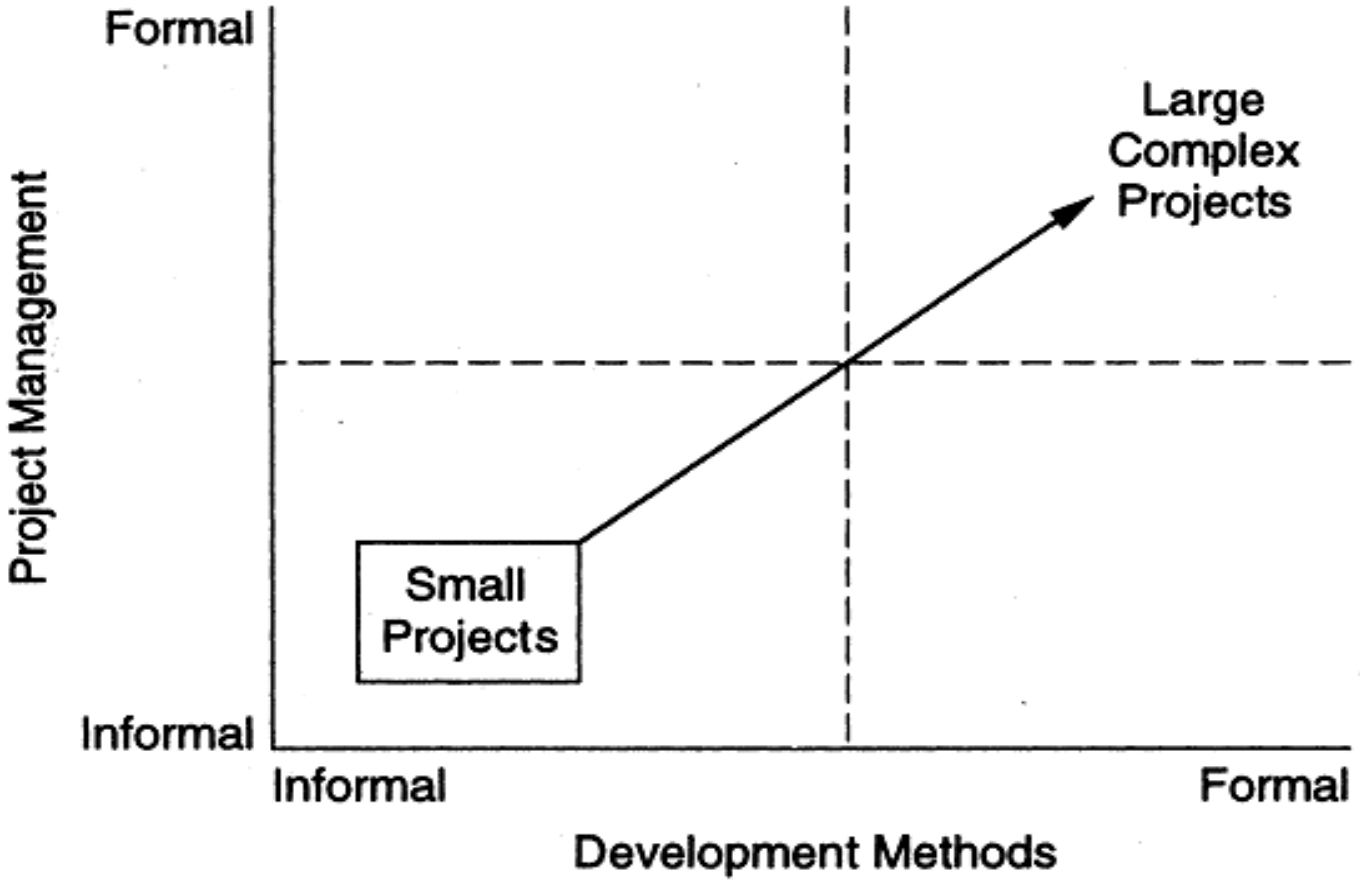
The Software Engineering Problem

- Scale
- Cost, Schedule and Quality
- Consistency

The Problem of Scale

- An illustration of issue of scale is counting the number of people in a room vs. taking a census
  - ✓Both are counting problems
  - ✓Methods used in first not useful for census.
  - ✓For large scale counting problem, must use different techniques and models.
  - ✓Management will become critical.

