CBCS SCHEME

USN 1 C R 2 0 1 5 C 1 8

18CS35

Third Semester B.E. Degree Examination, Feb./Mar. 2022
Software Engineering

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. Define software engineering. What are the different types of software products? (06 Marks)
b. Explain briefly the Software Engineering Ethics. (06 Marks)
c. List and explain the different types of Application Softwares. (08 Marks)

OR

2 a. What are the fundamental software process activities? With neat diagram, explain requirement engineering process. (08 Marks)
b. With neat diagram, explain Bohem's Spiral model. (08 Marks)
c. Explain Re-use oriented Software Engineering. (04 Marks)

Module-2

3 a. What is object orientation? Explain the characteristics of object oriented approach. (10 Marks)
b. Define model. Explain the three different models of object orientation. (10 Marks)

OR

4 a. Explain the following with suitable diagrams:
(i) Links and Associations
(ii) Generalization (10 Marks)
b. With neat diagram, explain the class model of a Windowing System. (10 Marks)

Module-3

5 a. With neat diagram, explain the context model for MHC-PMS system. (10 Marks)
b. Explain the state diagram of microwave oven. (10 Marks)

OR

6 a. Explain the Rational Unified Process. (06 Marks)
b. Explain Design Pattern with UML model of the observer model. (08 Marks)
c. What are the different implementation issues of Software Engineering? (06 Marks)

Module-4

7 a. What are the two distinct goals of Software Testing? (05 Marks)
b. Explain the three different types of testing carried out during software development. (05 Marks)
c. What are the different types of user testing? With neat diagram, explain the six stages of acceptance testing process. (10 Marks)

OR

8 a. Write the Lemman's law of program dynamic evolution. (06 Marks)
b. With neat diagram, explain the software reengineering process activities. (08 Marks)
c. What are the four strategic options for Legacy Systems? (06 Marks)

Module-5

9	a.	What are the factors affecting the pricing of software product?	(04 Marks)
	b.	With neat diagram, explain the project planning process.	(06 Marks)
	c.	With neat diagram, explain the COCOMO – II estimation model.	(10 Marks)

OR

10	a.	Explain the product standards and process standards in software quality management.	(06 Marks)
	b.	Explain three phases of software review process.	(08 Marks)
	c.	Explain the various inspection checks in the program inspection.	(06 Marks)

* * * * *

USN				

Internal Assessment Test 4 – March 2022

Sub:	Software Engineering				Sub Code:	18CS35	Branch:	ISE	
Date:	15/3/2022	Duration:	180 min's	Max Marks: 100	Sem/Sec:	III / A, B and C			OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a. What is software engineering? What are the different types of software products?	6	CO1	L2

Solution:

software engineering

An engineering discipline that is concerned with all aspects of software production from the early stages of system specification to maintaining the system after

Generic products.

- Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
- Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
- The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.

Customized or bespoke products.

- Software that is commissioned by a specific customer to meet their own needs.
- Examples – embedded control systems, air traffic control software, traffic monitoring systems.
- The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

		MARKS	CO	RBT
	b. Explain briefly the software engineering ethics.	6	CO1	L2

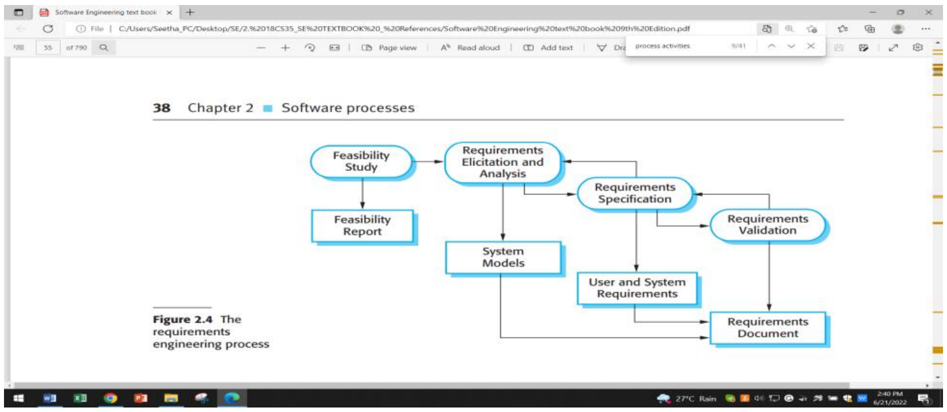
Solution:

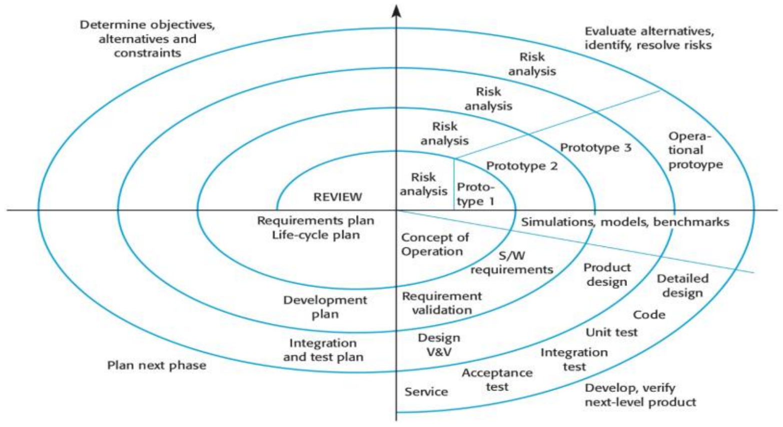
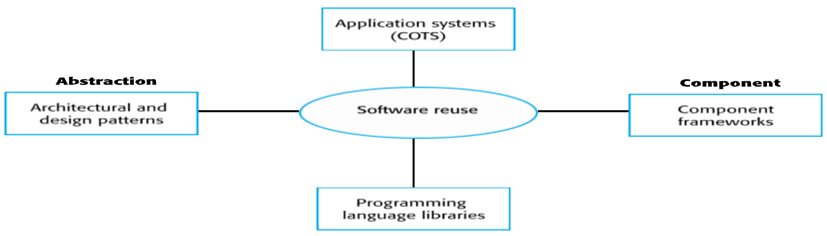
1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

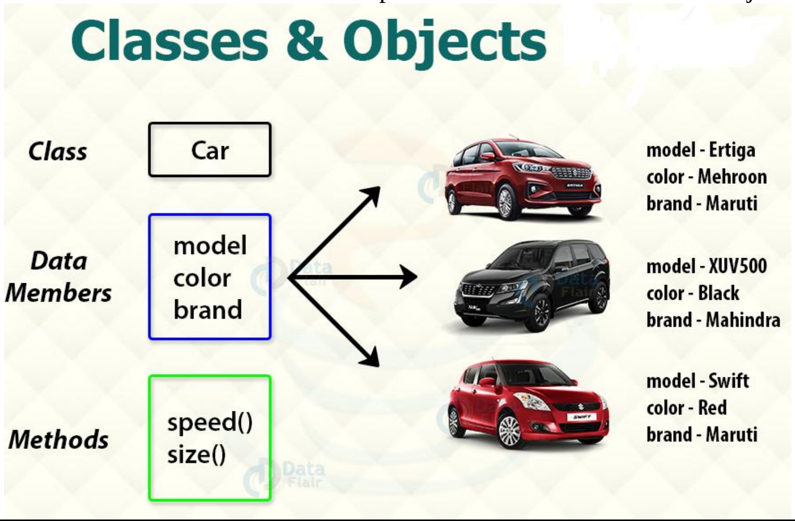
		MARKS	CO	RBT
	c. List and Explain the Different types of application softwares	8		

