Chapter 2 : Object-Oriented Software Development

Fundamentals of object-oriented programming
OOPs
Encapsulation
Abstraction
Polymorphism
Inheritance
Class
Object

ASSIT.PROF. JUTHAWUT CHANTHARAMALEE

CURRICULUM OF COMPUTER SCIENCE

FACULTY OF SCIENCE AND TECHNOLOGY, SUAN DUSIT UNIVERSITY

Outline of this presentation

1. Software Process
2. Object-Oriented Software Development
3. Software Life-Cycle Models
4. Object Orientation
5. Software Quality Assessment

Software Process

The software process is the way we produce software.

It incorporates the *software life-cycle model*, the *tools* we use, and the *individuals* building the software.

Object-Oriented Software Development

Three key words.

1. Software
2. Development
3. Object Orientation

Let us look at each in turn

Software

1. Programs
2. Documentation during the development of programs (e.g. specification)
3. Primary aids for running the programs (e.g. user manuals)
4. Secondary aids for running the programs (e.g. key boards overlays)

Software is not just programs!

Software Life Cycle

1. Software is like humans.
2. It has a life cycle.
3. Software in a system is conceptualized first.
4. It becomes obsolescent at the end.
5. The period in between is called the software life cycle.

Software Life Cycle Models

1. Build-and-Fix Model
2. Waterfall Model
3. Rapid prototyping model
4. Incremental Model
5. Spiral Model
6. Concurrent Development Model
7. Formal Methods Model

Built-and-Fix Model

1. Unfortunately, many s/w products are developed with built-and-fix model.

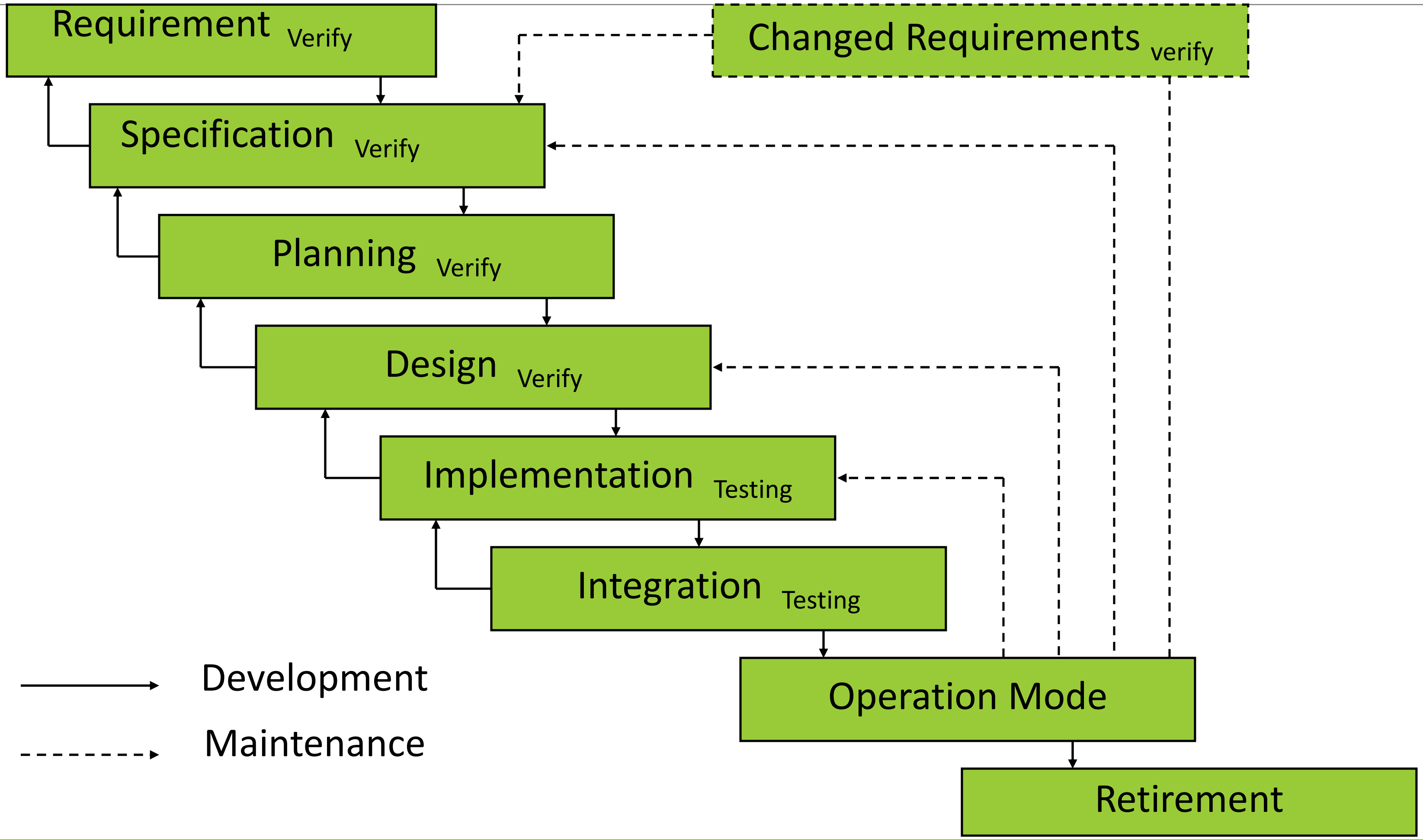
2. Without specification or any attempt in design, just build a product, and reworked as many times needed to satisfy the customer.

