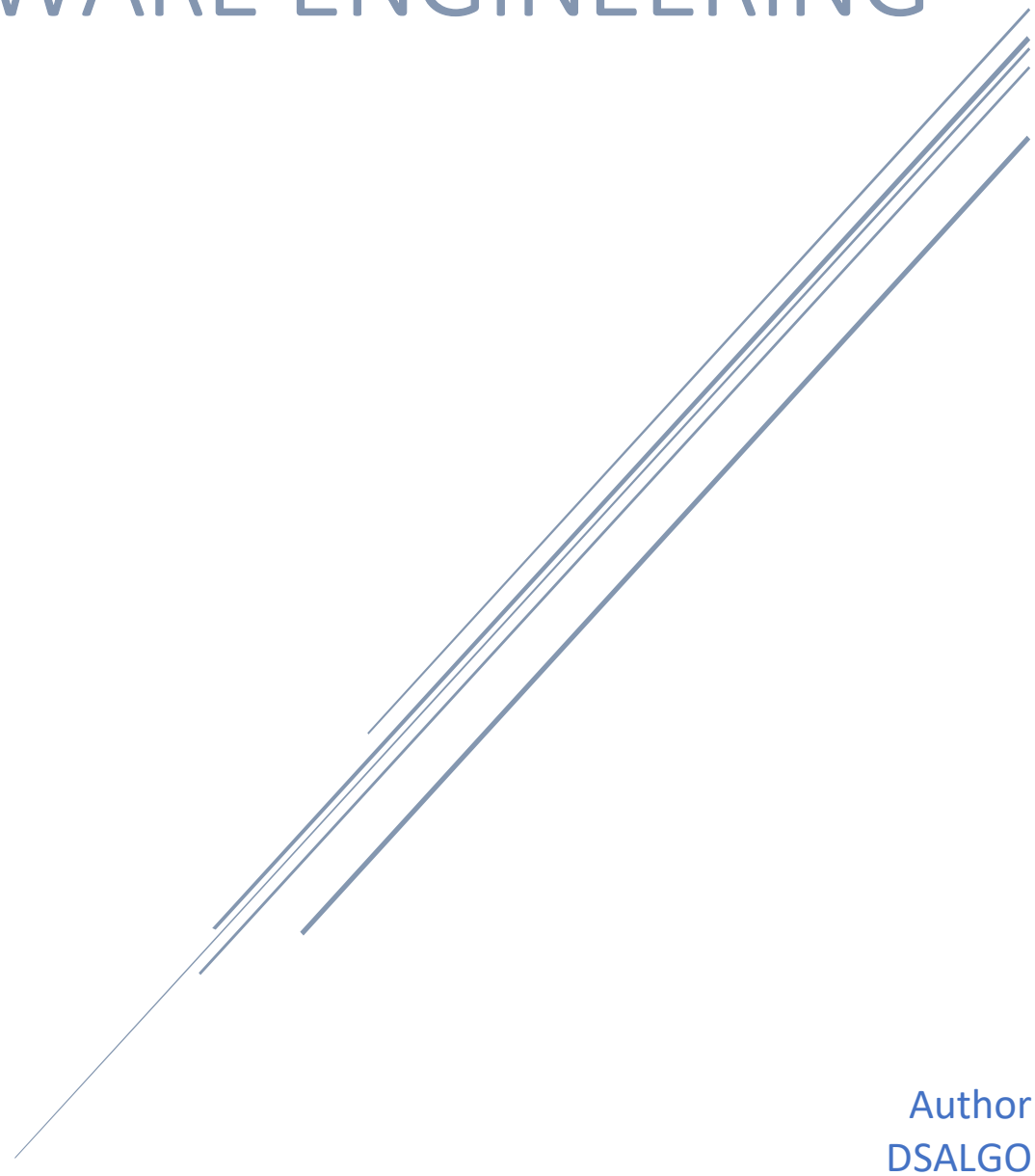
SOFTWARE ENGINEERING

UNIT -- 1



Author
DSALGO

Software engineering

- It is a field of computer science that involves the application of systematic, disciplined, and quantifiable approaches to the development, operation, and maintenance of software systems.
- It incorporates various aspects such as the design, development, testing, and evaluation of software and systems that enable computers to perform specific tasks.