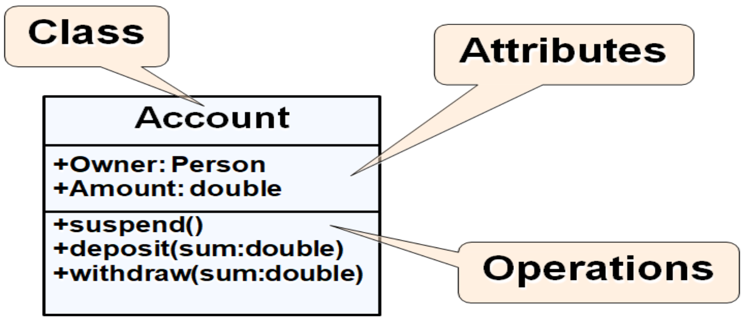
Solution:

1. Stand-alone applications These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
2. Interactive transaction-based applications: Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.

	3. Embedded control systems These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system. 4. Batch processing systems These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs. 5. Entertainment systems These are systems that are primarily for personal use and which are intended to entertain the user. 6. Systems for modeling and simulation These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.			
2	a. What are the fundamental software process activities? with neat diagram explain the requirement engineering process. Solution: • **Software specification**, where customers and engineers define the software that is to be produced and the constraints on its operation. • **Software development**, where the software is designed and programmed. • **Software validation**, where the software is checked to ensure that it is what the customer requires. • **Software evolution**, where the software is modified to reflect changing customer and market requirements.  38 Chapter 2 Software processes Feasibility Study Requirements Elicitation and Analysis Requirements Specification Requirements Validation Feasibility Report System Models User and System Requirements Requirements Document Figure 2.4 The requirements engineering process Fig: Requirement Engineering Process	8	CO1	L2

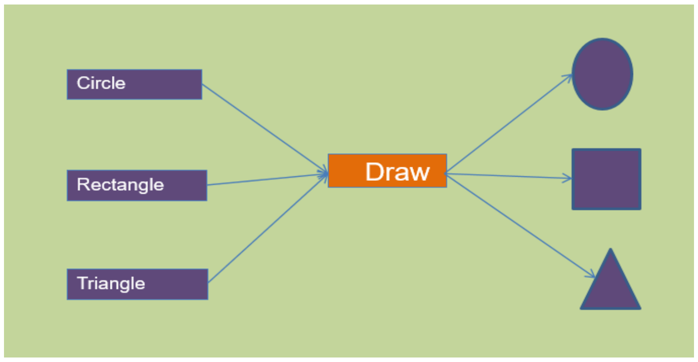
2	B. With neat diagram explain Bohem's spiral model? • **Process is represented as a spiral rather than as a sequence of activities with backtracking.** • **Each loop in the spiral represents a phase in the process.** • **Combines change avoidance with change tolerance. It assumes that changes are a result of project risks and includes explicit risk management activities to reduce these risks.** • **Risks are explicitly assessed and resolved throughout the process.**  Determine objectives, alternatives and constraints Evaluate alternatives, identify, resolve risks Risk analysis Risk analysis Risk analysis Risk analysis Prototype 3 Opera-tional protoype Prototype 2 Proto-type 1 REVIEW Requirements plan Life-cycle plan Simulations, models, benchmarks Concept of Operation S/W requirements Product design Detailed design Development plan Requirement validation Code Unit test Integration and test plan Design V&V Integration test Plan next phase Acceptance test Service Develop, verify next-level product	8		L2
	c Explain reuse oriented software engineering**.** **Solution:** **Reuse** ✧ From the 1960s to the 1990s, most new software was developed from scratch, by writing all code in a high- level programming language. ✧ The only significant reuse or software was the reuse of functions and objects in programming language libraries. ✧ Costs and schedule pressure mean that this approach became increasingly unviable, especially for commercial and Internet-based systems. ✧ An approach to development based around the reuse of existing software emerged and is now generally used for business and scientific software**.**  Application systems (COTS) Abstraction Architectural and design patterns Software reuse Component Component frameworks Programming language libraries **Fig. Software Reuse** **Reuse levels** ✧ The abstraction level ✧ At this level, you don't reuse software directly but use knowledge of successful abstractions in the design of your software. ✧ The object level ✧ At this level, you directly reuse objects from a library rather than writing the code yourself. ✧ The component level	4	CO4	

	✧ Components are collections of objects and object classes that you reuse in application systems. ✧ The system level At this level, you reuse entire application systems			
3	**a** What is object orientation? Explain the characteristics of object orientation approach? Solution: OO means that we organize software as a collection of discrete objects (that incorporate both data structure and behavior). There are four aspects (characteristics) required by an OO approach, • Identity. • Classification. • Inheritance • Polymorphism. i) Identity Identity means that data is organized into discrete, distinguishable entities called objects. An object has: • *state* - descriptive characteristics • *behaviors* - what it can do (or what can be done to it) • The state of a bank account includes its account number and its current balance • The behaviors associated with a bank account include the ability to make deposits and withdrawals • Note that the behavior of an object might change its state Software objects model read-world objects or abstract concepts • dog, bicycle, Bank account Real-world objects have states and behaviors • Dogs' states: name, color, breed, hungry • Dogs' behaviors: barking fetching ***Objects have three responsibilities:*** What they know about themselves – (e.g., Attributes) What they do – (e.g., Operations) What they know about other objects – (e.g., Relationships) **Classification** It means that objects with same data structure (attribute) and behavior (operations) are **grouped into a class.** • A *class* is simply a representation of a type of *object*. It is the blueprint/ plan/ template that describe the details of an *object*. • A class is the blueprint from which the individual objects are created.  Classes & Objects Class Car Data Members model color brand Methods speed() size() model - Ertiga color - Mehroon brand - Maruti model - XUV500 color - Black brand - Mahindra model - Swift color - Red brand - Maruti	10	CO3	L2



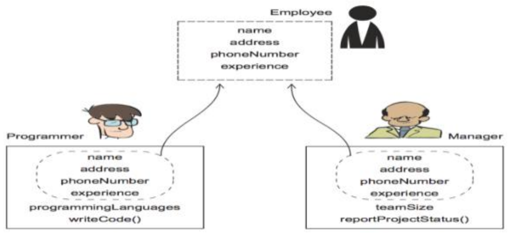
Polymorphism

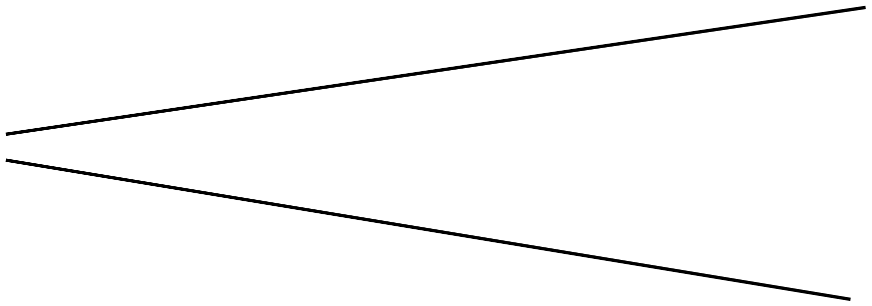
- It means that the same operation (i.e. action or transformation that the object performs) may behave differently on different classes.
- Ability of different objects to response same message in different ways.
- Ability of an object to take on multiple forms.
- In a programming language, class objects belonging to the same hierarchical tree may have functions with the same name, but with different behaviors.