The Problem of Scale (conti.)



The Problem of Cost, Schedule & Quality

- Cost

  The cost of developing a software includes cost of

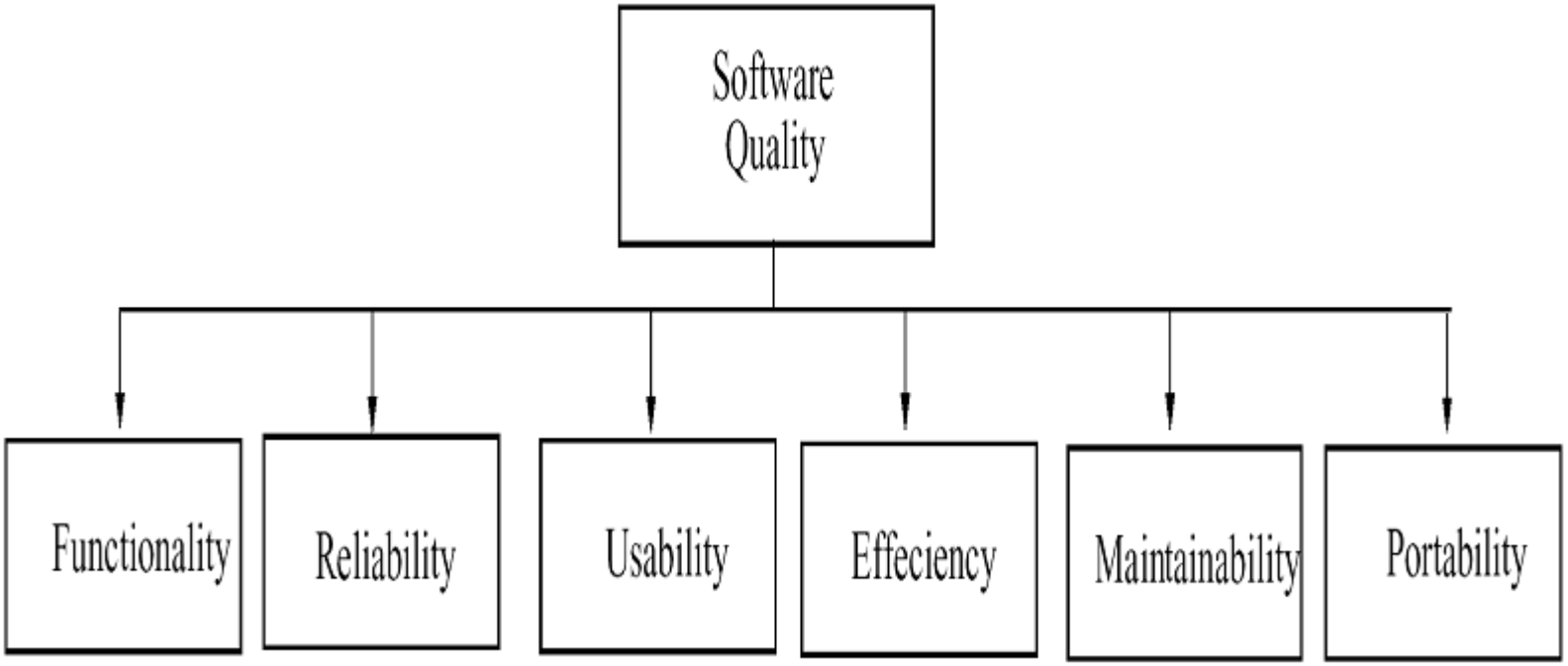
  - ✓ manpower (DLOC per person-month)
  - ✓ hardware
  - ✓ software and
  - ✓ other support resources.

The Problem of Cost, Schedule & Quality (conti.)

- Schedule
  - ✓ is an important factor.
  - ✓ Short (or reduced) cycle-time is highly desired. (even if the cost becomes higher)
  - ✓ RAD

The Problem of Cost, Schedule & Quality (conti.)

