USN | | | | | | | | | |

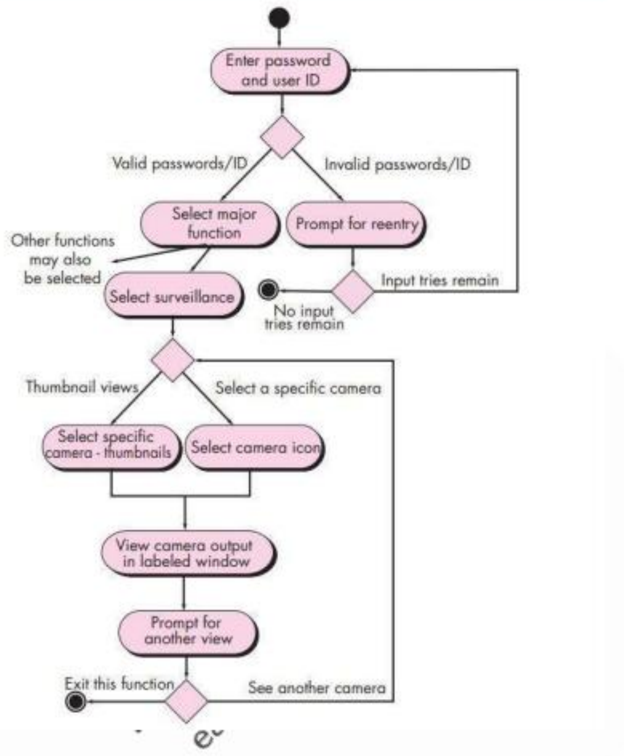
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

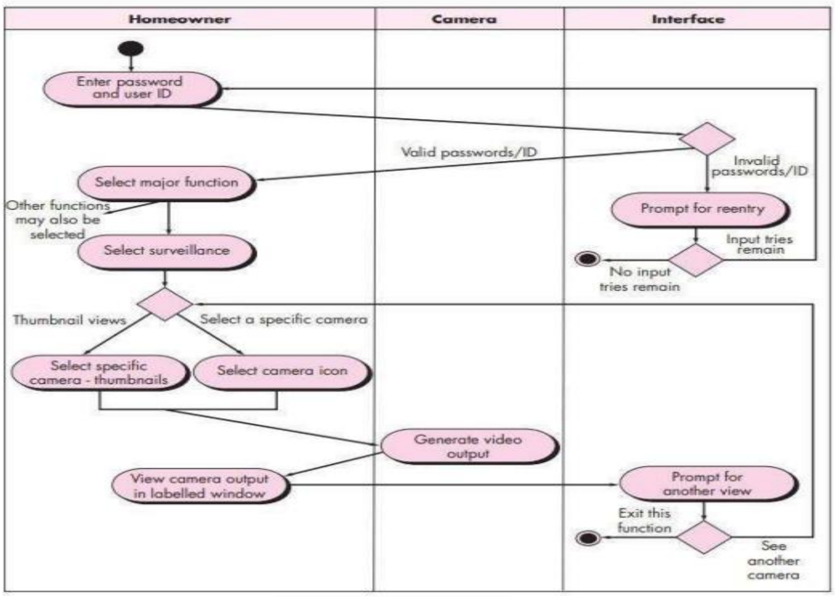
Internal Assessment Test 1 Answer scheme & Solutions – October 2024

Sub:	**Software Engineering**					Sub Code:	BCS501	Branch:	**AInDS**
Date:	**16/10/2024**	Duration:	**90 minutes**	Max Marks:	**50**	Sem	**VII**		**OBE**

		Answer any FIVE Questions	**MARKS**	**CO**	**RBT**
1		Draw activity diagram and Swimlane diagram for access camera surveillance via the internet.	10	CO1	L2

Figure 6.5 Activity diagram for Access camera surveillance via the Internet—display camera views function.





			MARKS	**CO**	**RBT**
2	a	Define software engineering? Briefly discuss the attributes of good software. **Software Engineering** is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.	6	CO1	L1

Software has characteristics that are considerably different than those of hardware:

1) Software is developed or engineered; it is not manufactured in the Classical Sense.

- Although some similarities exist between software development and hardware manufacturing, the two activities are fundamentally different.
- In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce **quality problems** that are nonexistent or easily corrected for software.
- Both the activities are dependent on people, but the relationship between people is totally varying. These two activities require the construction of a "product" but the approaches are different.
- Software costs are concentrated in engineering which means that software projects cannot be managed as if they were manufacturing.

