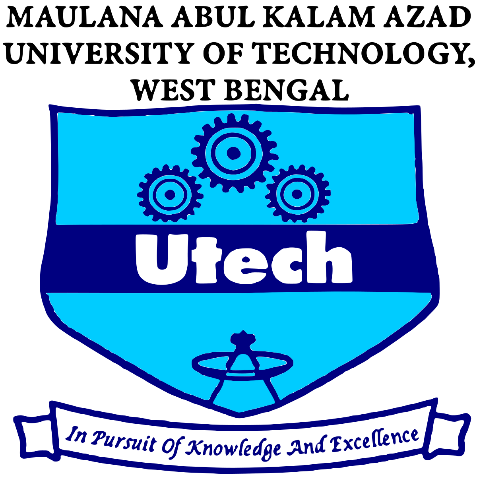
MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

**(FORMERLY KNOWN AS WEST BENGAL UNIVERSITY OF TECHNOLOGY)**



**CA 1**

**Name : Arijit Chabri**

**Reg no: 221000510898**

**Roll no: 10071022023**

**Steram: MCA ( 3rd Sem )**

**Season: 2022-2024**

**Subject: Software Engineering (MCAC303)**

1. What are the various phases of SDLC?

The software development life cycle is like a road map for crafting excellent software that's high-quality, affordable, and timely. The key aim is to reduce risks and ensure the software delights customers throughout its creation and beyond.

This involves crafting a thorough plan to steer the product's development. It means dividing the process into smaller parts that teams can handle, finish, and evaluate, making the entire journey smoother and more efficient.

## The 6 Phases of the Software Development Life Cycle:

1. **Planning stage:** The project planning stage is the first stage of the SDLC, during which you collect business needs from your customer or stakeholders. During this stage, you assess the product's viability, income potential, production costs, end-user requirements, etc
2. **Feasibility or Requirements of Analysis Stage**: The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

* Define any prototype system requirements
* Evaluate alternatives to existing prototypes
* Perform research and analysis to determine the needs of end-users

Furthermore, developers will often create a software requirement specification or SRS document.

This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

1. **Designinig the Software:** The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.
2. **Developing the project:** In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.
3. **Testing:** After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.

During this stage, unit testing, integration testing, system testing, acceptance testing are done

1. **Deployment:** Once the software is certified, and no bugs or errors are stated, then it is deployed. Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment. After the software is deployed, then its maintenance begins.
2. **Maintenance:** Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time. This procedure where the care is taken for the developed product is known as maintenance.
3. **What is feasibility study?**

The main aim of feasibility study is to determine weather it would be technically, financially and socially feasible to develop the product. The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behaviour of system.

Key aspects examined during a feasibility study include:

**Technical Feasibility**: Is the technology required for the project available and suitable? Can the necessary technical requirements be met?

**Financial Feasibility**: Is the project financially viable? Can the organization afford to undertake the project, and will the expected benefits outweigh the costs?

**Operational Feasibility**: Will the project integrate well with existing processes and systems? Are there any operational challenges that might arise during implementation?

**Schedule Feasibility**: Can the project be completed within the desired time frame? Are there any time constraints or dependencies that could affect the project's success?

**Resource Feasibility**: Are the necessary resources (such as personnel, equipment, and materials) available to support the project?

**Market Feasibility**: Is there a demand for the product or solution being proposed? Does it have a potential market and customer base?

**Legal and Regulatory Feasibility**: Are there any legal or regulatory constraints that could impact the project's execution?

1. What is SRS?

SRS stands for "Software Requirements Specification." It is a detailed document that outlines the functional and non-functional requirements of a software project. The SRS serves as a blueprint that describes what the software will do, how it will behave, and what it will look like from a user's perspective.

The main purpose of an SRS is to provide a clear and comprehensive understanding of the software's requirements to all stakeholders, including developers, designers, testers, and clients. It serves as a foundation for communication and collaboration among team members throughout the software development process.