- Quality



The Problem of Consistency

- Not just once or
- sometime / many-a-times but
- ALWAYS

Phases in Software Development

- Software development process consists of various phases.
- Why phases?
  1. It breaks a large problem into small parts.
  2. A phased process allows proper checking for quality and progress at some ‘defined’ points during the development.
- Four phases:
  1. Requirement Analysis
  2. Software Design
  3. Coding
  4. Testing

Requirements Analysis

- Requirements analysis is done to "understand" the problem the software is going to solve.
- The emphasis is on "what" is needed from the system, not *"how"* the system will achieve it.
- In Problem Domain
- Two major activities in this phase:
  1. Understanding the problem
  2. Requirement specification

Requirements Analysis (contd.)

- The Analyst
- Input (Requirements gathered through various techniques)
- Output (**SRS Document** or the requirements document)
- The SRS document must specify
  - ✓ all functional & performance requirements
  - ✓ the formats of inputs & outputs and
  - ✓ all design constraints

Software Design

- Purpose is to plan a solution of the problem specified by ____ in ____ phase.
- Moving from problem domain to solution domain.
- Design takes us towards “how” to satisfy the needs.
- The most critical factor affecting the quality of the software. (contributes a lot to success (or failure) of the software)

Software Design (contd.)

- The Software Architect / Developers with good experience
- Input (_____)
- Output (The design document)
- Two activities
  1. System Design (what)
  2. Detailed Design (how)

Coding

- Goal is to translate the design into code in a given programming language*.
- Affects testing & maintenance.
- The goal of coding should be to reduce the testing & maintenance efforts.
- Input (____)
- Output (____)

Testing

- A Quality Control activity.
- Goal is to uncover errors in the programs.
- Test plan
- Test case specification document
- Test report(s) / Error reports(s)

General Characteristics of Software Process

- Software Process
- Characteristics of software process
  1. Predictability
  2. Testability & Maintainability
  3. Early Defect Removal & Defect Prevention
  4. Process Improvement

Predictability

- Determines “how accurately” the outcome of a process can be predicted.
- Past experience can ONLY be used if a process is predictable.
- Process under statistical control.

Testability & Maintainability

- Testing is not a “side activity”.

Phase	% Efforts
Requirements Analysis	10
Design	20
Coding	20
Testing	**50**

Testability & Maintainability (contd.)

- Development and/or Maintenance

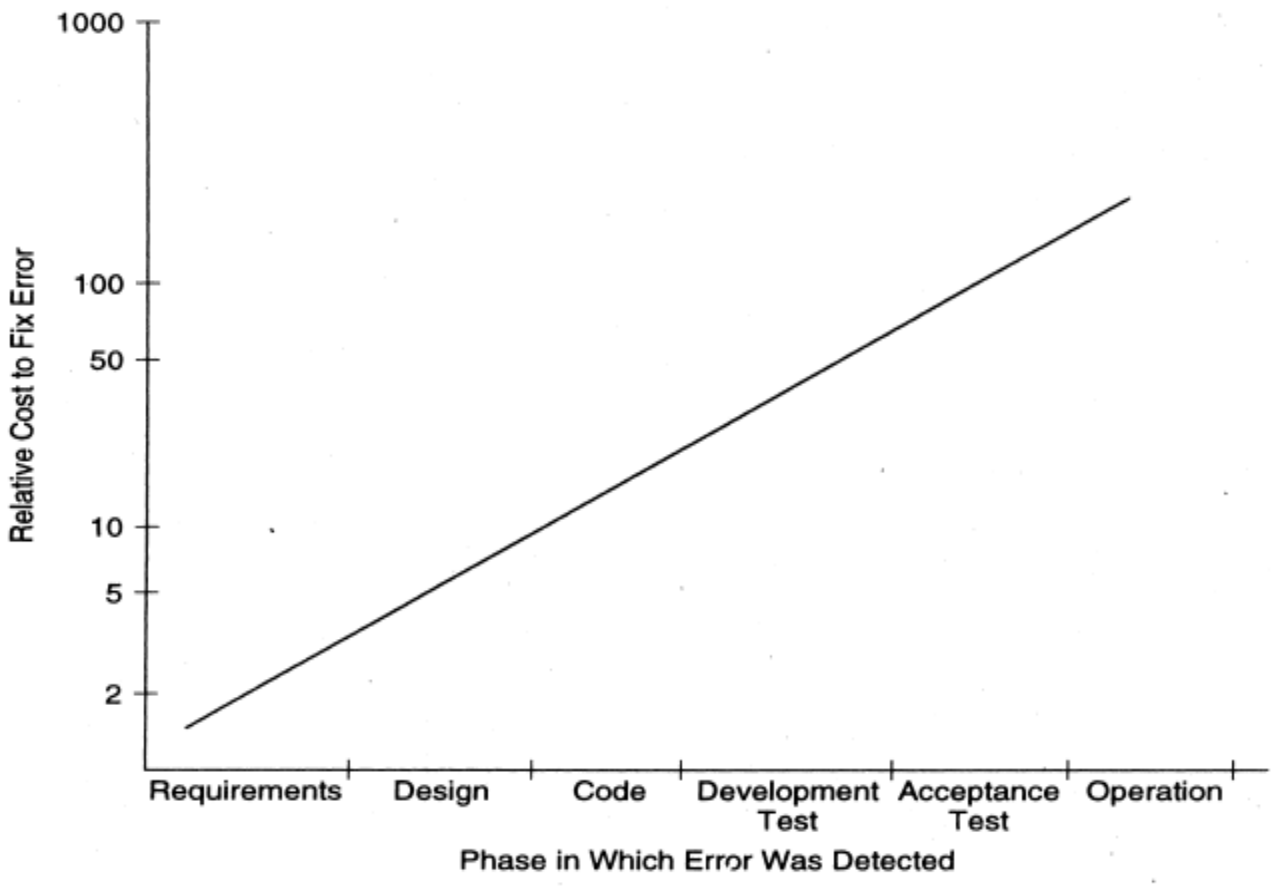
Activity	% time spent
Writing programs	**13**
Reading programs and manuals	16
Work-related Communication	32
Others	39