Goal of SE

1. Meeting User Needs
2. Managing Project Risks
3. Communication
4. Ensuring Software Quality
5. Controlling Development Costs
6. Ensuring Legal Compliance

"goal is to deliver software that adds value to users, organizations, and society as a whole, contributing to innovation, efficiency, and progress"

Importance of SE

1. Quality Assurance
2. Cost Efficiency
3. Risk Management
4. Innovation and Creativity
5. User Satisfaction
6. Scalability and Adaptability

Characteristics of SE

1. Systematic approach
2. Requirements analysis
3. Design focus
4. Team collaboration
5. Development methodologies
6. Quality assurance
7. Documentation
8. Continuous improvement

Software Crisis

- software crisis refers to a period in the history of software development marked by significant challenges and issues in creating, managing, and maintaining software systems.
- It emerged in the late 1960s and early 1970s as the demand for software solutions rapidly increased, leading to a growing recognition of various problems and limitations associated with software development.
- Overall, the software crisis highlighted the need for improvements in software engineering practices, tools, methodologies, and education to address the complex and evolving nature of software development.

Reasons for the Software Crisis:

1. Complexity
2. Limited Resources
3. Changing Requirements
4. Technical Challenges

Results of the Software Crisis:

1. Project Failures
2. Schedule Delays
3. Inefficiencies
4. Cost Overruns
5. Quality Issues
6. Skill Shortages

Module

- Self-contained unit of software for encapsulating related functionality.
- Focuses on organizing code based on functional requirements.
- Represents a logical or functional aspect of the system.
- Developed and maintained within the same codebase or project.
- **Examples**: User Authentication , Product Mangagement , Order processing

Software Component:

- Reusable unit of software providing specific technical functionality.
- Encapsulates technical concerns with well-defined interfaces.
- Designed for reuse, interoperability, and scalability.
- Developed and maintained independently, deployable and managed separately.
- **Examples**:Database System , Logging , Payment gateway

"modules organize code by functionality, while software components focus on reusable technical capabilities."

Software Engineering Process:

- Involves computer science, IT, and discrete mathematics.
- Concerned with software development, low construction costs.
- Focuses on quality, involves testing, prototyping, and maintenance.
- Develops intangible products, follows established standards.
- Often uses Agile methodologies.
- **Example :**Designing a Video Game (Software Engineering):

Conventional Engineering Process:

- Involves science, mathematics, and empirical knowledge.
- Concerned with building physical products, high construction costs.
- Emphasizes mass production, has well-defined design requirements.
- Develops tangible products, follows regulations and standards.
- May use traditional project management approaches.
- **Example :**Constructing a Building

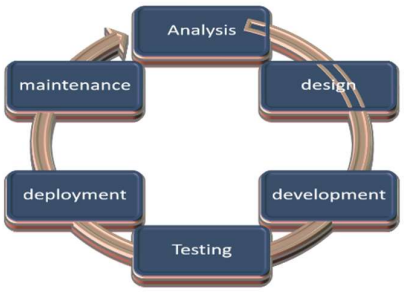
Software quality

- It refers to the degree to which a software product or system meets specified requirements, customer expectations, and industry standards.
- It encompasses various aspects/attributes .

1. **Functionality**: Correctness, Completeness, Suitability
2. **Reliability**: Availability, Fault tolerance, Resilience, Stability
3. **Usability**: Intuitiveness, Learnability, Efficiency, Error tolerance
4. **Performance**: Speed, Scalability, Resource utilization
5. **Security**: Confidentiality, Integrity, Authentication, Authorization
6. **Maintainability**: Readability, Modifiability, Extensibility, Testability

SDLC

- SDLC stands for Software Development Life Cycle.
- It's a structured approach used by software development teams.
- Involves phases like Analysis, design, implementation, testing, deployment, and maintenance.
- Guides the entire software development process from conception to completion.
- Different SDLC models like Waterfall, Agile, Iterative, and Spiral offer various approaches.
- Encourages learning from previous cycles to improve future development processes.
- Emphasizes understanding and meeting customer needs throughout the development lifecycle.



