RAMAIAH
Institute of Technology

Dept. of Computer Science and Engineering (AI & ML) & Computer Science and Engineering (Cyber Security)

Programme: B E – Computer Science and Engineering (AI&ML) & Computer Science and Engineering (Cyber Security)

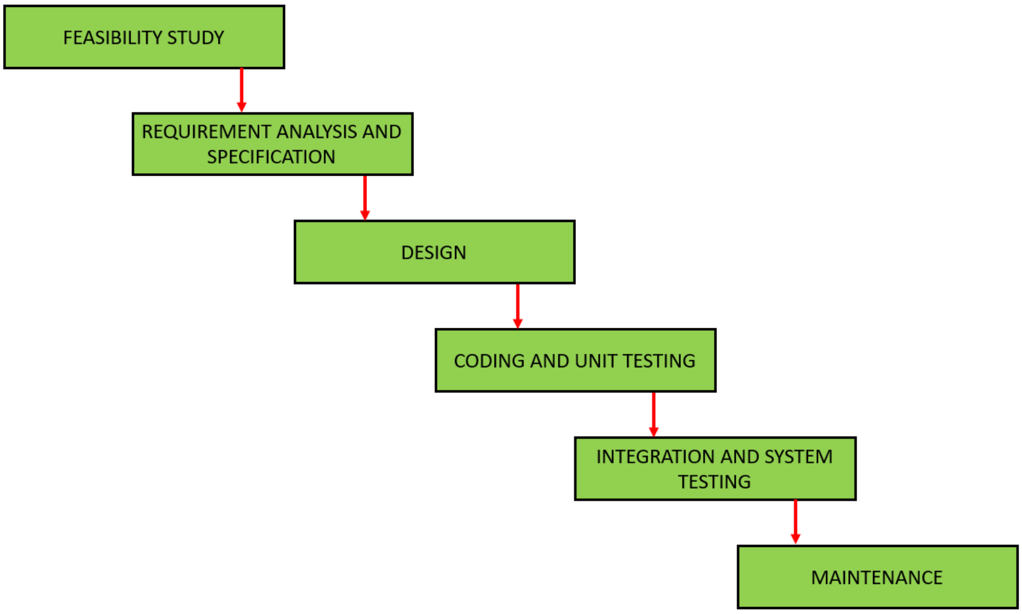
Internal Assessment – I

TERM : **4/10/2023 to 27/1/2024**	COURSE NAME: Software Engineering
DATE : 01/12/2023 TIME:3.30-4.30PM	COURSE CODE : CI54/CY54
MAX MARKS: **30**	PORTIONS : 2.5

📵 Mobile Phones are banned

Instructions to Candidates: **Answer any TWO full questions**. Marks: **15x2=30**

Q. NO	**Questions**	**BL (L1 to L6)***	**CO**	**Marks**
1.a	Illustrate the working of waterfall model with its advantages and disadvantages.	**L2**	**CO1**	**5**
b	The Medical Health Care-Patient Monitoring software system(MHC-PMC), which is used to maintain information about patients receiving treatment for mental health problems, do the following.. 1. Discus the functional requirement for the (MHC-PMS). 2. Explain three types of non-functional requirement and discover the non-functional requirements using (MHC-PMS).	**L3**	**CO2**	**5**
c	Discus the transaction processing system and information system explain with example using architectural design diagram.	**L2**	**CO3**	**5**
2.a	Why incremental process development is the most effective approach for developing business software systems discuss using a process diagram. Why is this model less appropriate for real-time systems engineering?	**L2**	**CO1**	**5**
b	Describe the various stages within requirements engineering process through a well-organized diagram.	**L2**	**CO2**	**5**
c	Discuss different architectural view. Assume that you developing a E-commerce website for online medicine delivery application, construct and explain the architecture using MVC pattern.	**L2**	**CO3**	**5**
3.a	With a suitable diagram describe the extreme programming release cycle and discus the Extreme programming practices.	**L2**	**CO1**	**5**
b	Discus the requirement elicitation and analysis process. Design the use case diagram for MHC-PMC system and explain it.	**L2**	**CO2**	**5**
c	What is an architectural Pattern? Illustrate with neat sketches, the repository and layered architectural design and its advantages and disadvantages.	**L2**	**CO3**	**5**