Early Defect Removal & Defect Prevention

- Errors can occur at any stage.

Phase	% errors
Requirements Analysis	20
Design	30 (20 +10)
Coding	50 (20 + 30)

Early Defect Removal & Defect Prevention (contd.)



Process Improvement

- Evaluate the existing process & understand the weaknesses in the process.

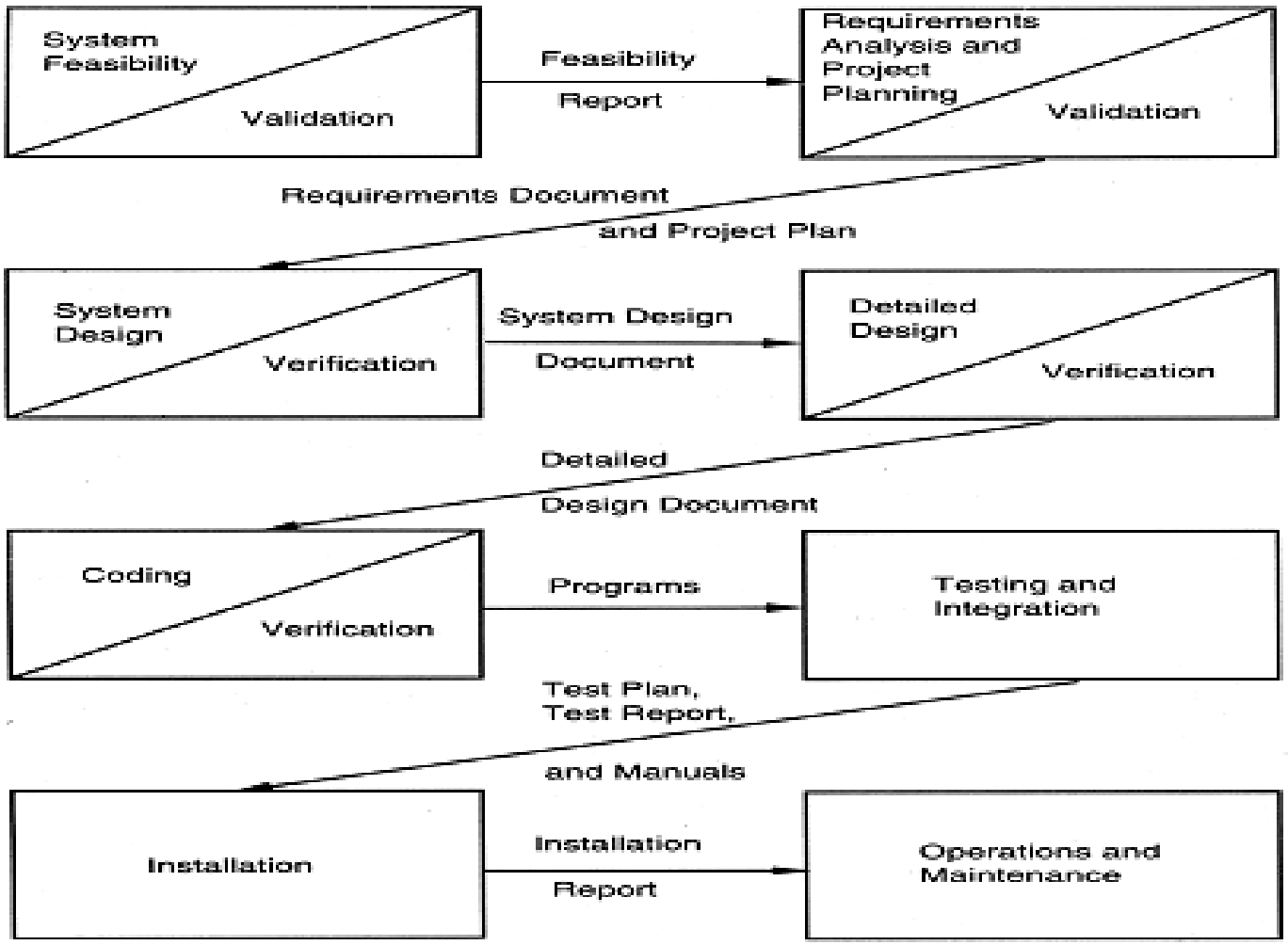
Software Process Models

- Waterfall
- Prototype
- Iterative Enhancement
- Spiral

Waterfall Model

- The simplest process model.
- Phases are organized in a linear order.
- Phases are
  1. Feasibility Analysis
  2. Requirements Analysis
  3. Design
  4. Coding
  5. Testing & Integration
  6. Installation
  7. Operations & Maintenance

Waterfall Model (contd.)



Waterfall Model (contd.)

- Project Outputs
  - ✓ Requirements Document
  - ✓ Project Plan
  - ✓ System Design Document
  - ✓ Detailed Design Document
  - ✓ Test plan & Test reports
  - ✓ Final Code
  - ✓ Software Manuals
  - ✓ Review Reports