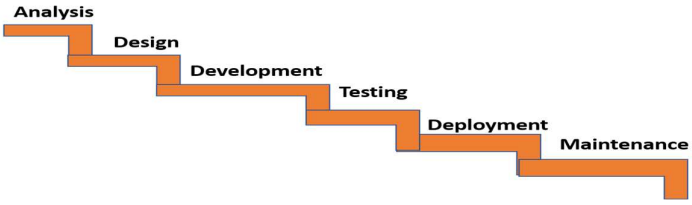
- **Analysis**: This is where the team identifies the needs of the clients and the goals of the software.
- **Design**: In this stage, the team outlines the specifications of the software and how it will operate.
- **Development**: This is when the actual coding and development of the software takes place.
- **Testing**: Once the software is developed, it is tested to ensure it works as intended.
- **Deployment**: After testing, the software is launched or 'deployed'.
- **Maintenance**: This is the final phase, where the software is updated and improved over time to meet the changing needs and demands of the users.

"Trick : A Daring Dog Takes Daily Marches"

Waterfall Model

- The Waterfall Model is a sequential software development process
- where progress flows steadily downwards (like a waterfall) through several phases.
- Each phase must be completed before the next phase begins, with little to no overlap between them.
- **Characteristics :**

1.Sequential Approach
2.No Iterations
3.Rigidity
4.Suitability
5.Phases
6.Documentation
7.Predictability



- **Advantages:**

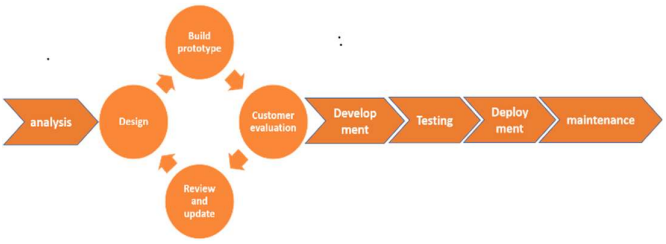
1.Easy to Understand
2.Properly Defined Phases
3.Suitable for Small Projects
4.Clear Milestones
5.Facilitates Good Documentation

- **Disadvantages:**

1.No Feedback Loop
2.No Overlapping of Phases
3.Late Discovery of Defects
4.Difficulty in Handling Changes
5.Limited Flexibility

Prototype Model

- It is an iterative software development approach
- where a prototype (a working model of the system) is built to gather feedback and refine requirements before developing the final product



- **Build Prototype:**
  - Create basic version of S/w based on initial requirements.
  - Focus on showcasing key functionalities.
  - Rapid development to provide stakeholders with an early model for evaluation.
- **Customer Evaluation:**
  - Gather feedback from end-users and stakeholders.
  - Validate requirements & ensure alignment with user needs.
  - Involve customers throughout the development process.
- **Review & Update:**
  - Iteratively refine the prototype based on feedback.
  - Address usability issues, missing features, and design flaws.
  - Continue updating until stakeholders are satisfied.

- **Advantages**:

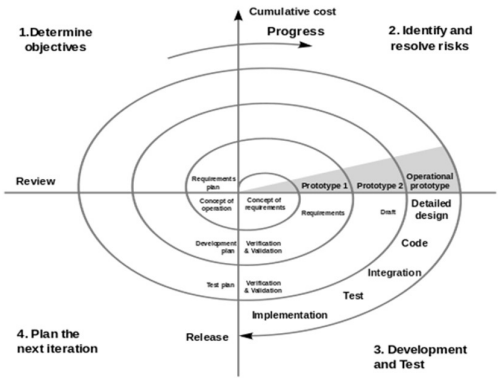
1.Early Feedback
2.Reduced Development Time
3.Risk Mitigation
4.Requirement Clarification
5.Flexibility

- **Disadvantages**:

1.Incomplete Functionality
2.Time and Cost exceeds
3.Documentation Overlook
4.feature creep
5.Potential Misinterpretation

Spiral Model

- The Spiral Model in Software Development Life Cycle (SDLC) consists of four phases
- which are iteratively repeated for each development stage.



- **Planning**:
  - project team analyzes the requirements and feasibility of the project.
  - The team identifies project objectives, constraints, and alternative solutions.
  - A preliminary project plan, including schedules, budgets, and resource allocations, is developed.
  - Stakeholders collaborate to define project goals and establish communication channels.
- **Risk Analysis:**
  - The Risk Analysis phase focuses on identifying and mitigating potential risks associated with the project.