3. Unsatisfactory for any size of s/w development, we better specify the various phases of software process.

Why use a life cycle model?

1. Life cycle model breaks down the development process into phases or stages.
2. This is because software development is complex.
3. Breaking down the development process makes it easier to manage.
4. Each phase can be performed in various ways.

Waterfall Model

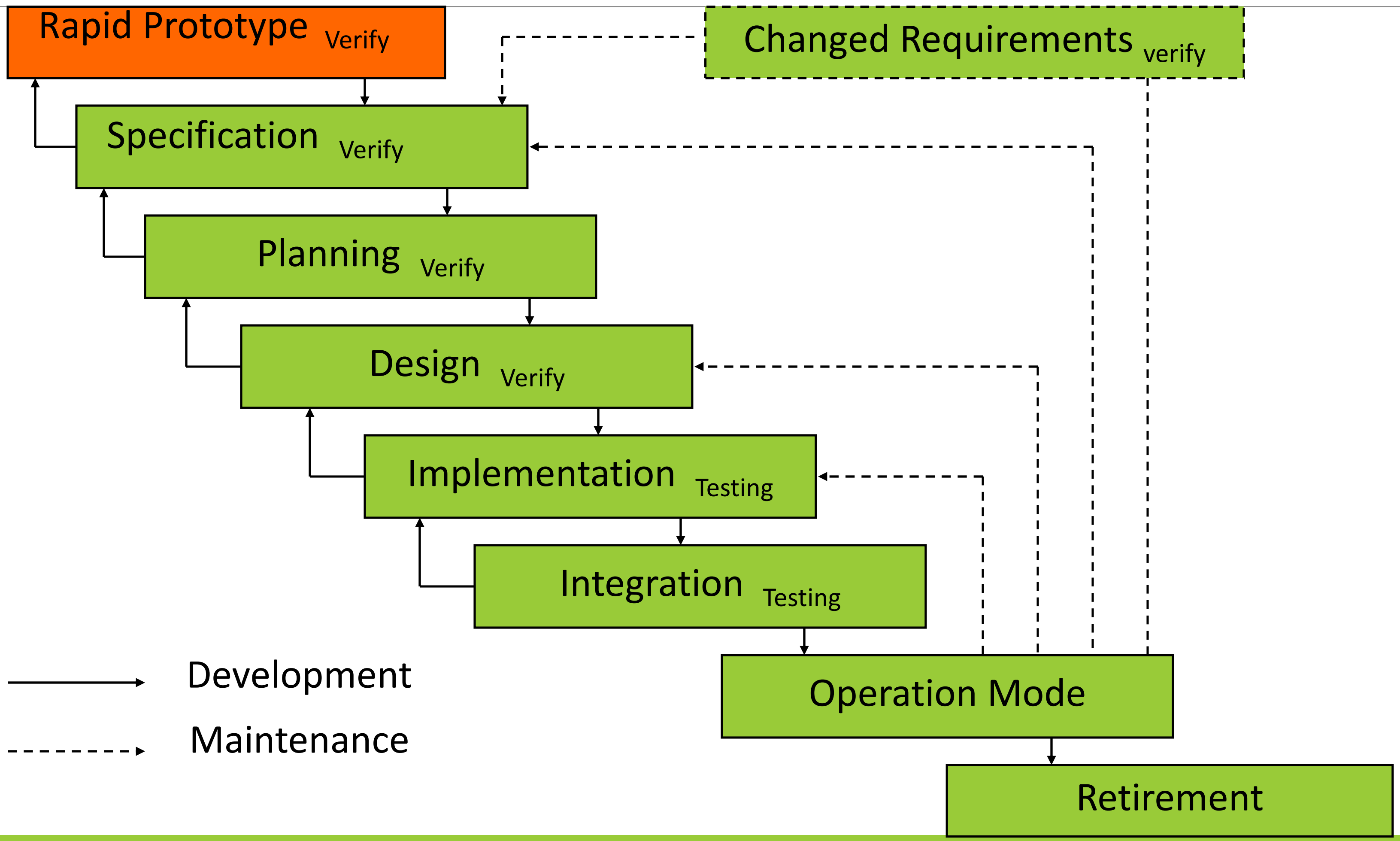


Rapid Prototyping Model

A rapid prototype is a working model that is functionally equivalent to a subset of the product (internal structure is not concerned yet).

The sole use of rapid prototyping is to determine what the client’s real needs are, construct the rapid prototype as rapidly as possible to speed up the s/w development process.