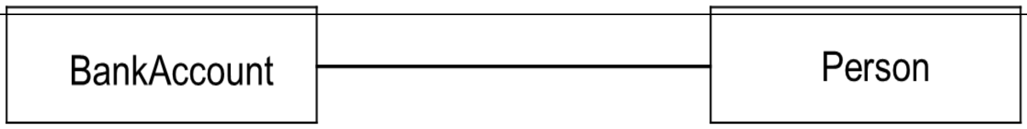


iv)Inheritance:

- It is the sharing of attributes and operations among classes based on a hierarchical relationship.
- Subclasses can be formed from broadly defined class.
- Each subclass incorporates or inherits all the properties of its super class and adds its own unique properties.

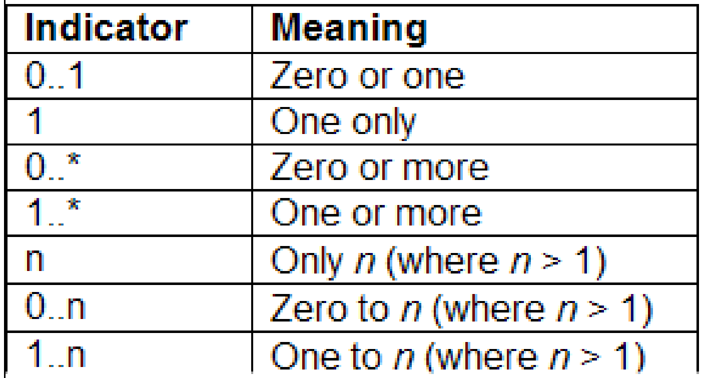


	b. Explain three different models of object orientation. Solution 1**. Class Model**—for the objects in the system & their relationships. It describes the static structure of the objects in the system and their relationships. Class model contains class diagrams- a graph whose nodes are classes and arcs are relationships among the classes. 2. **State model**—for the life history of objects. It describes the aspects of an object that change over time. It specifies and implements control with state diagrams-a graph whose nodes are states and whose arcs are transition between states caused by events. 3**. Interaction Model**—for the interaction among objects. It describes how the objects in the system co-operate to achieve broader results. This model starts with use cases that are then elaborated with sequence and activity diagrams. • Use case – focuses on functionality of a system – i.e what a system does for users. • Sequence diagrams – shows the object that interact and the time sequence of their interactions. • Activity diagrams – elaborates important processing step	10	CO3	L2
4	a. Explain the following with suitable diagrams (i)Links and Association ii) Generalization **Solution:** Links and associations are the means for establishing relationships among objects and classes. • A link is a physical or conceptual connection among objects. E.g. JoeSmith *WorksFor Simplex Company*. • An association is a description of a group of links with common structure and common semantics. E.g. a person *WorksFor a company*.  Association: • If two classes in a model need to communicate with each other, there must be link between them, and that can be represented by an association (connector). • Associations are inherently bi-directional. • The association name is usually read in a particular direction but the binary association may be traversed in either direction • Association can be represented by a line between these classes with an arrow indicating the navigation direction. In case arrow is on the both sides, association has bidirectional association. 	10	CO3	L2

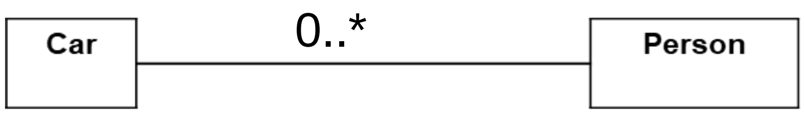


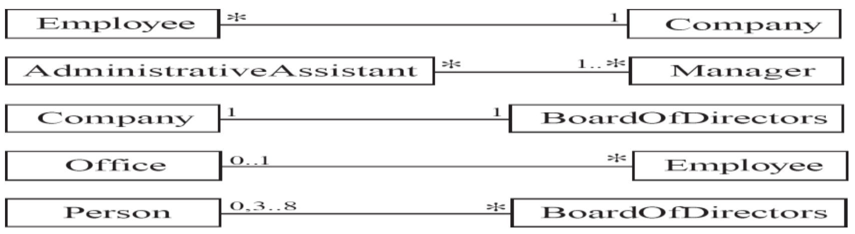
Multiplicity

- Multiplicity defines the number of objects associated with an instance of the association.
- UML diagrams explicitly list multiplicity at the end of association lines.
- Intervals are used to express multiplicity:



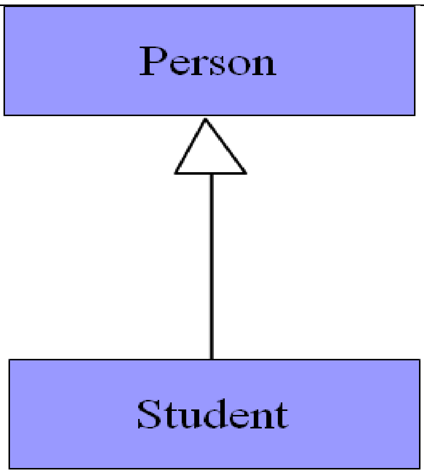
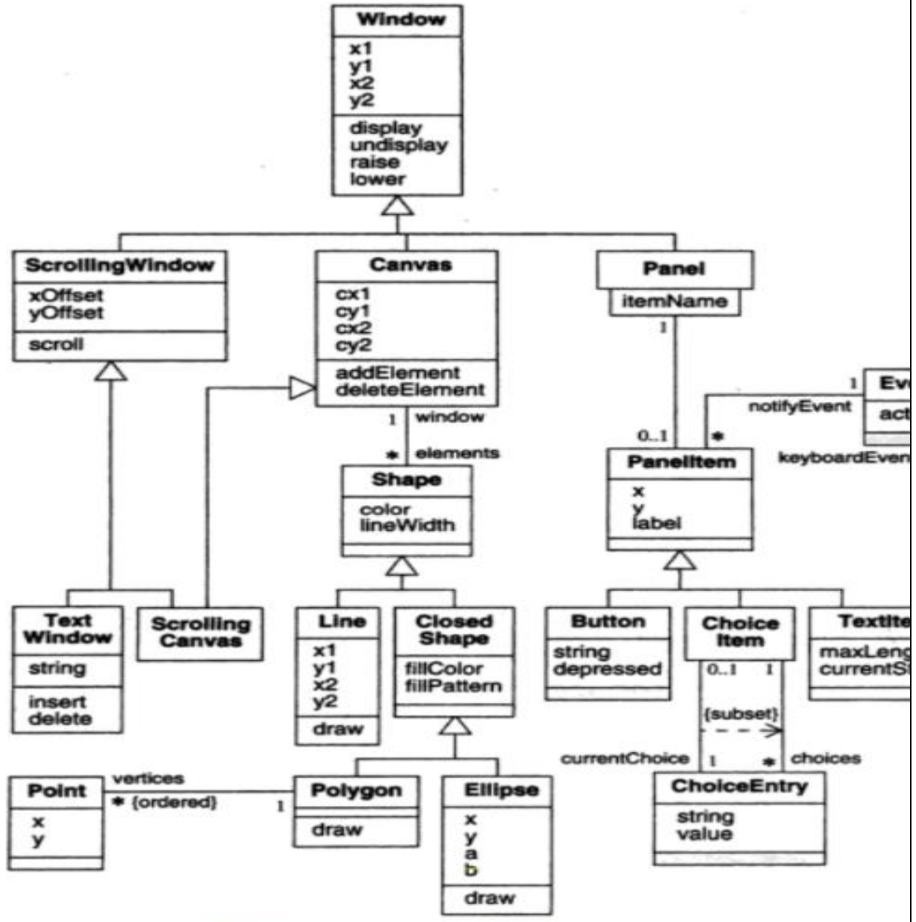
Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
n	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)