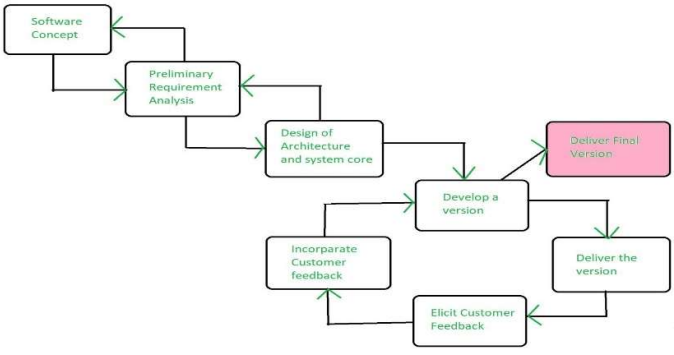
- The team conducts a thorough risk assessment, considering technical, organizational, and environmental factors.
- Strategies for risk mitigation, such as prototyping, proof-of-concepts, and contingency planning, are developed.

- **Development**:
  - project team executes the planned activities and produces the deliverables.
  - Software requirements are translated into design specifications, and coding begins based on the chosen development approach.
  - Quality assurance processes, such as code reviews, testing, and documentation, are integrated into the development workflow.
- **Evaluation**:
  - The Evaluation phase involves assessing the results of the development activities and gathering feedback from stakeholders.
  - The developed software is evaluated against project objectives, requirements, and quality standards.
  - Stakeholders provide feedback on usability, functionality, performance, and other aspects of the software.
  - Lessons learned from the evaluation phase are used to refine the project plan and inform future iterations.

- **Advantages**:
  1. Effective Risk Management
  2. Flexibility to Accommodate Changes
  3. Early Prototyping for Validation
  4. Increased Customer Involvement
  5. Support for Gradual Builds and Incremental Releases
- **Disadvantages**:
  1. Complexity in Management
  2. High Implementation Cost
  3. Not Suitable for Small Projects
  4. Dependency on Expertise
  5. Documentation Overhead

evolutionary developement model

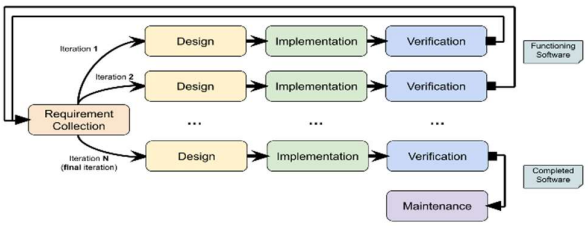
- Evolutionary models are iterative, allowing for the development of more complete versions of the software over time.
- The system is divided into modules or functional units, which are implemented and delivered incrementally.
- Developers start by creating the core modules of the system.
- The initial product skeleton is refined by adding new functionalities in successive versions.
- Each version of the product is a functioning system capable of performing additional useful work.
  - This model is also referred to as the successive versions model.



- **Advantages**:
  1.Early Delivery of Functionality 4.Risk Reduction
  2.Flexibility 5.Stakeholder Engagement
  3.Continuous Improvement
- **Disadvantages**:
  1.Increased Complexity 3.Potential Scope Creep
  2.Higher Cost 4.Uncertainty in Planning

Iterative enhancement model

- The stages of the Iterative Enhancement Model typically involve iterative cycles of development, feedback, and improvement
- Here's a breakdown of the key stages:



1.Initial Planning and Requirements Gathering:

- Identify project objectives, requirements, and constraints.
- Collaborate with stakeholders to define scope and priorities.

2.Iteration Planning:

- Define tasks and goals for each iteration.
- Prioritize development tasks based on stakeholder feedback.

3.Development and Implementation:

- Implement planned features and enhancements.
- Make incremental improvements based on iteration priorities.

4.Testing and Validation:

- Conduct quality assurance processes, including testing.
- Address bugs and issues identified during testing.

5.Feedback and Evaluation:

- Gather stakeholder feedback on software usage.
- Analyze feedback to identify areas for improvement.

6.Iteration Review and Reflection:

- Hold a review session at the end of each iteration.
- Identify strengths, weaknesses, and opportunities for improvement.

7.Iterative Cycle Repeats:

- Repeat the cycle for each iteration.
- Build upon previous iterations for gradual software evolution and improvement.

Advantages:

1.Flexibility 2.Continuous Improvement
3.Stakeholder Engagement 4.Risk Management

Disadvantages:

1.Increased Complexity 2.Higher Cost
3.Potential Scope Creep 4.Uncertainty in Planning