Rapid Prototyping Model

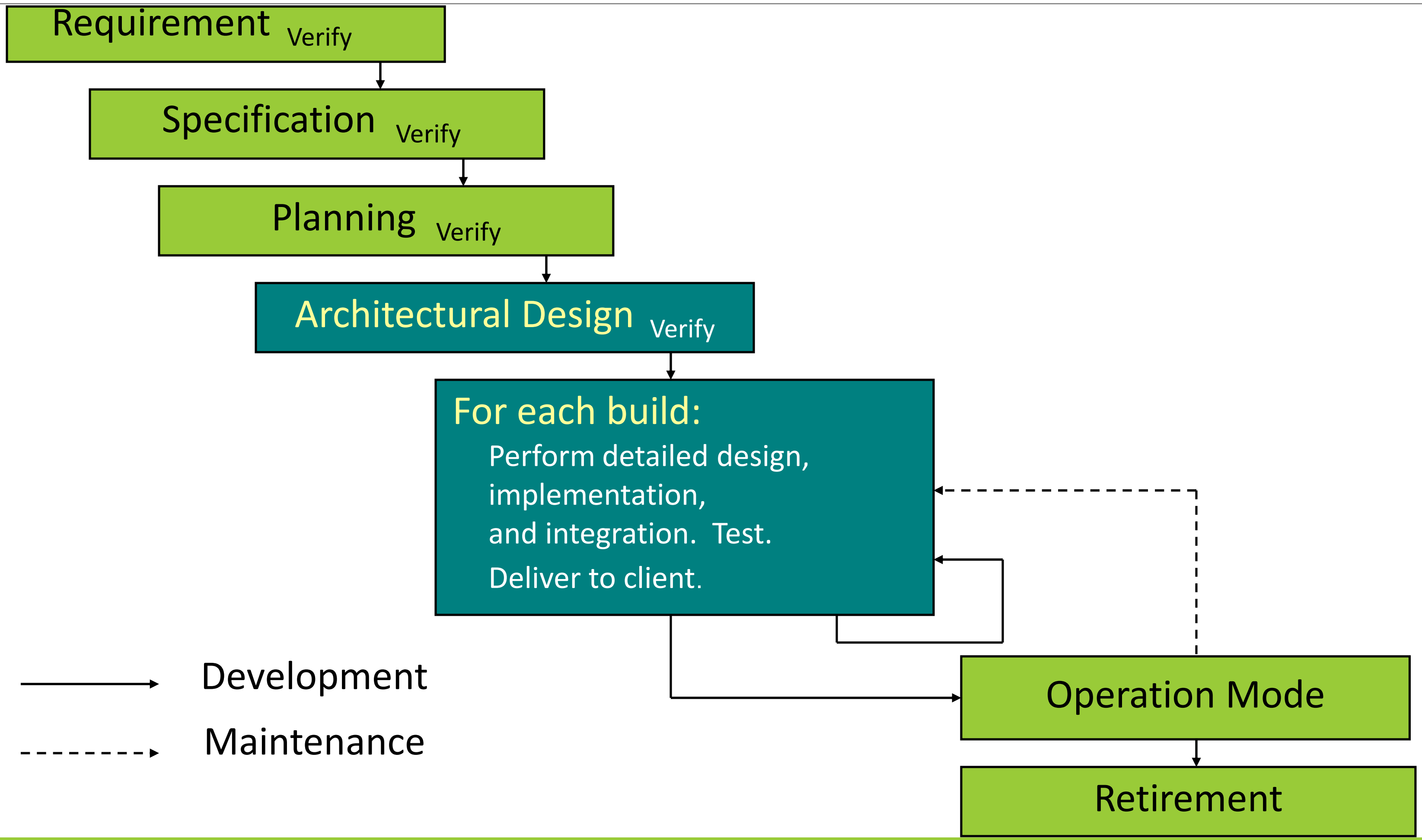


Incremental Model

The s/w product is designed, implemented, integrated, and tested as a series of incremental *builds*, where a build consists of code pieces from various modules interacting to provide a specific functional capability.

It is sometimes necessary to re-specify, re-design, re-code, or at worst, throw away what has already been completed and start again.