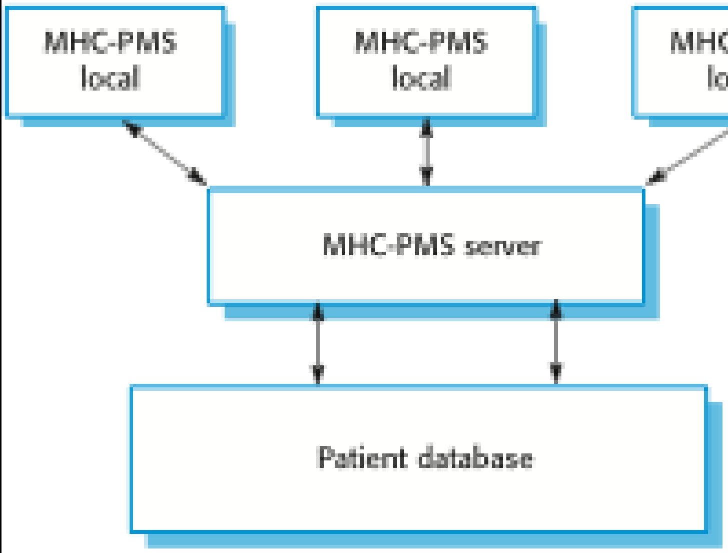


(ii)Generalization

- Deriving a class out of a parent class having some inherited property(from the parent class) and some new property of the derived class.
- The term generalization is for the inheritance in the bottom to the up direction i.e. from derived class to the parent class.
- Generalization is the relationship between a class (superclass) and one or more variations of the class (subclasses).
- A superclass holds common attributes, attributes and associations.
- The subclasses adds specific attributes, operations, and associations. They inherit the features of their superclass.
- Generalization is called a “IS A” relationship

	 A *generalization* connects a subclass to its superclass.			
	b. With neat diagram explain the class model of a Windowing system **Solution:**  Window x1 y1 x2 y2 display undisplay raise lower ScrollingWindow xOffset yOffset scroll Canvas cx1 cy1 cx2 cy2 addElement deleteElement Panel itemName 1 0..1 1 Ev act notifyEvent * keyboardEven 1 window * elements Shape color lineWidth PanelItem x y label Text Window string insert delete Scrolling Canvas Line x1 y1 x2 y2 draw Closed Shape fillColor fillPattern Button string depressed Choice Item 0..1 1 {subset} Textite maxLeng currentS currentChoice 1 * choices ChoiceEntry string value Point x y vertices * {ordered} 1 Polygon draw Ellipse x y a b draw Figure 3.26 Class model of a windowing system	10	CO1	L2
5	a. With neat diagram explain the context model of MHC-PMS system **Solution:** • The MHC-PMS (Mental Health Care-Patient Management System) is an information system that is intended for use in clinics. • It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity. • When the local systems have secure network access, they use patient	10	CO4	L2

information in the database but they can download and use local copies of patient records when they are disconnected.



MHC-PMS key features

- Individual care management
    - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.
- Patient monitoring
    - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.
- Administrative reporting
- The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

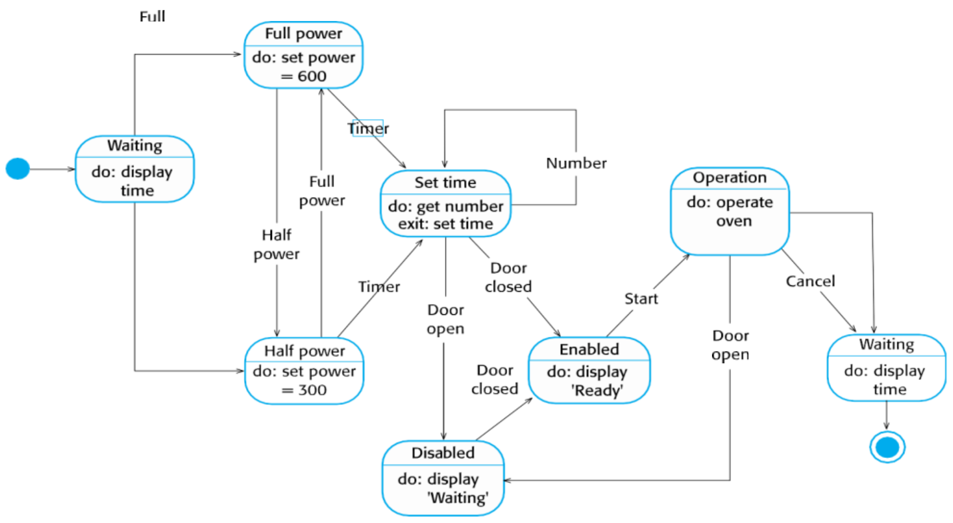
MHC-PMS

- **Privacy**
  It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.
- **Safety**
  Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
  The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

	b. Explain the state diagram of microwave oven	10	CO4	L1

Solution:

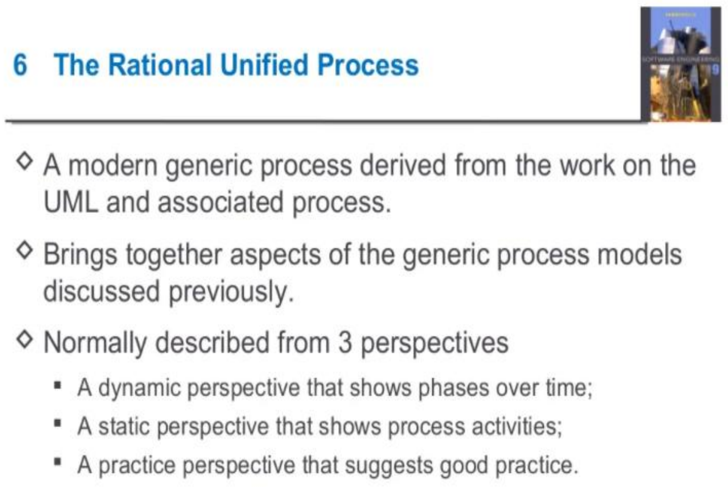
- ✧ State model describes the sequences of operations that occur in response to external stimuli.
- ✧ The state model consists of multiple state diagrams, one for each class with temporal behavior that is important to an application.
- ✧ The state diagram is a standard computer science concept that relates events and states.
- ✧ Events represent external stimuli and states represent values objects
- ✧ **The basic elements of state diagrams are**
    - ✧ ***Events*** **– An event is an occurrence at a point in time**
    - ✧ ***states*** **– the state in which the object finds itself at any moment**
    - ✧ ***transitions*** **– take the object from one state to another**
    - ✧ ***actions*** **– take place as a result of a transition**



6	a. Explain the rational unified process	6	CO5	L3

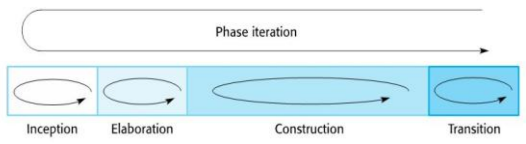
Solution:

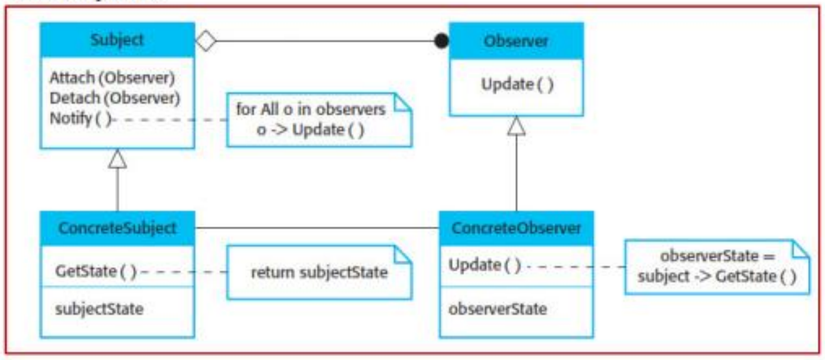
Its goal is to deliver a high quality product that the customer actually wants.

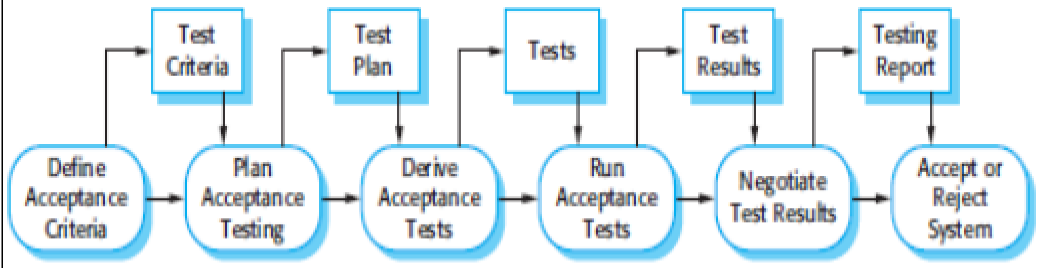


6 The Rational Unified Process

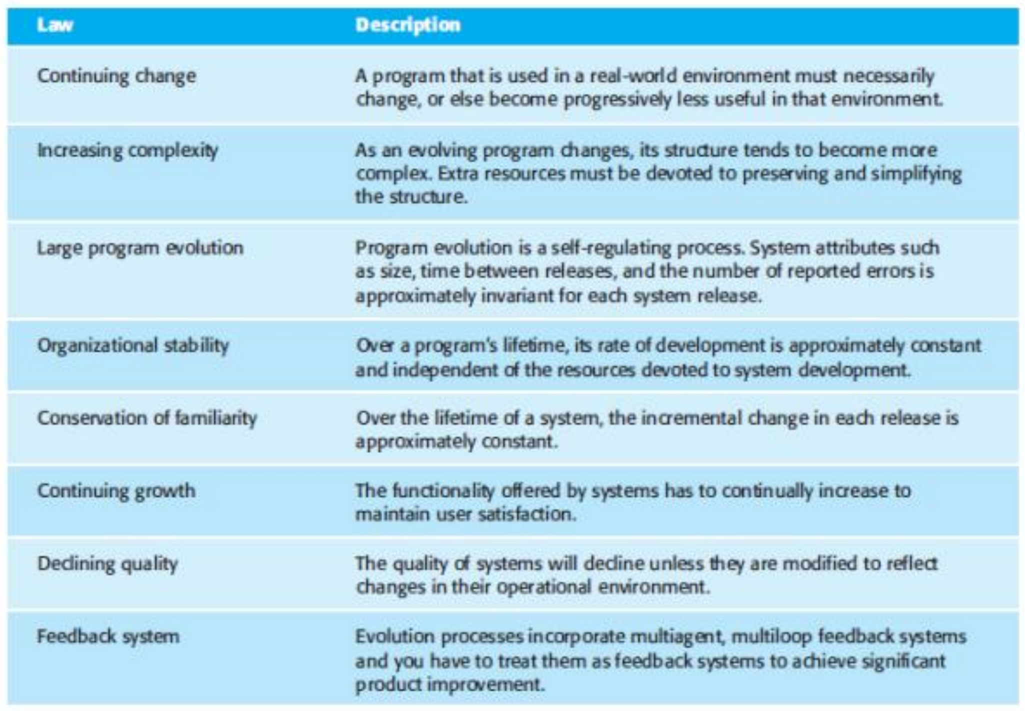
- ◇ A modern generic process derived from the work on the UML and associated process.
- ◇ Brings together aspects of the generic process models discussed previously.
- ◇ Normally described from 3 perspectives
    - ▪ A dynamic perspective that shows phases over time;
    - ▪ A static perspective that shows process activities;
    - ▪ A practice perspective that suggests good practice.