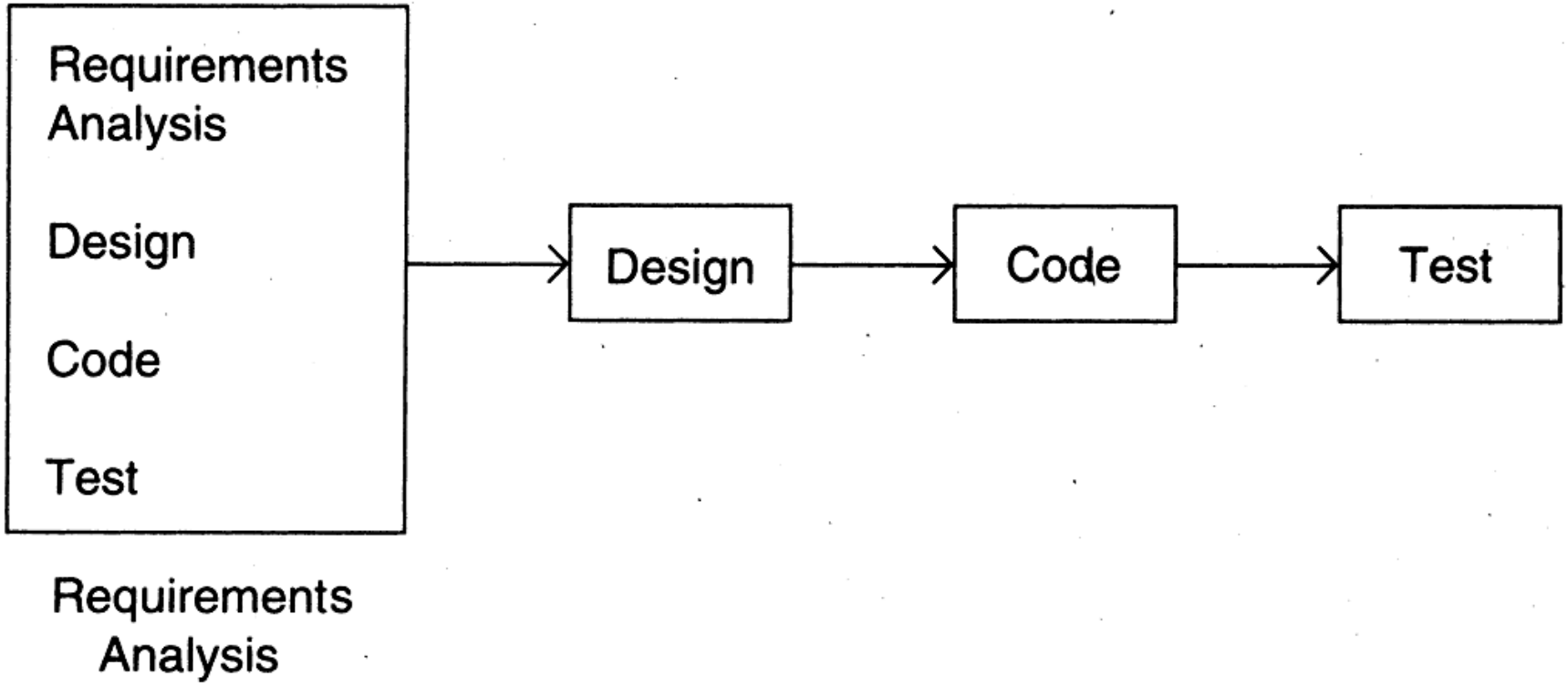
Waterfall Model (contd.)

- Limitations
  1. Assumes that the requirements can be frozen.
  2. Obsolete hardware.
  3. Requirements must be completely specified.
  4. Document-driven process.
- Conclusion
  - ✓ Well-suited for routine type of projects.
  - ✓ Most widely used process model.

Prototype Model

- A prototype is a working model of something built for study/testing/display.
- **{** The prototype is built on “currently known” requirements.
- This prototype is then given to clients & end-users to play with it.
- Clients & end-users provide their “feedback”.
- Based on the feedback, the prototype is modified to incorporate changes. **}**

Prototype Model (contd.)



Prototype Model (contd.)

- What about cost?

  Cost can be kept low …

  ✓Build only features needing clarification.

  ✓"quick and dirty" – quality not important, scripting etc can be used.

  ✓Things like exception handling, recovery , standards are omitted.

Prototype Model (contd.)

- Limitation
  1. Cost & Schedule may be hit.
- Conclusion
  - ✓ Well-suited for projects where requirements are hard to specify. (complicated & large system)
  - ✓ Reduces risks associated with the project.

