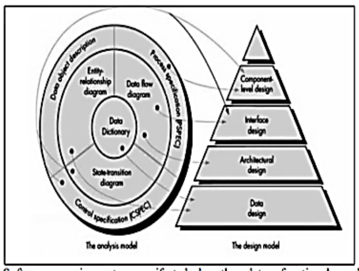
Prescriptive process model and agile process model.

Prescriptive process model	agile process mode
Prescriptive process models stress detailed definition, identification, and application of process activates and tasks.	Agile process models emphasize project "agility" and follow a set of principles that lead to a more informal approach to software process.
A prescriptive model also describes how each of these elements are related to one another.	Agile methods note that not only do the software requirements change, but so do team members, the technology being used.
It is Process oriented.	It is people oriented.
It follows Life cycle model (waterfall, spiral) development model.	It follows Iterative and Incremental development model.
Documentation required is to be comprehensive and constant.	Documentation required is to be minimal and evolving.
Predictive planning is required	Adaptive planning is required.

Sr .n o	White box testing	Black Box Testing
1	The tester needs to have the knowledge of internal code or program.	This technique is used to test the software without the knowledge of internal code or program.
2	It aims at testing the structure of the item being tested.	It aims at testing the functionality of the software.
3	It is also called structural testing, clear box testing, code-based testing, or glass box testing.	It also knowns as data-driven, box testing, data-, and functional testing.
4	Testing is best suited for a lower level of testing like Unit Testing, Integration testing.	This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing.
5	Statement Coverage, Branch coverage, and Path coverage are White Box testing technique.	Equivalence partitioning, Boundary value analysis are Black Box testing technique
6	Can be based on detailed design documents.	Can be based on Requirement specification document.

Draw and explain Transition diagram from requirement model to design model

Transition diagram from requirement model to design model

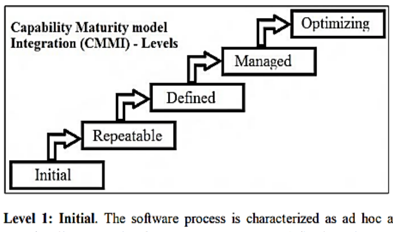


Software requirements, manifested by the data, functional, and behavioural models, feed the design task. Using one of a number of design methods, the design task produces a data design, an architectural design, an interface design, and a component design. Each of the elements of the analysis model provides information that is necessary to create the four design models required for a complete specification of design.

Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer's requirements and at the same time assessed for quality against a set of predefined criteria for —goodl design. In the software engineering context, design focuses on four major areas of concern: data, architecture, interfaces, and components Design begins with the requirements model.

Describe CMMI. Give significance of each level.

The Capability Maturity Model Integration (CMMI), a comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity. The CMMI represents a process meta-model in two different ways: (1) Continuous model and (2) Staged model. The continuous CMMI meta-model describes a process in two dimensions. Each process area (e.g. project planning or requirements management) is formally assessed against specific goals and practices and is rated according to the following capability levels:



Level 1: Initial. The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

Level 2: Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 3: Defined. The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level 2

Level 4: Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3

Level 5: Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.

Identify and enlist requirement for given modules of employ management software

i. Employee detail

ii. Employee salary

iii.Employee performance

This is with perspective of employee management software. Requirements for following modules will be as

i. Employee details
  a. Getting information about the customer
  b. Updation of employee details (department, change of address, emp_code etc)
  c. Assignment of tasks , duties and responsibilities.
  d. Recording of employee attendance.

ii. Employee salary
  a. Salary calculation
  b. Allowances, special bonus calculation and approval
  c. Tax statement/certificate
  d. Apply loan/approvals

iii. Performance
  a. Recording annual performance
  b. Details about parameters for performance appraisal
  c. Analysis performance and determining hike in payment.

Calculate using COCOMO model
i)Effort
ii)Project duration
iii)Average staff size
If estimated size of project is 200 KLOC using organic mode.

Given data: size=200 KLOC mode= organic

1. Effort:

   $E = a\ (KLOC)^{b}$

   For organic a=2.4 and b= 1.05

   $E= 2.4\ (200)^{1.05}$

   $=$ 626 staff members

2. Project duration:

   $TDEV= c\ (E)^{d}$

   Where TDEV= time for development

   c and d are constant to be determined

   E = effort

   For organic mode, c= 2.5 and d= 0.38

   $TDEV= 2.5\ (626)^{0.38}$

   $=$ 29 months

3. Average staff size:

   SS = E/TDEV

   SS = 626/29 = 22 staffs

Compare CMMI and ISO for software w.r.to
i)scope
ii)Approach
Iii) Implementation.

