Week 1

* Software engineering is the branch of computer science that deals with the design, development, testing, and maintenance of software applications.
* Traditional programming has a narrower scope and is primarily centered around the code from creation to testing while Software engineering encompasses the entire lifecycle of a software from its conception to deployment and maintenance.

The SDLC is a structured process used to design, develop and test good quality software. The steps are;

1. Planning: As in software development, planning is an essential stage in everything. The organization's developers also do requirement analysis at this same period. Customer feedback and sales department/market surveys are used to get this.   
     
   The foundation of a simple project is made up of the data from this research. The outcome of planning is the project's quality.
2. Defining: All of the specifications for the target software are defined at this point. Stakeholders such as market analysts and customers approve these needs.   
   Using SRS (Software Requirement Specification) satisfies this. This document serves as a kind of checklist for everything that needs to be established and defined over the course of the project.
3. Designing: Software designers can use SRS as a guide to create the optimal software architecture. As a result, the Design Document Specification (DDS) contains several designs for the product architecture based on the requirements specified in the SRS.   
     
   Stakeholders and market analysts evaluate this DDS. The design that makes the most sense and is most practicable is selected for development after all relevant considerations are considered.
4. Developing Product: The product's essential development begins at this point. Developers utilise a special programming code in accordance with the DDS design for this. It is crucial that coders adhere to the guidelines established by the association. At this point, standard programming tools like interpreters, compilers, debuggers, etc. are also used. Certain widely used languages, such as Python, Java, C/C++, etc., are used in accordance with software regulations.
5. Product Testing and Integration: Software must be tested after it has been developed to make sure it operates as intended. Nevertheless, not much testing is done during the SDLC. As a result, all potential problems are currently being monitored, addressed, and retested. This guarantees that the product satisfies SRS quality standards.   
     
   Software documentation is a crucial component of the software development life cycle, along with training and support. An effective document serves as a tool and a repository for the knowledge needed to understand software functionalities, processes, and maintenance. Information about how to operate the product is also included in the documentation.
6. Deployment and Maintenance of products: The final product is tested thoroughly before being phased out in accordance with the organization's plan. It is then put to the test in an actual industrial setting. Making sure it operates smoothly is crucial. The company distributes the product in its entirety if it does well. Once the organization has gathered positive feedback, it either distributes it unaltered or with additional enhancements to make it even more advantageous for the clients. Still, this is insufficient on its own. Thus, in addition to the deployment, the product's oversight.

* Agile VS Waterfall Models compared and contrasted below;

1. **Roles**: Waterfall assigns team members to projects in a disciplined manner, giving each member a specified set of tasks to perform. On the other hand, the agile model fosters a more self-organizing team structure by enabling team members to work together on various project components over time.
2. **Planning**: In a waterfall project, planning is a sequential process that starts at the outset and involves outlining all needs and goals in great detail. Agile planning, on the other hand, is a continuous process that is modified when new requirements or information become available during the project's life cycle.
3. **Scope**: Even when change requests are handled appropriately, the waterfall technique usually discourages alterations to the project's scope. This is due to the methodology's requirement that a significant amount of time be spent initially refining the plan, which may increase the cost of revisions once the project has started. Agile, on the other hand, allows the development team to swiftly react to changing needs and is more flexible when it comes to scope modifications.
4. **Time frames**: Long-term projects with set deadlines are the focus of the waterfall technique. Each phase of the project depends on the one before it, and is performed in a linear fashion. Agile, on the other hand, uses brief iterations to produce value quickly, enabling teams to make changes to their plans over time and complete tasks in less time.
5. **Speed:** Because all criteria must be approved before development can start, waterfall projects typically take longer. On the other side, because agile uses iterative development cycles, projects are typically completed faster than waterfall ones.

When to use the Agile Model;

Agile project management works well when there are tight deadlines and budgets, when complex systems necessitate frequent feedback loops, or when the final goal may be ambiguous or challenging to define. Because it enables fast iteration and testing along the route, it is very useful for designing software applications. Agile is also beneficial for teams who must work closely together, such those that are dispersed across multiple regions.