Sl. NO	SCHEMEANDSOLITIONS	Marks
1.a	**Illustrate the working of waterfall model with its advantages and disadvantages.** Diagram of Waterfall model- 1M  FEASIBILITY STUDY REQUIREMENT ANALYSIS AND SPECIFICATION DESIGN CODING AND UNIT TESTING INTEGRATION AND SYSTEM TESTING MAINTENANCE Phases explanation of Waterfall Model explanation in Software Engineering- 2M ○ Requirements Gathering and Analysis ○ System Design ○ Implementation ○ Testing ○ Deployment ○ Maintenance Examples of Waterfall Model Advantages of waterfall model- 1M Disadvantages of waterfall model- 1M	5M

b	**The Medical Health Care-Patient Monitoring software system(MHC-PMC), which is used to maintain information about patients receiving treatment for mental health problems, do the following..** 1. Discus the functional requirement for the (MHC-PMS).2M Here are examples of functional requirements for the MHC-PMS system, used to maintain information about patients receiving treatment for mental health problems: 1. A user shall be able to search the appointments lists for all clinics. 2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day. 3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number. 2. Explain three types of non-functional requirement and discover the non-functional requirements using (MHC-PMS) 2M 1**. Product requirements** These requirements specify or constrain the behavior of the software. Examples include performance requirements on how fast the system must execute and how much memory it requires, reliability requirements that set out the acceptable failure rate, security requirements, and usability requirements. 2. **Organizational requirements** These requirements are broad system requirements derived from policies and procedures in the customer's and developer's organization. Examples include operational process requirements that define how the system will be used, development process requirements that specify the programming language, the development environment or process standards to be used, and environmental requirements that specify the operating environment of the system. 3**. External requirements** This broad heading covers all requirements that are derived from factors external to the system and its development process. These may include regulatory requirements that set out what must be done for the system to be approved for use by a regulator, such as a central bank; legislative requirements that must be followed to ensure that the system operates within the law; and ethical requirements that ensure that the system will be acceptable to its users and the general public. Non-functional the non-functional requirements using (MHC-PMS) 1M PRODUCT REQUIREMENT The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day. ORGANIZATIONAL REQUIREMENT Users of the MHC-PMS system shall authenticate themselves using their health authority identity card. EXTERNAL REQUIREMENT The system shall implement patient privacy provisions as set out in HStan-03-2006-priv	**5M**

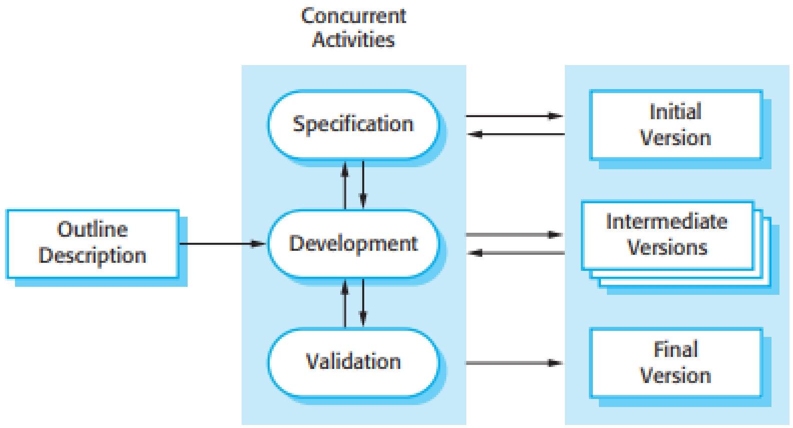
.

c	**Discus the transaction processing system and information system explain with example using architectural design diagram.** Diagram with explanation of transaction processing system using architectural design with any example- 2.5M Diagram with explanation of information processing system using architectural design with any example- 2.5M	**5M**

2.a

Why incremental process development is the most effective approach for developing business software systems discuss using a process diagram. Why is this model less appropriate for real-time systems engineering?