Incremental Model



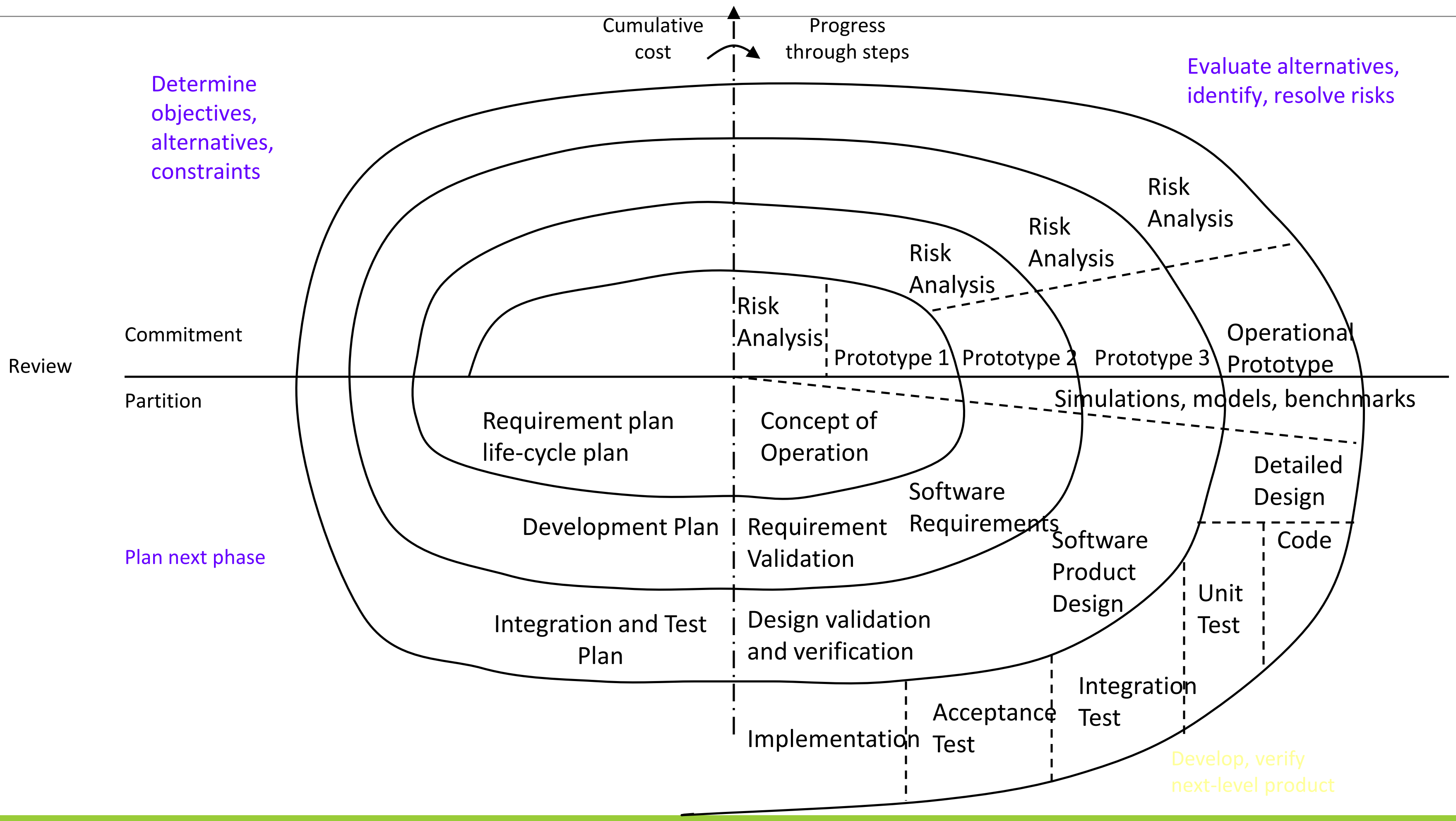
Spiral Model

The idea of minimizing risk via the use of prototypes and other means is the concept underlying the *spiral model*.

A simplified spiral model is as a waterfall model with each phase preceded by risk analysis.

- Before commencing each phase, an attempt is made to control (resolve) the risks. If it is impossible to resolve all the significant risks at a stage, then the project is immediately terminated.

Full Spiral Model [Boehm, IEEE 1998]



Software Development

Software is developed using a life cycle model.

Just a life cycle model is insufficient for development.

We need:

- A broad philosophy
- A set of tools which support the philosophy.
- A language which supports the philosophy.

Software Development Paradigm

1. A paradigm provides a general approach to work during each phase of the life cycle model.
2. A paradigm is a broad philosophy.
3. A paradigm is not a specific model.

Some Software Development Paradigms

1. Functional Composition
2. Logic Programming
3. Structured Development
4. Object Orientation

Functional Development

1. A problem is expressed in termed of a set of mathematical functions.
   - e.g. *Double(x) = Add(x, x).*
2. An algorithm is not specified.
3. Language such as Miranda, Gofer, Haskell support this paradigm.
4. Poor execution speed.

Logic Programming

Consists of a problem description only.

- e.g. *Factorial(0) = 1.*

  Factorial(N) = N x Factorial(N -1).

Doesn’t describe how to solve the problem.

Languages Prolog & Lisp support this paradigm.

Structured Development

Also called SASD, SADT & Functional Decomposition.

Breaks the system into processes & decomposes them.

Languages C, Fortran, Pascal, Cobol, Basic and a lot more support this paradigm.

By far the most popular paradigm.

Object Orientation

- Most recent paradigm.
- Treats a problem as a collection of objects.
- Becoming very popular now.
- More and more languages support this paradigm now.

Tools for Object Orientation

- Rambaugh (OMT)
- Coad-Yourdon
- Booch
- UML

Languages for OO

- C++
- Smalltalk
- Eiffel
- Object C
- Object Pascal
- Java

Software Quality Assessment

Capability Maturity Model

- a strategy for improving the software process, irrespective of the actual life-cycle model used.