2) Software doesn't "Wear Out"

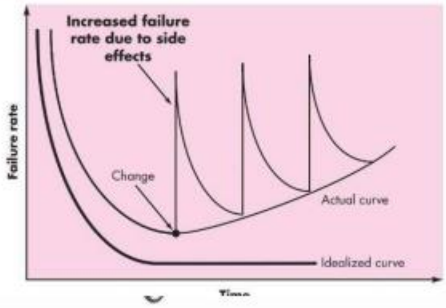
- In early stage of hardware development process the failure rate is very high due to manufacturing defects, but after correcting defects failure rate gets reduced.
- Hardware components suffer from the growing effects of many other environmental factors. Stated simply, the hardware begins to **wear out.**
- Software is not susceptible to the environmental maladies (extreme temperature, dusts and vibrations) that cause hardware to wear out [Fig:1.1]

The following figure shows the relationship between failure rate and time.

FIGURE 1.1
Failure curve for hardware

- When a hardware component wears out, it is replaced by a spare part. There are no software spare parts.
- Every software failure indicates an error in design or in the process through which the design was translated into machine-executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance. However, the implication is clear—the software doesn't wear out. But it does **deteriorate** (frequent changes in requirement) **[Fig:1.2].**

FIGURE 1.2
Failure curves for software



3) Most Software is custom-built rather than being assembled from components:

- A software part should be planned and carried out with the goal that it tends to be reused in various projects (algorithms and data structures).
- Today software industry is trying to make library of reusable components E.g. Software GUI is built using the reusable components such as message windows, pull down menu and many more such components.
- In the hardware world, component reuse is a natural part of the engineering process.

	b	Briefly explain the software engineering ethics? The dictionary defines the word **principle** as **"an important underlying law or assumption** required in a system of thought." David Hooker has Proposed seven principles that focus on software Engineering practice. **The First Principle: The Reason It All Exists** A software system exists for one reason: to provide value to its users. **The Second Principle: KISS (Keep It Simple, Stupid!)** Software design is not a haphazard process. There are many factors to consider in any design effort. All design should be as simple as possible, but no simpler. **The Third Principle: Maintain the Vision** A clear vision is essential to the success of a software project. Without one, a project almost unfailingly ends up being "of two [or more] minds" about itself. **The Fourth Principle: What You Produce, Others Will Consume** Always specify, design, and implement knowing someone else will have to understand what you are doing. **The Fifth Principle: Be Open to the Future** A system with a long lifetime has more value. Never design yourself into a corner. Before beginning a software project, be sure the software has a business purpose and that users perceive value in it. **The Sixth Principle: Plan Ahead for Reuse** Reuse saves time and effort. Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated. **The Seventh principle: Think**! Placing clear, complete thought before action almost always produces better results. When you think about something, you are more likely to do it right.	4	CO1	L1

3		What are the components used for use case diagram? Draw use case diagram for hospital management and use all the components in the diagram?	10	CO2	L2

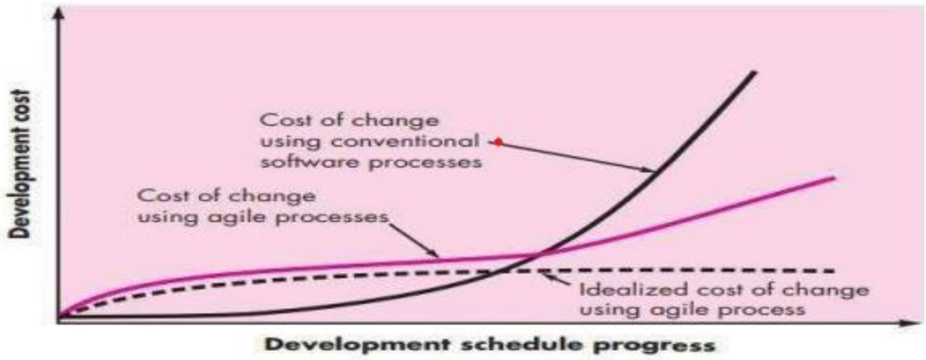
Components of a Use Case Diagram

1. **Actors**:
   - Represent entities (human, system, or organization) interacting with the system.
   - Represented as stick figures.
2. **Use Cases**:
   - Represent the functionalities or actions the system performs.
   - Represented as ovals.
3. **System Boundary**:
   - Defines the scope of the system.
   - Represented as a rectangle enclosing use cases.
4. **Relationships**:
   - **Association**: Links actors to use cases (solid lines).
   - **Include**: Indicates a use case includes the functionality of another (dashed arrow with "<<include>>").
   - **Extend**: Indicates a use case's optional or conditional behavior (dashed arrow with "<<extend>>").
   - **Generalization**: Represents inheritance between use cases or actors (arrow with hollow triangle).