Diagram of incremental model- 1M



Phases of incremental model- 2M

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. **Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

Business software systems usually complex, software intensive, and frequently being changes when business goals or processes are changed, it allows for quicker feedback and iteration, enabling teams to make adjustments and improvements to the software as they go. So incremental development is better. 1M

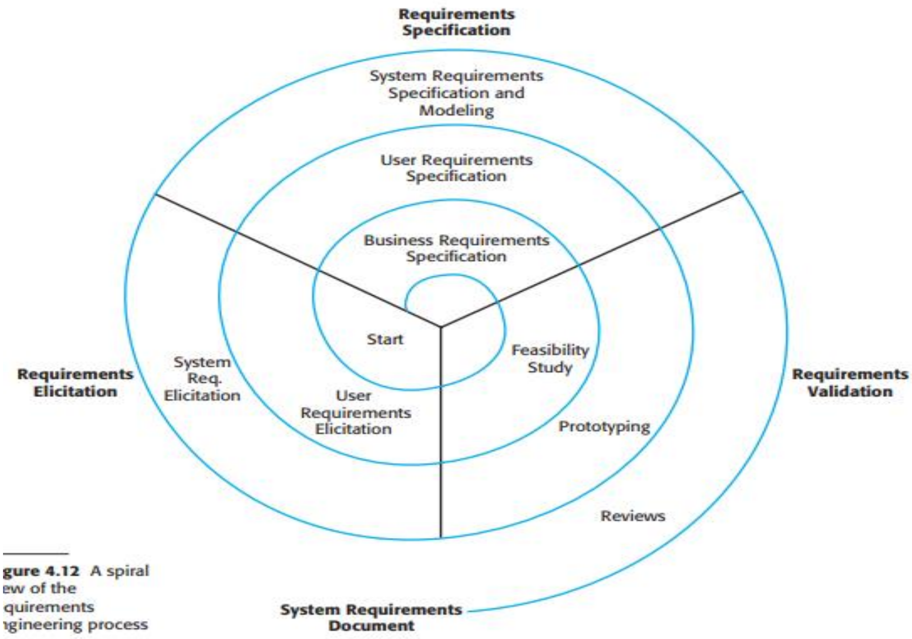
Real-time systems usually involve many hardware components which are not easy to change and cannot be incremental. Also real-time systems usually safety critical which needed be built based on well planned process. 1M

5M

.

b **Describe the various stages within requirements engineering process through a well-organized diagram.**

Diagram-1M


Requirements Specification
System Requirements Specification and Modeling
User Requirements Specification
Business Requirements Specification
Start
Feasibility Study
Requirements Elicitation
System Req. Elicitation
User Requirements Elicitation
Prototyping
Requirements Validation
Reviews
System Requirements Document

gure 4.12 A spiral ew of the quirements ngineering process

Process Phases- 4M

Requirements Elicitation

This is the process of gathering information about the needs and expectations of stakeholders for the software system. This step involves interviews, surveys, focus groups, and other techniques to gather information from stakeholders.

Requirements Analysis

This step involves analyzing the information gathered in the requirements elicitation step to identify the high-level goals and objectives of the software system. It also involves identifying any constraints or limitations that may affect the development of the software system.

Requirements Specification

This step involves documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner. This step also involves prioritizing and grouping the requirements into manageable chunks.

Requirements Validation

This step involves checking that the requirements are complete, consistent, and accurate. It also involves checking that the requirements are testable and that they meet the needs and expectations of stakeholders.

Requirements Management

This step involves managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant.

Requirement Engineering

The Requirements Engineering process is a critical step in the software development life cycle as it helps to ensure that the software system being developed meets the needs and expectations of stakeholders, and that it is developed on time, within budget, and to the required quality.

5M

c

Discuss different architectural view. Assume that you developing a E-commerce website for online medicine delivery application, construct and explain the architecture using MVC pattern.