Key components typically included in an SRS document are:

* **Introduction**: Provides an overview of the software project, its objectives, scope, and context within the larger system or environment.
* **Functional Requirements**: Describes the specific functionalities the software must perform, often presented in the form of use cases or user stories.
* **Non-Functional Requirements**: Specifies attributes that define the software's quality, performance, security, scalability, and other aspects that aren't directly related to specific functionalities.
* **User Interface (UI) and User Experience (UX) Requirements:** Describes how the user interfaces will look, feel, and behave, including layouts, navigation, and interactions.
* **System Architecture:** Provides an overview of the software's high-level architecture, including components, modules, and their interactions.
* **Data and Database Requirements:** Describes data structures, storage, and database interactions required by the software.
* **External Interfaces:** Specifies interactions with external systems, APIs, and third-party components.
* **Constraints and Assumptions:** Documents any limitations, constraints, or assumptions that impact the software's development or usage.
* **Performance Requirements:** Outlines criteria for system performance, such as response times, throughput, and resource usage.
* **Security Requirements**: Details security measures and protocols to ensure the software's protection against potential threats.
* **Quality Assurance and Testing Requirements:** Describes testing methodologies, acceptance criteria, and quality standards.
* **Documentation Requirements:** Specifies the documentation that needs to be created alongside the software, such as user manuals, technical documentation, and training materials.

To make sure that all stakeholders have a common knowledge of the software's scope and needs, it is essential to create a thorough and accurate SRS. In later phases of development, it helps avoid misunderstandings, scope shifts, and expensive revisions.

1. **What are the 5 components of information systems technology?**

Information Systems Technology typically consists of five main components:

* **Hardware:** This component includes all the physical equipment and devices that are used to collect, process, store, and transmit data. It encompasses items like computers, servers, mobile devices, networking equipment, and peripherals like printers and scanners.
* **Software:** Software refers to the programs, applications, and operating systems that enable users to interact with the hardware and perform various tasks. This includes both system software (like operating systems) and application software (like word processors, spreadsheets, and databases).
* **Data:** Data is the raw material that information systems process. It can include text, numbers, images, videos, and any other form of information. Proper management and organization of data are crucial for effective decision-making and operations within an organization.
* **Procedures**: Procedures are the rules, guidelines, and processes that dictate how an organization uses its information systems. They help define how tasks are performed, how data is input and processed, and how information is shared within the organization.
* **People:** People are an essential component of any information system. They include the individuals who design, develop, operate, maintain, and use the technology. Effective training, communication, and collaboration among users and IT professionals are crucial for the success of an information system.

These five components work together to create an integrated and functional information system that supports an organization's goals and objectives.

1. **What is spiral model and waterfall model?**

The Spiral Model and the Waterfall Model are two distinct software development methodologies used to guide the process of creating software. Each model has its own approach, characteristics, and advantages. Let's explore both models:

**Waterfall Model:**

The Waterfall Model is a linear and sequential software development methodology. It divides the software development process into distinct, well-defined phases that must be completed in a specific order:

* **Requirements:** Gathering and documenting the software's requirements from stakeholders.
* **Design:** Creating a detailed design plan based on the gathered requirements.
* **Implementation:** Developing and coding the software based on the design specifications.
* **Testing:** Verifying and validating the software to ensure it meets the requirements and is free of defects.
* **Deployment:** Deploying the fully developed and tested software to the production environment.
* **Maintenance:** Ongoing support, updates, and bug fixes as needed.

The Waterfall Model is characterized by its rigid structure, where each phase must be completed before proceeding to the next. While it provides clear documentation and is suitable for projects with well-defined requirements, it can be less adaptable to changes that arise during development.

**Spiral Model:**

The Spiral Model is an iterative and risk-driven approach to software development. It focuses on managing uncertainties and mitigating risks throughout the project's lifecycle. The model consists of repeating cycles, or "spirals," each containing four main phases:

* **Planning**: Identifying goals, constraints, and risks, and developing a strategy for the project.
* **Risk Analysis:** Evaluating potential risks and developing mitigation strategies.
* Engineering: Designing, implementing, and testing the software based on the identified requirements.
* **Evaluation**: Reviewing the progress and results of the previous phases, assessing risks, and making decisions about whether to proceed to the next spiral.

The Spiral Model emphasizes continuous refinement and adaptation. It allows for incremental development, frequent evaluations, and adjustments based on changing requirements or emerging risks. This makes it well-suited for projects with evolving or uncertain requirements.