Iterative Enhancement Model

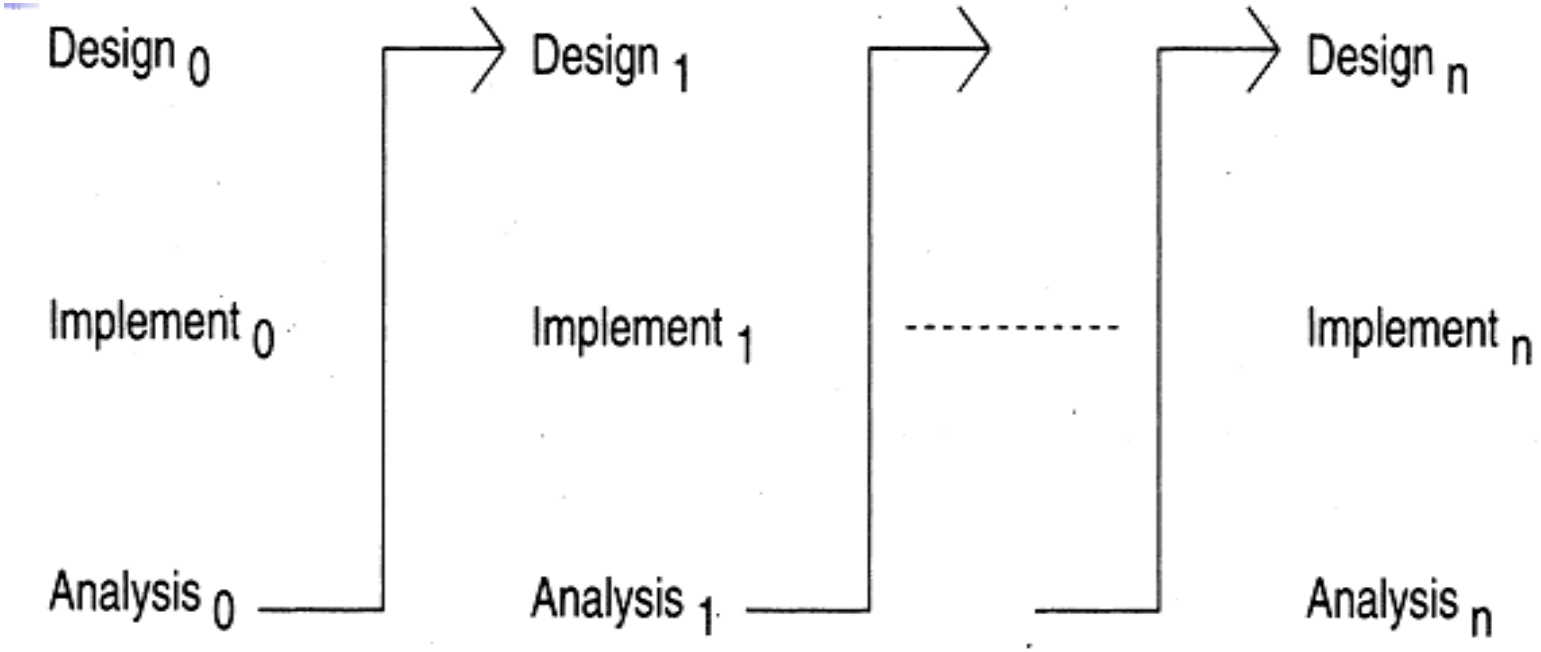
- The basic idea is – the software should be developed in small increments.
- Increments, here means the software with some functionality.
- Increments are improved until the software with all functionalities is developed.
- At each step, extensions & design modifications can be done.*

Iterative Enhancement Model (contd.)

- A project control list is created which contains, in order, all the tasks that must be performed to obtain the final version of the software.
- In the first step, a simple implementation is done for some of key aspects (that are easy to understand & implement).
- Each step consists of implementation of selected task, coding, testing, analysis of developed increment & updating the project control list.

Iterative Enhancement Model (contd.)

- The process is iterated until the project control list is empty.



Iterative Enhancement Model (contd.)