The views that he suggests are: 2M

1. **A logical view**, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.
2. **A process view**, which shows how, at run-time, the system is composed of interacting processes. This view is useful for making judgments about nonfunctional system characteristics such as performance and availability.
3. **A development view**, which shows how the software is decomposed for development, that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.
4. **A physical view**, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

MVC model diagram 1M explanation 2M

1. Model:

- Represents the database layer where information about medicines, users, orders, and other relevant data is stored. Object-Relational Mapping Acts as a bridge between the application code and the database, providing a way to interact with the database using objects. The Model notifies the Controller about changes in data, ensuring that the Controller is aware of the current state.

2. View:

- UI/UX Components Includes the user interface elements, such as web pages or mobile app screens, where users interact with the application. Templates: Dynamically generate HTML or other UI elements based on data received from the Controller. Users interact with the UI, making requests like searching for medicines, adding items to the cart, or placing an order. The updated View is presented to the user, completing the cycle.

3. Controller:

- Business Logic contains the core logic of the application, responsible for processing user requests, interacting with the Model, and updating the View. Routing: Maps URLs to specific controllers and actions, determining how different requests are handled. API Endpoints: Defines endpoints for communication with mobile apps or third-party services.

  The Controller receives these user requests and processes them, invoking the necessary business logic.

  The Controller interacts with the Model to perform actions like updating the database, retrieving information, and applying business rules.

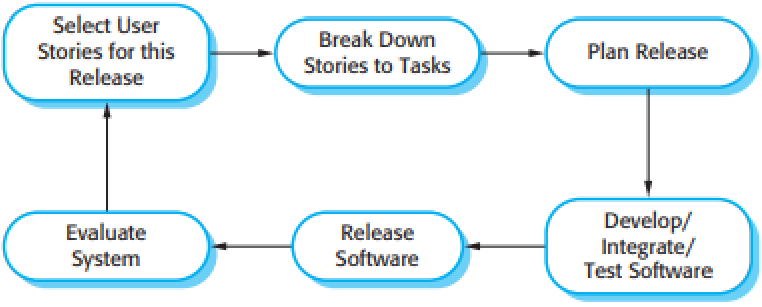
  The Controller updates the View, ensuring that the user interface reflects the current state of the application.

5M

3.a

With a suitable diagram describe the extreme programming release cycle and discus the Extreme programming practices.

Diagram-1M


Select User Stories for this Release
Break Down Stories to Tasks
Plan Release
Develop/ Integrate/ Test Software
Release Software
Evaluate System

Phases explanation- 2M

Planning

This is the first stage of the Extreme Programming development life cycle. Its main task is to set goals of the entire project and certain iterative cycles. At this stage the team meets with the customer and asks him on all aspects of the future software. The customer formulates his vision of the product is user stories. The developers estimate and prioritize them in the release plan. After that the work on turning them into tasks begins.

Designing

At this stage of the project the team must define the main features of the future code. The main thing is to create a simple design, because simplicity is one of the main principles of XP methodology. Extreme Programming developers often share responsibilities at the stage of designing. Each developer is responsible for the design of a certain part of the code.

Coding

Extreme Programming developers believe that a good code must be simple. That is why they refactor it constantly. The procedure of refactoring allows them to simplify the code or its parts without affecting the functionality of the final product.