	Phases in the Rational Unified Process  Phase iteration Inception Elaboration Construction Transition Chapter 2 Software Processes 2 . Inception Establish the business case for the system. o You should identify all external entities (people and systems) that will interact with the system and define these interactions. You then use this information to assess the contribution that the system makes to the business. If this contribution is minor, then the project may be cancelled after this phase. Elaboration Develop an understanding of the problem domain and the system architecture. o The goals of the elaboration phase are to develop an understanding of the problem domain, establish an architectural framework for the system, develop the project plan, and identify key project risks. On completion of this phase you should have a requirements model for the system, which may be a set of UML use-cases, an architectural description, and a development plan for the software. Construction System design, programming and testing. o Testing. Parts of the system are developed in parallel and integrated during this phase. On completion of this phase, you should have a working software system and associated documentation that is ready for delivery to users. Transition Deploy the system in its operating environment. o The final phase of the RUP is concerned with moving the system from the development community to the user community and making it work in a real environment. This is something that is ignored in most software process models but is, in fact, an expensive and sometimes problematic activity. On completion of this phase, you should have a documented software system that is working correctly in its operational environment			

	b. Explain design pattern with UML model of the observer model. **Solution:** The four essential elements of design patterns were defined by the 'Gang of Four' in their patterns A name that is a meaningful reference to the pattern. A description of the problem area that explains when the pattern may be applied. A solution description of the parts of the design solution, their relationships, and Their responsibilities. This is not a concrete design description. It is a template for a design solution that can be instantiated in different ways. This is often expressed graphically and shows the relationships between the objects and object classes in the solution. A statement of the consequences—the results and trade-offs—of applying the pattern. This can help designers understand whether or not a pattern can be used in a particular situation. ***A UML model of the Observer pattern***  Subject Attach (Observer) Detach (Observer) Notify ( ) for All o in observers o -> Update ( ) Observer Update ( ) ConcreteSubject GetState ( ) subjectState return subjectState ConcreteObserver Update ( ) observerState observerState = subject -> GetState ( )	8	CO5	L2
	c. What are the different implementation issues of software engineering? Solution: Implementation issues Focus here is not on programming, although this is obviously important, but on other implementation issues that are often not covered in programming texts: Reuse Most modern software is constructed by reusing existing components or systems. When you are developing software, you should make as much use as possible of existing code. Configuration management during the development process, you have to keep track of the many different versions of each software component in a configuration management system. Host-target development Production software does not usually execute on the same computer as the software development environment. Rather, you develop it on one computer (the host system) and execute it on a separate computer (the target system). Reuse From the 1960s to the 1990s, most new software was developed from scratch, by writing all code in a high-level programming language. The only significant reuse or software was the reuse of functions and objects in programming language libraries. Costs and schedule pressure mean that this approach became increasingly unviable, especially for commercial and Internetbased systems. An approach to development based around the reuse of existing software emerged and is now generally used for business and scientific software.	6		
7	a. What are the two distinct goals of Software testing? **Solution:** The testing process has two distinct goals: 1. To demonstrate to the developer and the customer that the software meets its requirements. For custom software, this means that there should be at least one test for every requirement in the requirements document. For generic software products, it means that there should be tests for all of the system	5	CO2	L2

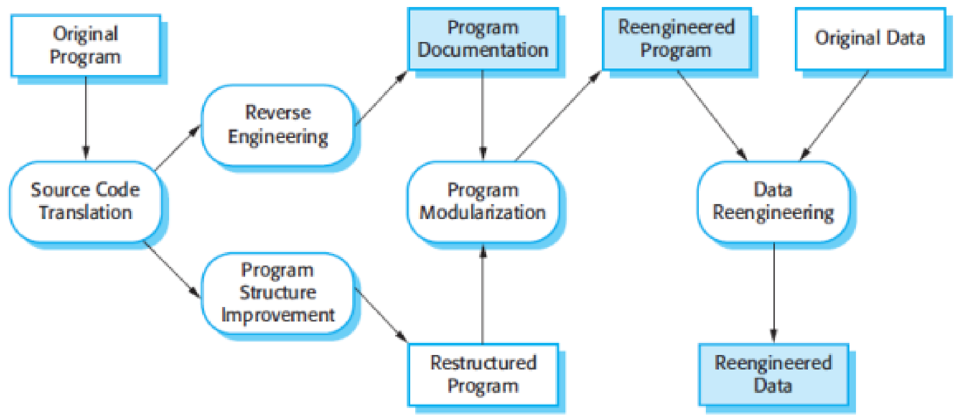
	features, plus combinations of these features, that will be incorporated in the product release. 2. To discover situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification. Defect testing is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations, and data corruption.			
	b. Explain the three different types of testing carried out during software development? **Solution:** During development, testing may be carried out at three levels of granularity: 1. Unit testing, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods. 2. Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces. 3. System testing, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions. • Development testing is primarily a defect testing process, where the aim of testing is to discover bugs in the software. It is therefore usually interleaved with debugging— the process of locating problems with the code and changing the program to fix these problems.	10	CO4	L2
	c. What are the different types of user testing? With neat diagram, explain the six stages of acceptance testing Solution: There are three different types of user testing: 1. Alpha testing, where users of the software work with the development team to test the software at the developer's site. 2. Beta testing, where a release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers. 3. Acceptance testing, where customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment.  Test Criteria Test Plan Tests Test Results Testing Report Define Acceptance Criteria Plan Acceptance Testing Derive Acceptance Tests Run Acceptance Tests Negotiate Test Results Accept or Reject System Define acceptance criteria This stage should, ideally, take place early in the process before the contract for the system is signed. The acceptance criteria should be part of the system contract and be agreed between the customer and the developer.			

	Plan acceptance testing This involves deciding on the resources, time, and budget for acceptance testing and establishing a testing schedule. The acceptance test plan should also discuss the required coverage of the requirements and the order in which system features are tested. Derive acceptance tests Once acceptance criteria have been established, tests have to be designed to check whether or not a system is acceptable. Acceptance tests should aim to test both the functional and non-functional Run acceptance tests The agreed acceptance tests are executed on the system. Ideally, this should take place in the actual environment where the system will be used, but this may be disruptive and impractical. Therefore, a user testing environment may have to be set up to run these tests. Negotiate test results It is very unlikely that all of the defined acceptance tests will pass and that there will be no problems with the system. If this is the case, then acceptance testing is complete and the system can be handed over Reject/accept system This stage involves a meeting between the developers and the customer to decide on whether or not the system should be accepted			
8	a. Write the Lemman's law of program dynamic evolution.	6	CO5	L2



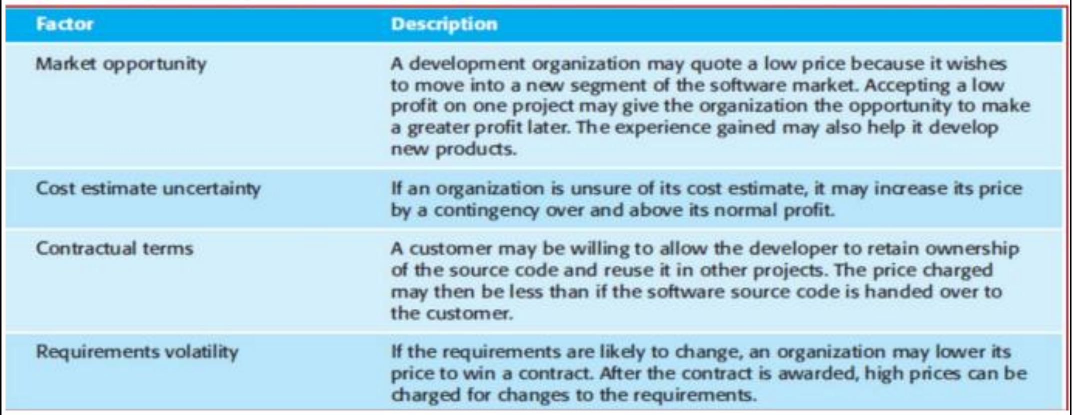
Law	Description
Continuing change	A program that is used in a real-world environment must necessarily change, or else become progressively less useful in that environment.
Increasing complexity	As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.
Large program evolution	Program evolution is a self-regulating process. System attributes such as size, time between releases, and the number of reported errors is approximately invariant for each system release.
Organizational stability	Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.
Conservation of familiarity	Over the lifetime of a system, the incremental change in each release is approximately constant.
Continuing growth	The functionality offered by systems has to continually increase to maintain user satisfaction.
Declining quality	The quality of systems will decline unless they are modified to reflect changes in their operational environment.
Feedback system	Evolution processes incorporate multiagent, multiloop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.