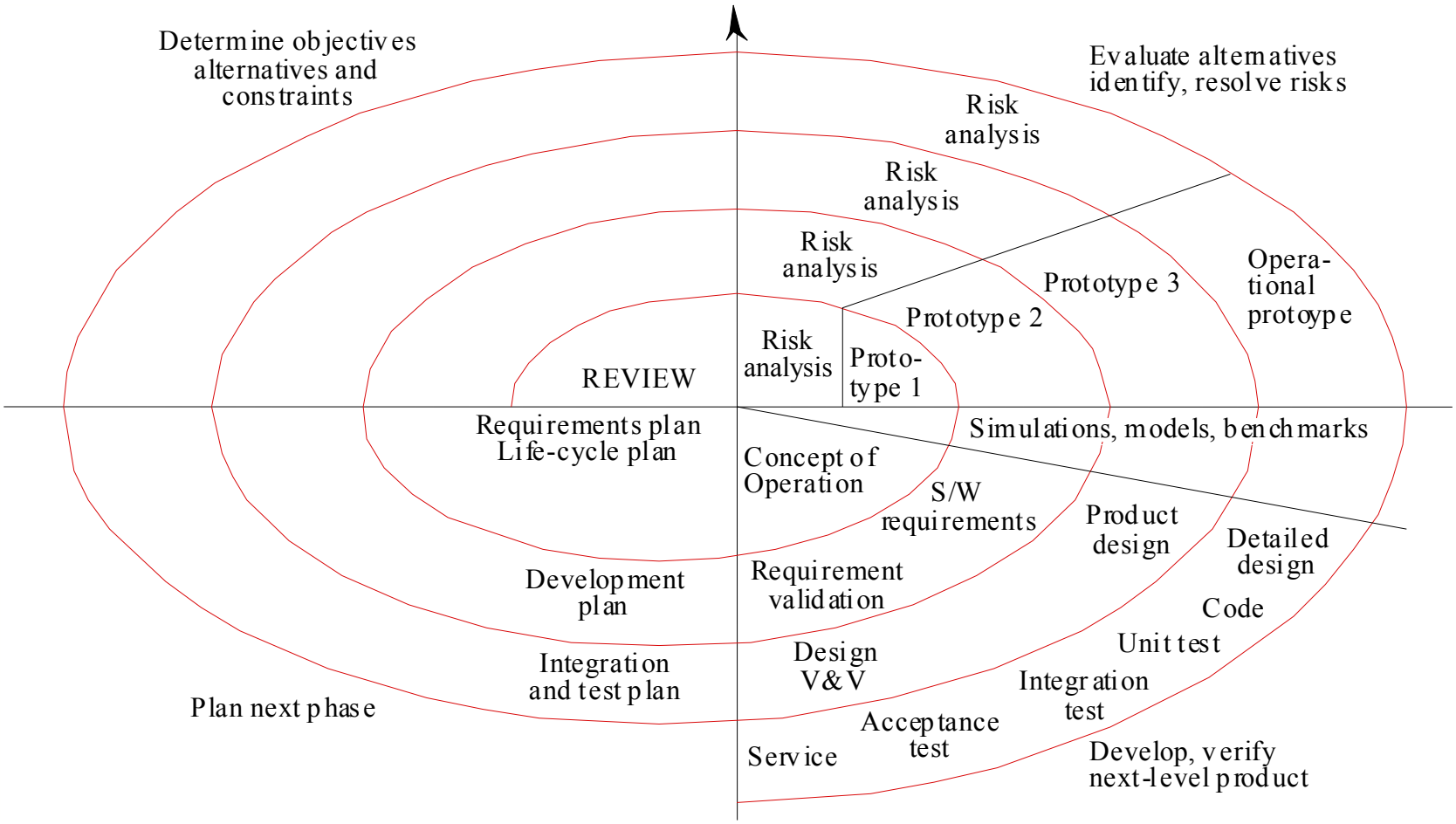
- Limitations
  1. If the client is to provide & approve the specifications, it is not clear how the process can be applied.
  2. Cost determination & hence generating business contract is difficult (because ___).
- Conclusion
  - ✓ Tries to combine the benefits of waterfall model & prototype model.

The Spiral Model

- Has been proposed by Boehm.
- Activities are organized in a ‘spiral’ that has many cycles rather than sequence.
- Four major phases are:
  1. Identification of objectives for that cycle, the different alternatives and the constraints.
  2. Evaluation different alternatives and risks.
  3. Development of software.
  4. Planning the next phase.
- Suitable for high-risk projects.

The Spiral Model (contd.)

This figure is for understanding only, the students are not supposed to draw it in exams.



The Spiral Model (contd.)

- Objective setting
  - Specific objectives for the phase are identified
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks
- Development and validation
  - A development model for the system is chosen which can be any of the generic models
- Planning
  - The project is reviewed and the next phase of the spiral is planned

Software Metrics

- Metrics (and not matrix) means a system of measurement.
- Software metrics can be used to quantify different characteristics of a software or the software development process.
- Two types:
  1. Product metrics
  2. Process metrics

Software Metrics (contd.)

- Product metrics are used to quantify the characteristics of the software.

  e.g. size, FP, complexity of the software.

- Process metrics are used to quantify the characteristics of the process.

  e.g. productivity, cost & resource requirements, QA of the process