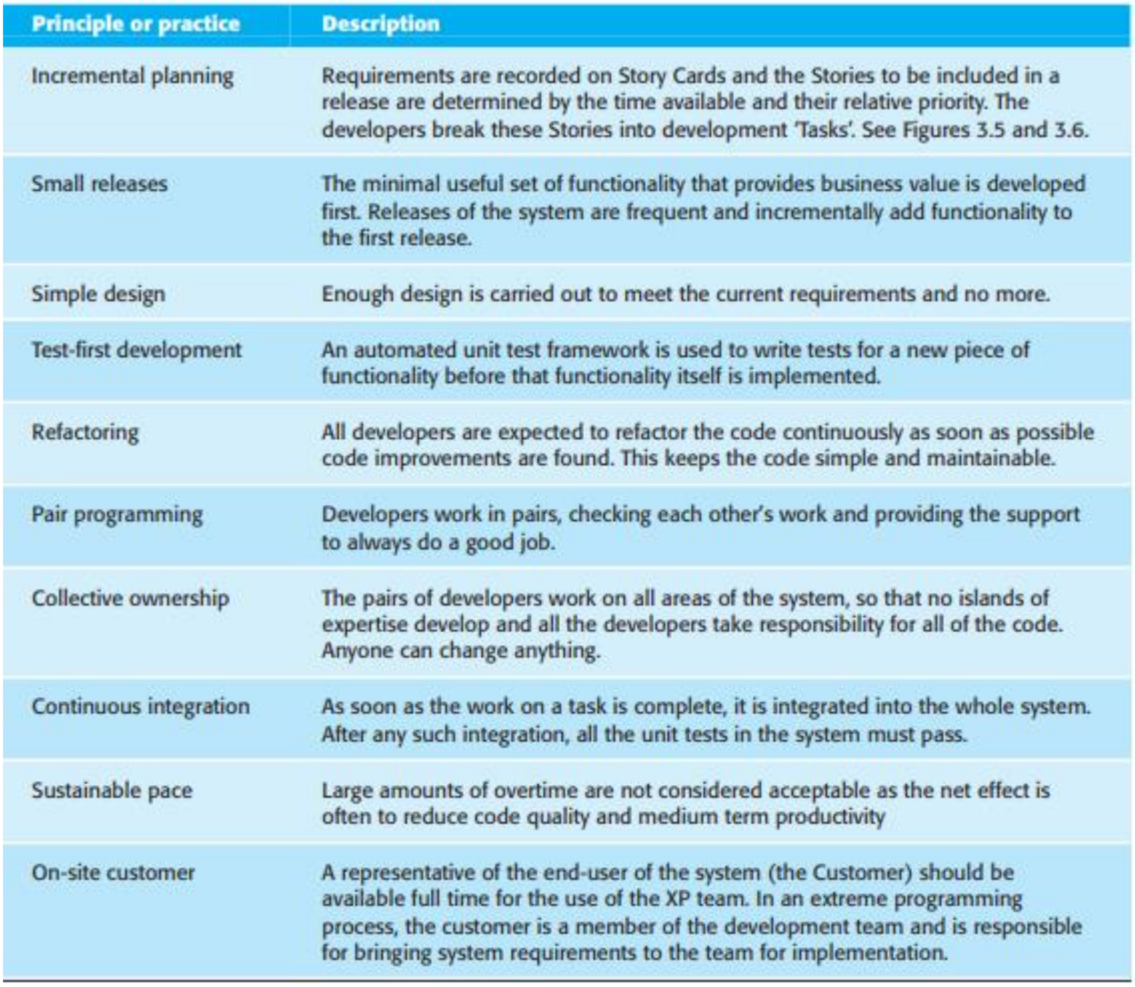
Testing

In Extreme Programming the procedure of testing is usually conducted not after the final or intermediate product is done, but along with the procedure of code writing.

Listening

At the final stage of the life cycle the XP team must get a feedback from the customer. He is the only person who estimates the final and intermediate products.

5M



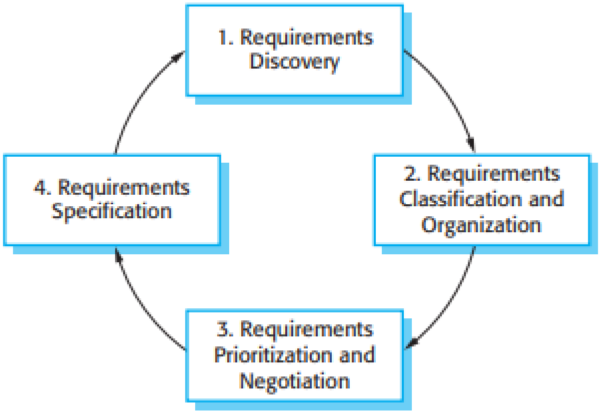
Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

5M

b

Discus the requirement elicitation and analysis process. Design the use case diagram for MHC-PMC system and explain it.