Use Case Diagram for Hospital Management System

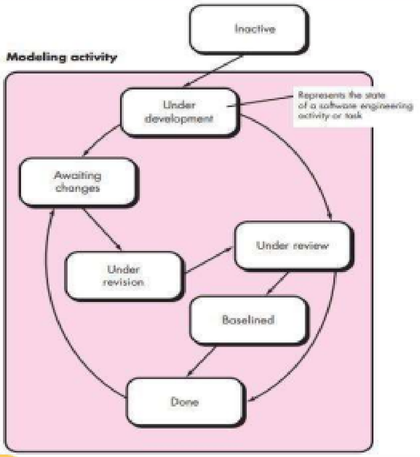
This system includes functionalities for:

- **Actors**:
  - Patient
  - Doctor
  - Receptionist
  - Administrator
- **Use Cases**:
  - Schedule Appointment
  - Manage Patient Records
  - Conduct Diagnosis
  - Generate Bill
  - Register Patient
- **Relationships**:
  - The receptionist registers a patient.
  - The patient schedules an appointment with the receptionist.
  - The doctor conducts diagnosis and uses patient records.
  - The administrator generates bills for the patient.

4		What is agility in the context of software engineering work? How you define change costs as a function of time in development ➢ In conventional software development the cost of change increases non linearly as a project progresses (**Fig Solid Black curve).** ➢ **An agile process reduces the cost of change because software is released in increments** and change can be better controlled within an increment. ➢ Agility argue that a well-designed agile process "flattens" the cost of change curve shown in following figure (**shaded, solid curve**), allowing a software team to accommodate changes late in a software project without dramatic cost and time impact. ➢ When incremental delivery is coupled with other agile practices such as continuous unit testing and pair programming, the cost of making a change is attenuated(reduced). Although debate about the degree to which the cost curve flattens is ongoing, there is evidence to suggest that a significant reduction in the cost of change can be achieved. application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.  	10	CO2	L2

5		Discuss about concurrent and specialized process models and mention advantages and disadvantages.	10	CO2	L2

Figure 2.8
One element of the concurrent process model



The activity modelling may be in any one of the states noted at any given time, similarly other activities, actions or tasks (Communication, Construction) can be represented in analogous manner.

All Software Engineering activities exist concurrently but reside in different states. E.g. Early in a project the communication activity has completed its 1st iteration and exists in the **awaiting changes** state.

The **modelling activity** (which existed in **inactive state**) while initial communication was completed now make a transition into the **under-development state.**

If however, the customer indicates that changes in requirement must be made, the modelling activity moves from under-development state to awaiting changes state.

Concurrent modelling defines a series of events that will trigger transitions from state to state for each of the software engineering activities, actions or tasks.

Concurrent modelling is applicable for all types of software development and provides an accurate picture of the current state of the project

- The component-based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of

  software. However, the component-based development model constructs applications from prepackaged software components.

- Modeling and construction activities begin with the identification of candidate components. These components can be designed as either conventional software modules or object-oriented classes or packages of classes. Regardless of the technology that is used to create the components.

- The component-based development model incorporates the following steps
  1. Available component-based products are researched and evaluated for the application domain in question.
  2. Component integration issues are considered.
  3. A software architecture is designed to accommodate the components.
  4. Components are integrated into the architecture.
  5. Comprehensive testing is conducted to ensure proper functionality.

- The component-based development model leads to software reuse, and reusability provides software engineers with a number of measurable benefits.

6		what is requirement engineering and briefly exxplain about requirement engineering process **Requirement Engineering (RE)** Requirement engineering is the process of systematically gathering, analyzing, documenting, and managing the needs and requirements of stakeholders for a software system. It ensures that the developed system fulfills its intended purpose and aligns with stakeholders' expectations. **Requirement Engineering Process** The requirement engineering process consists of the following key steps: 1. **Requirement Elicitation**: o Identify and collect requirements from stakeholders. o Techniques: interviews, surveys, workshops, observation, and brainstorming. 2. **Requirement Analysis**: o Evaluate collected requirements for feasibility, completeness, consistency, and ambiguity. o Prioritize requirements based on stakeholders' needs and project constraints. 3. **Requirement Specification**: o Document requirements in a structured format like a Software Requirements Specification (SRS). o Include functional, non-functional, and system constraints. 4. **Requirement Validation**: o Verify and validate that the documented requirements reflect stakeholders' needs accurately. o Techniques: reviews, walkthroughs, and prototypes. 5. **Requirement Management**: o Handle changes to requirements due to evolving stakeholder needs or project scope. o Maintain traceability between requirements and ensure consistency. This process ensures that the software developed aligns with user expectations, reduces the risk of rework, and promotes efficient project execution.	10	CO2	L2