	b. With neat diagram, explain the software reengineering process activities **Solutions:**	8	CO5	L2

	 Original Program Source Code Translation Reverse Engineering Program Structure Improvement Program Documentation Program Modularization Restructured Program Reengineered Program Original Data Data Reengineering Reengineered Data **Fig: The reengineering process** The activities involved in reengineering process are as follows 1. Source code translation ¬ Using a translation tool, the program is converted from an old programming language to a more modern version of the same language or to a different language. 2. Reverse engineering ¬ The program is analyzed and information extracted from it. ¬ This helps to document its organization and functionality. ¬ This process is usually completely automated. 3. Program structure improvement ¬ The control structure of the program is analyzed and modified to make it easier to read and understand. ¬ This can be partially automated but some manual intervention is usually required. 4. Program modularization ¬ Related parts of the program are grouped together. ¬ Where appropriate, redundancy is removed. ¬ This is a manual process. 5. Data reengineering ¬ The data processed by the program is changed to reflect program changes. ¬ This may mean redefining database schemas, converting existing databases to the new structure, clean up the data, finding and correcting mistakes, removing duplicate records, etc. ¬ Tools are available to support data reengineering.			
	c. What are the four strategic options for legacy systems There are four strategic options: 1. Scrap the system completely This option should be chosen when the system is not making an effective contribution to business processes. 2. Leave the system unchanged and continue with regular maintenance This option should be chosen when the system is still required but is fairly stable and the system users make relatively few change requests. 3. Reengineer the system to improve its maintainability This option should be chosen when the system quality has been degraded by change and where a new change to the system is still being proposed. 4. Replace all or part of the system with a new system This option should be chosen when factors, such as new hardware, mean that the old system cannot continue in operation or where off-the-shelf systems would allow the new system to be developed at a reasonable cost.	6		

9	What are the factors affecting the pricing of software product?	4	CO4	L2

Solutions:

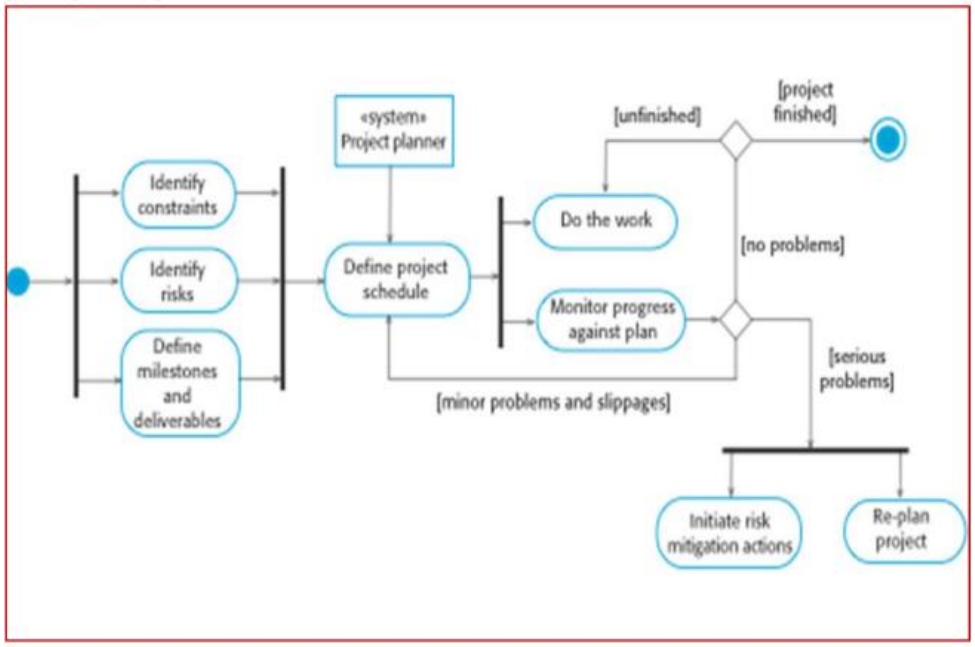


Factor	Description
Market opportunity	A development organization may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the organization the opportunity to make a greater profit later. The experience gained may also help it develop new products.
Cost estimate uncertainty	If an organization is unsure of its cost estimate, it may increase its price by a contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organization may lower its price to win a contract. After the contract is awarded, high prices can be charged for changes to the requirements.

	b. With neat diagram, explain the project planning process	06	CO4	L2

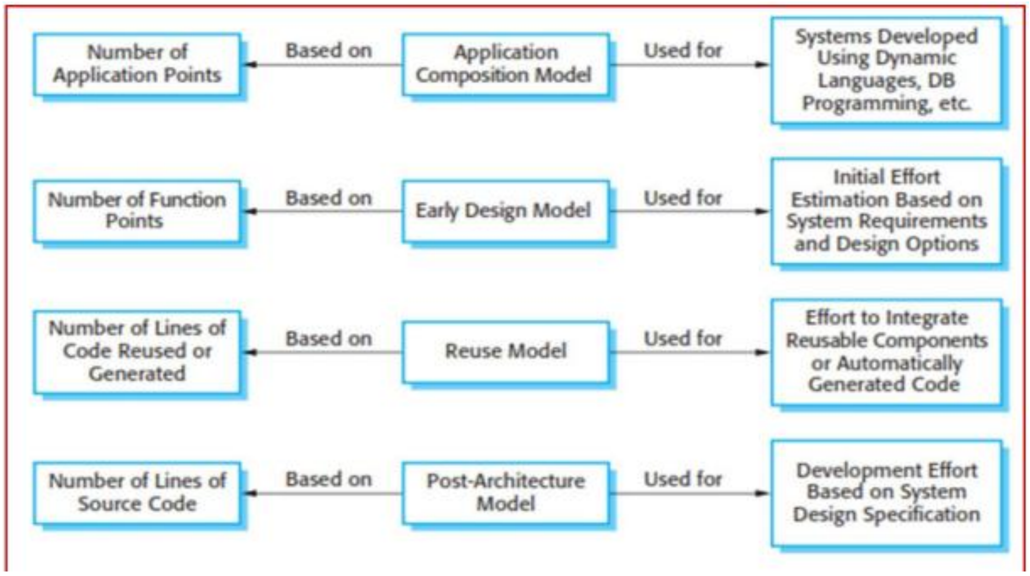
Solutions:

The planning process Project planning is an iterative process that starts when you create an initial project plan during the project startup phase. Plan changes are inevitable. As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule and risk changes. Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be re-planned.



	c. With neat diagram, explain the COCOMO-II estimation model.	10		

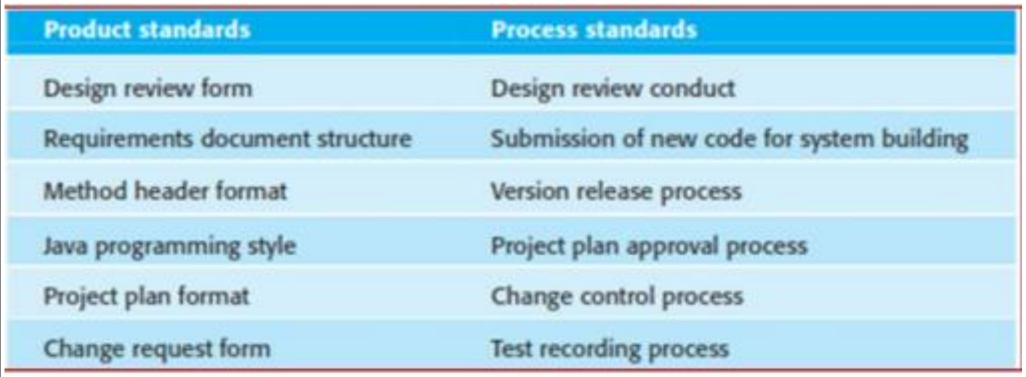
The COCOMO 2 model An empirical model based on project experience. Well-documented, 'independent' model which is not tied to a specific software vendor. Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2. COCOMO 2 takes into account different approaches to software development, reuse, etc. COCOMO 2 models COCOMO 2 incorporates a range of sub-models that produce increasingly detailed software estimates. The sub-models in COCOMO 2 are: Application composition model. Used when software is composed from existing parts. Early design model. Used when requirements are available but design has not yet started. Reuse model. Used to compute the effort of integrating reusable components. Post-architecture model. Used once the system architecture has been designed and more information about the system is available.


Number of Application Points
Based on
Application Composition Model
Used for
Systems Developed Using Dynamic Languages, DB Programming, etc.
Number of Function Points
Based on
Early Design Model
Used for
Initial Effort Estimation Based on System Requirements and Design Options
Number of Lines of Code Reused or Generated
Based on
Reuse Model
Used for
Effort to Integrate Reusable Components or Automatically Generated Code
Number of Lines of Source Code
Based on
Post-Architecture Model
Used for
Development Effort Based on System Design Specification

10	a. Explain the product and process standards in software quality management.	06	CO5	L2

Solutions:

Software standards Standards define the required attributes of a product or process. They play an important role in quality management. Standards may be international, national, organizational or project standards. Product standards define characteristics that all software components should exhibit e.g. a common programming style. Process standards define how the software process should be enacted. Importance of standards Encapsulation of best practice- avoids repetition of past mistakes. They are a framework for defining what quality means in a particular setting i.e. that organization's view of quality. They provide continuity - new staff can understand the organisation by understanding the standards that are used.

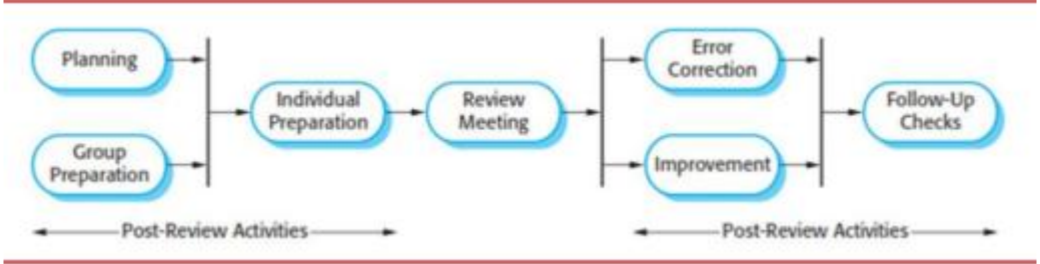


Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

	b. Explain three phases of software review process.	8	CO1	L2

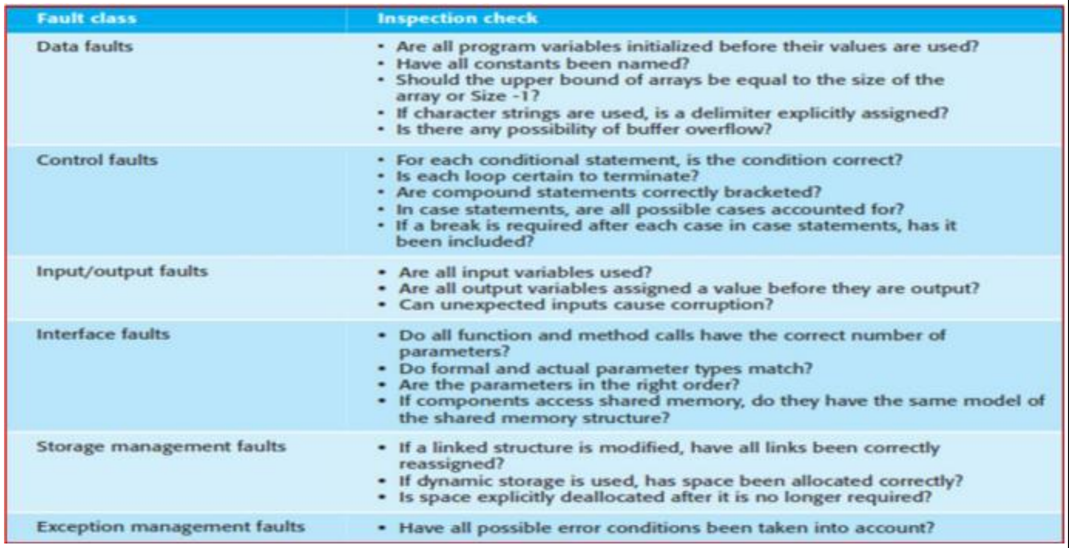
Solutions:

The software review process Pre-review activities: Pre-review activities are concerned with review planning and review preparation. Review planning involves setting up a review team, arranging a time and place for the review, and distributing the documents to be reviewed. During review preparation, the team may meet to get an overview of the software to be reviewed. Individual review team members read and understand the software or documents and relevant standards.



The review meeting During the review meeting, an author of the document or program being reviewed should 'walk through' the document with the review team. Post-review activities After a review meeting has finished, the issues and problems raised during the review must be addressed. This may involve fixing software bugs, refactoring software so that it conforms to quality standards, or rewriting documents.

	c. Explain the various inspection checks in the program inspection.	8		



Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size -1? • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?
Interface faults	• Do all function and method calls have the correct number of parameters? • Do formal and actual parameter types match? • Are the parameters in the right order? • If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	• If a linked structure is modified, have all links been correctly reassigned? • If dynamic storage is used, has space been allocated correctly? • Is space explicitly deallocated after it is no longer required?
Exception management faults	• Have all possible error conditions been taken into account?