Diagram-1M


1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

The process activities are: 3M

1. Requirements discovery This is the process of interacting with stakeholders of the system to discover their requirements. Domain requirements from stakeholders and documentation are also discovered during this activity. There are several complementary techniques that can be used for requirements discovery, which I discuss later in this section.

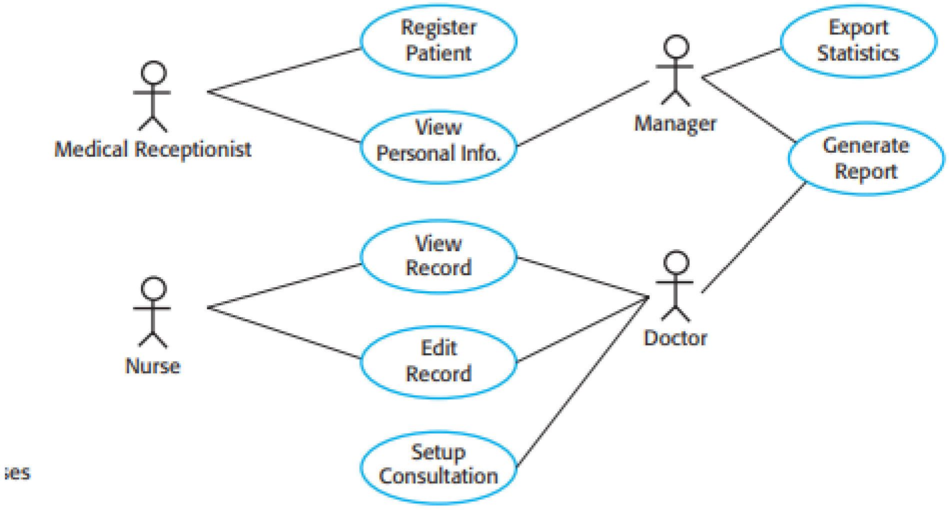
2. Requirements classification and organization This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters. The most common way of grouping requirements is to use a model of the system architecture to identify sub-systems and to associate requirements with each sub-system. In practice, requirements engineering and architectural design cannot be completely separate activities.

3. Requirements prioritization and negotiation Inevitably, when multiple stakeholders are involved, requirements will conflict. This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation. Usually, stakeholders have to meet to resolve differences and agree on compromise requirements.

4. Requirements specification The requirements are documented and input into the next round of the spiral. Formal or informal requirements documents may be produced,

Figure shows that requirements elicitation and analysis is an iterative process with continual feedback from each activity to other activities. The process cycle starts with requirements discovery and ends with the requirements documentation. The analyst’s understanding of the requirements improves with each round of the cycle. The cycle ends when the requirements document is complete.

Design the use case diagram for MHC-PMC system and explain it. 1M



5M

c

What is an architectural Pattern? Illustrate with neat sketches, the repository and layered architectural design and its advantages and disadvantages.

What is architectural pattern? 1M

An architectural pattern is a general, reusable solution to a recurring problem in software architecture. It provides a set of guidelines and best practices for organizing the structure and behavior of software systems. Architectural patterns help define the high-level structure of an application, guiding developers in making design decisions to achieve specific qualities like modularity, scalability, maintainability, and reusability.