Difference between CMMI and ISO based on

SCOPE: CMMI is rigid and extends only to businesses developing software intensive systems. ISO is flexible and applicable to all manufacturing industries. CMMI focuses on engineering and project management processes whereas ISO's focus is generic in nature.

CMMI mandates generic and specific practices and businesses have a choice of selecting the model relevant to their business needs from 22 developed process areas. ISO requirements are same for all companies, industries, and disciplines.

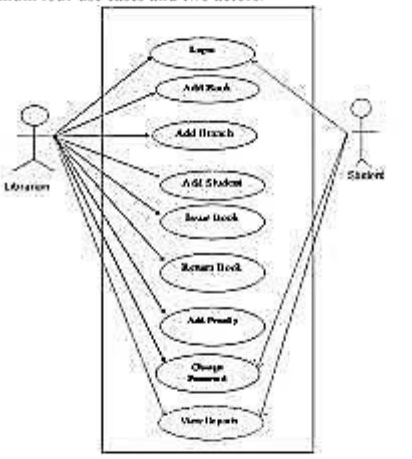
APPROACH:CMMI requires ingraining processes into business needs so that such processes become part of corporate culture and do not break down under the pressure of deadlines. ISO specifies to conformance and remains oblivious as to whether such conformance is of strategic business value or not.CMMI approaches risk management as an organized and technical discipline by identifying risk factors, quantifying such risk factors, and tracking them throughout the project life cycle. ISO was until recently neutral on risk management. ISO 31000:2009 now provides generic guidelines for the design, implementation, and maintenance of risk management processes throughout an organization.

Although CMMI focuses on linkage of processes to business goals, customer satisfaction is not a factor in the ranking whereas customer satisfaction is an important part of ISO requirements.

IMPLEMENTATION:

Neither CMMI nor ISO requires the establishment of new processes. CMMI compares the existing processes to industry best practices whereas ISO requires adjustment of existing processes to confirm to the specific ISO requirements. In practice, some organizations tend to rely on extensive documentation while implementing both CMMI and ISO. Most organizations tend to constitute in-house teams, or rely on external auditors to see through the implementation process.

Sketch a use case diagram for library management system with minimum four use cases and two actors.



Describe any four principles of communication for software engineering :

Principle 1 Listen:

- Try to focus on the speaker's words, rather than formulating your response to those words.
- Ask for clarification if something is unclear, but avoid constant interruptions.
- Never become contentious in your words or actions (e.g., rolling your eyes or shaking your head) as a person is talking.

Principle 2 Prepare before you communicate:

- Spend the time to understand the problem before you meet with others. If necessary, perform some research to understand business domain.
- If you have responsibility for conducting a meeting, prepare an agenda in advance of the meeting.

Principle 3 someone should facilitate the activity:

- Every communication meeting should have a leader (a facilitator)
- To keep the conversation moving in a productive direction,
- To mediate any conflict that does occur, and
- To ensure that other principles are followed.

Principle 4 Face-to-face communication is best:

- It usually works better when some other representation of the relevant information is present.
- For example, a participant may create a drawing /document that serve as a focus for discussion.

Differentiate between Software Quality Management and Software Quality Assurance (any two points).

Software Quality Assurance (QA)	Software Quality Control (QC)
• It is a procedure that focuses on providing assurance that quality requested will be achieved	• It is a procedure that focuses on fulfilling the quality requested.
• QA aims to prevent the defect	• QC aims to identify and fix defects
• It is a method to manage the quality- Verification	• It is a method to verify the quality-Validation
• It does not involve executing the program	• It always involves executing a program
• It's a Preventive technique	• It's a Corrective technique
• It's a Proactive measure	• It's a Reactive measure

Explain the concept of black box testing and white box testing.

Black Box Testing:

- It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.
- It also known as data-driven, box testing, data-, and functional testing.
- This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing.
- It is mostly done by software testers.
- No knowledge of implementation is needed.
- It is functional test of the software.
- Testing can start after preparing requirement specification document.

White Box Testing:

- It is a way of testing the software in which the tester has knowledge about the internal structure r the code or the program of the software.
- It is also called structural testing, clear box testing, code-based testing, or glass box testing.
- Testing is best suited for a lower level of testing like Unit Testing, Integration testing.
- It is mostly done by software developers.
- Knowledge of implementation is required.
- It is structural test of the software.
- Testing can start after preparing for Detail design document.