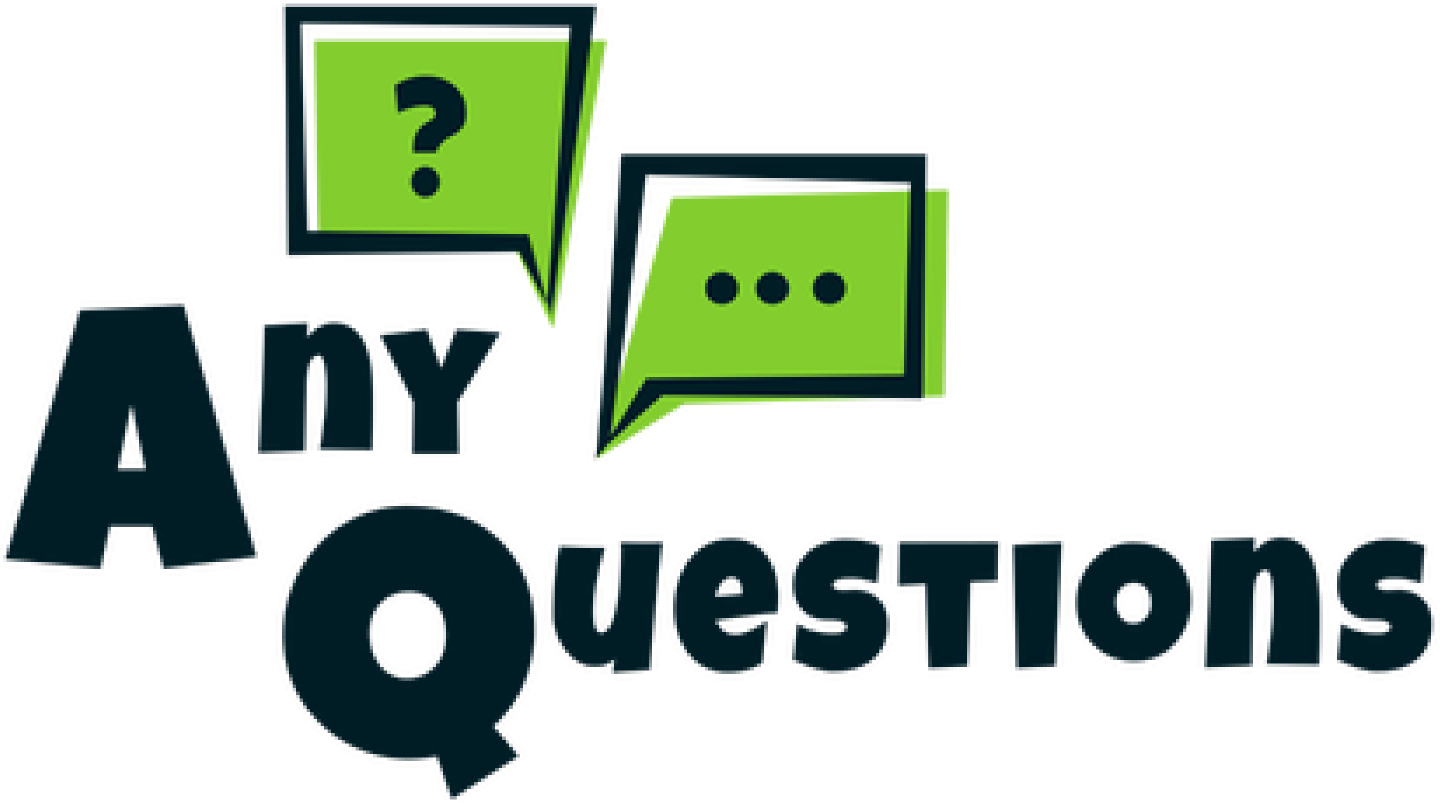
ISO 9000

- ISO 9000 is a series of five related standards that are applicable to a wide variety of industrial activities, including design, development, production, installation, and servicing.
- Standard ISO 9001 for quality systems is the standard that is most applicable to software development.
- ISO 9000-3 gives specific guidelines to assist in applying ISO 9001 to software development.

Capability Maturity Model (CMM)

- Proposed by W. Humphrey (1986), Software Engineering Institute (SEI), CMU
- The strategy of CMM is to improve organizational-wide management of software process.
- The improved process should result in better quality software, then less suffering from time and cost overrun.

Maturity Level	Characterization
1. Initial Level	1. Ad hoc process
2. Repeatable Level	2. Basic project management
3. Defined Level	3. Process definition
4. Managed Level	4. Process measurement
5. Optimizing Level	5. Process control


?
...
Any
Questions