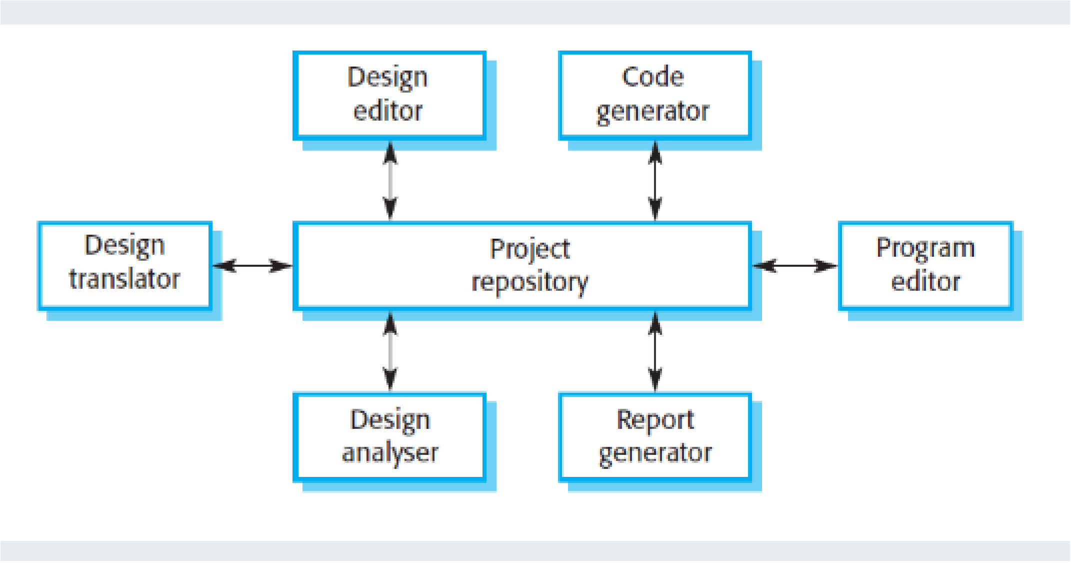
Illustrate with neat sketches, the repository and layered architectural design and its advantages and disadvantages. 4M

Repository model is a system that will allow interfacing sub-systems to share the same data. Sub-system must exchange data so that they can work together effectively. This may be done in two ways:

1. All shared data is held in a central database that can be accessed by all subsystems. It is called repository model.

2. Each sub-system maintains its own database. Data is interchanged with other sub-systems by passing messages to them.

Example: CASE Toolset



Advantages:

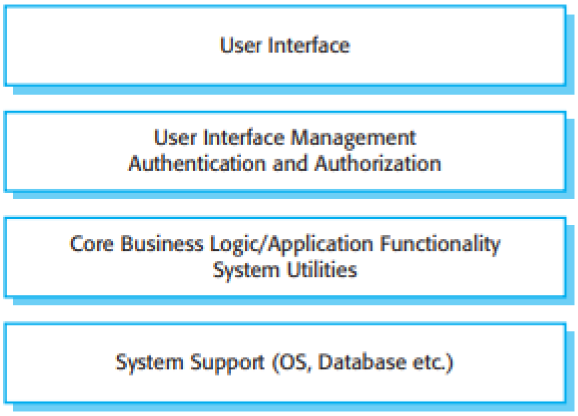
- It is an efficient way to share large amounts of data. There is no need to transmit data explicitly from one sub-system to another.
- Sub-systems that produce data need not be concerned with how that data is used by other subsystems.
- Activities such as backup, security, access control and recovery from error are centralized.

Disadvantages:

- It is a compromise between the specific needs of each tool. Performance may be adversely affected by this compromise.
- Evolution may be difficult as a large volume of information is generated according to an agreed data model.

- Different sub-systems may have different requirements for security, recovery and backup policies. The repository model forces the same policy on all subsystems.

Layered Architecture design


User Interface
User Interface Management
Authentication and Authorization
Core Business Logic/Application Functionality
System Utilities
System Support (OS, Database etc.)

Organizes the system into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system.

Used when building new facilities on top of existing systems; when the development is spread across several teams with each team responsibility for a layer of functionality; when there is a requirement for multi-level security.

Advantages

- Allows replacement of entire layers so long as the interface is maintained. Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.
- The framework is simple and easy to learn and implement.

- There is reduced dependency because the function of each layer is separate from the other layers.
- Testing is easier because of the separated components, each component can be tested individually.
- Cost overheads are fairly low.

Disadvantages

- In practice, providing a clean separation between layers is often difficult and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it. Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.
- Scalability is difficult because the structure of the framework does not allow for growth.
- They can be difficult to maintain. A change in a single layer can affect the entire system because it operates as a single unit.
- There is interdependence between layers since a layer depends on the layer above it to receive data.
- Parallel processing is not possible.