* Requirements Engineering is a methodical and rigorous approach to the conception, development, and verification of requirements for a software system. The requirements engineering process involves multiple tasks that aid in understanding, documenting, and managing stakeholder needs in order to ensure the successful construction of a software product. The process involves the following steps;

1. Feasibility Study
2. Requirements Elicitation
3. Requirements Specification
4. Requirements for Verification and Validation

* Modularity in Software Design

The term "modularity" in software design refers to the process of breaking software up into independent, separate modules, each in charge of a particular feature or functionality. By isolating functional boundaries, this approach improves code maintainability, scalability, and reusability and makes complex systems easier to manage and evolve.

* How it improves Scalabilty and Maintainability of software systems

A software system is often enhanced by adding new features as they become available. Therefore, in situations where there is a change in the external conditions, modular systems are better equipped to add or remove necessary components. It is known as scalability, and it allows you to ascertain whether the software can grow with your needs.

* Testing in Software Engineering

The process of running a program to identify faults is called testing. Our program needs to be error-free in order to function properly. The software will be free of all errors if the testing is successful.

* Levels of Software Testing
* There are three (3) major types of functional software testing, namely;

1. Unit Testing:

One way to test individual software modules or components is through unit testing. It is usually carried out by developers to make sure that each software component is operating as intended. Unit tests are intended to test single code segments, such a function or procedure, and are typically automated. At the lowest stage of the software development life cycle, unit testing is the process of testing individual code units separately.

1. Integration Testing: One way to verify how various software application units or components interact with one another is through integration testing. It is employed to locate and fix any problems that might occur when several software components are integrated. Verifying that the various software components interact as intended is the goal of integration testing, which is usually carried out before to functional testing and following unit testing.
2. System Testing: System testing is a subset of software testing that assesses the general performance and functionality of a whole software solution that is fully integrated. It checks to see if the system satisfies the criteria and is ready to be delivered to end customers. This kind of testing is carried out before to acceptance testing but following integration testing.

* Version Control Systems

Version control systems are a class of software tools that monitor code updates and assist in tracking down changes made to files.

* Importance of Version Control Systems

1. Increases the rate of project development by offering effective cooperation.
2. Maximizes staff capabilities, productivity, and product delivery speed through improved help and communication.
3. Reduce the likelihood of mistakes and disagreements while the project is being developed by tracking every little modification.
4. Through this VCS, project participants or employees can participate from anywhere, regardless of their current location.
5. A separate working copy is kept for each project contributor and isn't merged with the main file until the working copy has been verified. The most well-known examples are Microsoft TFS, Git, and Helix core.
6. Aids in recovery following a disaster or other unforeseen circumstance.
7. Explains to us the Who, What, When, and Why of the changes.

* A very common type of Version Control System that has been widely adopted across the industry is the Distributed VCS and the best example is Git. With these types of VCS, four (4) things are required to make your changes visible to others and they are;

1. You commit
2. You push
3. They pull
4. They update

* Software Project Management

Roles of a software project manager include;

1. Project estimation
2. Scheduling
3. Staffing
4. Risk Management
5. Tracking Progress
6. Monitoring and Reviewing

* Software Maintenance

Software maintenance is the process of making changes to a software product after it has been delivered in order to fix bugs, enhance functionality, or adjust it to a different environment.

* Types of Maintenance Activities

1. **Problem Identification Phase**

Input: Modification request.

Process: Assign change number, Classify modification request, Accept or reject change, Prioritize.

Control: Uniquely identified modification request, Enter modification request repository.

Output: Validated modification request, Validated process determinations.

1. **Analysis Phase**

Input: Project document, Repository information, Validated modification request.

Process: Feasibility analysis, Detailed analysis.

Control: Conduct technical review, Verify test strategy, Verify whether the documentation is updated or not, Identify security issues.

Output: Feasibility report, Detailed analysis report, Updated requirements, Preliminary modification list, Test strategy.

1. **Delivery Phase**  
   Enter: Verified/approved system.  
   Procedure: Setup and Instruction.  
   Control: Document describing the version.  
   Output: A paper describing the version.

* Ethical Issues in Software Engineering

1. Unethical Data Collection
2. Algorithmic Bias
3. Weak Security
4. Wrong Priorities

* Ways to mitigate these problems include;

1. Proactiveness
2. Honesty
3. Accountability
4. Responsibility

REFERENCES

1. <https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/>
2. <https://www.geeksforgeeks.org/version-control-systems/?ref=header_search>
3. <https://fullscale.io/blog/ethical-issues-in-software-development/>