Software engineering

Software

Software is a program or set of programs containing instructions that provide desired functionality.

Software engineering

Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software. Software engineering includes a variety of techniques, tools, and methodologies, including requirements analysis, design, testing, and maintenance.

Key Principles of Software Engineering

Modularity: Breaking the software into smaller, reusable components that can be developed and tested independently.

Abstraction: Hiding the implementation details of a component and exposing only the necessary functionality to other parts of the software.

Encapsulation: Wrapping up the data and functions of an object into a single unit, and protecting the internal state of an object from external modifications.

Reusability: Creating components that can be used in multiple projects, which can save time and resources.

Maintenance: Regularly updating and improving the software to fix bugs, add new features, and address security vulnerabilities.

Testing: Verifying that the software meets its requirements and is free of bugs.

Design Patterns: Solving recurring problems in software design by providing templates for solving them.

Agile methodologies: Using iterative and incremental development processes that focus on customer satisfaction, rapid delivery, and flexibility.

Continuous Integration & Deployment: Continuously integrating the code changes and deploying them into the production environment.

Classification of software

Purpose: Software can be classified as [system software](https://www.geeksforgeeks.org/system-software/) (e.g. operating systems, device drivers) or [application software](https://www.geeksforgeeks.org/application-software/) (e.g. word processors, games).

Platform: Software can be classified as native software (designed for a specific operating system) or cross-platform software (designed to run on multiple operating systems).

Deployment: Software can be classified as installed software (installed on the user’s device) or cloud-based software (hosted on remote servers and accessed via the internet).

License: Software can be classified as proprietary software (owned by a single entity) or open-source software (available for free with the source code accessible to the public).

Development Model: Software can be classified as traditional software (developed using a waterfall model) or [agile software](https://www.geeksforgeeks.org/software-engineering-agile-software-development/) (developed using an iterative and adaptive approach).

Size: Software can be classified as small-scale software (designed for a single user or small group) or enterprise software (designed for large organizations).

User Interface: Software can be classified as [Graphical User Interface (GUI)](https://www.geeksforgeeks.org/difference-between-cli-and-gui/) software or [Command-Line Interface (CLI)](https://www.geeksforgeeks.org/difference-between-cli-and-gui/) software.

Components of Software Characteristics

Functionality

It refers to the degree of performance of the software against its intended purpose.

Reliability

A set of attributes that bears on the capability of software to maintain its level of performance under the given condition for a stated period of time.

Efficiency

It refers to the ability of the software to use system resources in the most effective and efficient manner. The software should make effective use of storage space and executive command as per desired timing requirements.

Usability

It refers to the extent to which the software can be used with ease. the amount of effort or time required to learn how to use the software.

Maintainability

It refers to the ease with which modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

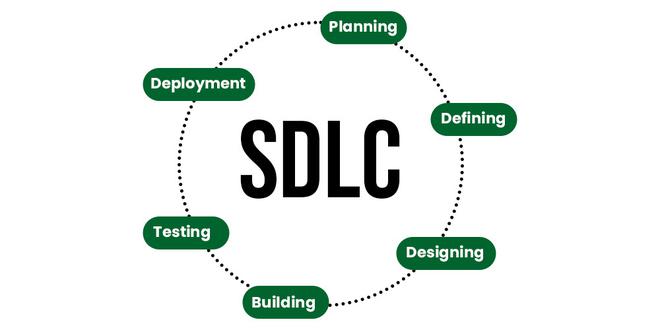
Portability

A set of attributes that bears on the ability of software to be transferred from one environment to another, without minimum changes.

Software lifecycle

* SDLC stands for software development life cycle.
* It is a process followed for software building within a software organization.
* SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software.
* The life cycle defines a method for improving the quality of software and the all-around development process.

The SDLC model involves six phases or stages while developing any software. SDLC is a collection of these six stages, and the stages of SDLC are as follows:



Stage-1: Planning and Requirement Analysis

* Planning is a crucial step in everything, just as in software development.
* In this same stage, requirement analysis is also performed by the developers of the organization.
* This is attained from customer inputs, and sales department/market surveys.
* The information from this analysis forms the building blocks of a basic project.
* The quality of the project is a result of planning.
* Thus, in this stage, the basic project is designed with all the available information.

Stage-2: Defining Requirements

* In this stage, all the requirements for the target software are specified.
* These requirements get approval from customers, market analysts, and stakeholders.   
  This is fulfilled by utilizing [SRS (Software Requirement Specification)](https://www.geeksforgeeks.org/software-engineering-quality-characteristics-of-a-good-srs/).
* This is a sort of document that specifies all those things that need to be defined and created during the entire project cycle.

Stage-3: Designing Architecture

* SRS is a reference for software designers to come up with the best architecture for the software. Hence, with the requirements defined in SRS, multiple designs for the product architecture are present in the [Design Document Specification (DDS)](https://www.geeksforgeeks.org/design-documentation-in-software-engineering/).   
  This DDS is assessed by market analysts and stakeholders.
* After evaluating all the possible factors, the most practical and logical design is chosen for development.

Stage-4: Developing Product

* At this stage, the fundamental development of the product starts.
* For this, developers use a specific programming code as per the design in the DDS. Hence, it is important for the coders to follow the protocols set by the association.
* Conventional programming tools like compilers, interpreters, debuggers, etc. are also put into use at this stage.
* Some popular languages like C/C++, Python, Java, etc. are put into use as per the software regulations.

Stage-5: Product Testing and Integration

* After the development of the product, testing of the software is necessary to ensure its smooth execution.
* Although, minimal testing is conducted at every stage of SDLC. Therefore, at this stage, all the probable flaws are tracked, fixed, and retested.
* This ensures that the product confronts the quality requirements of SRS.

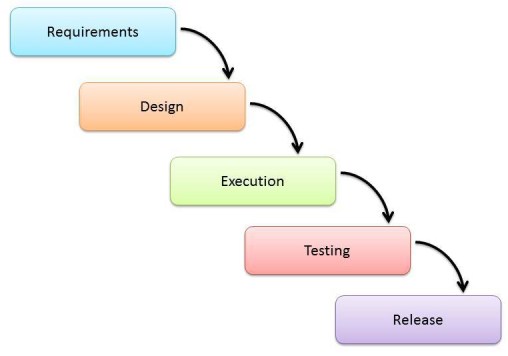
Stage 6: Deployment and Maintenance of Products

* After detailed testing, the conclusive product is released in phases as per the organization’s strategy. Then it is tested in a real industrial environment.
* It is important to ensure its smooth performance. If it performs well, the organization sends out the product as a whole.
* After retrieving beneficial feedback, the company releases it as it is or with auxiliary improvements to make it further helpful for the customers.
* However, this alone is not enough. Therefore, along with the deployment, the product’s supervision.

Different models

### **Waterfall Model**

* The [Waterfall Model](http://melsatar.blog/2018/02/16/the-waterfall-model-a-different-perspective/) is a linear sequential flow.
* In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation.
* This means that any phase in the development process begins only if the previous phase is complete.



### [**Prototype Model**](https://www.javatpoint.com/software-engineering-prototype-model)

* The prototyping model starts with the requirements gathering.
* The developer and the user meet and define the purpose of the software, identify the needs, etc.
* A '**quick design**' is then created.
* This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype.
* The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype.
* Looping takes place in this step, and better versions of the prototype are created. These are continuously shown to the user so that any new changes can be updated in the prototype.
* This process continue until the customer is satisfied with the system.
* Once a user is satisfied, the prototype is converted to the actual system with all considerations for quality and security.

### [Big bang model](https://www.javatpoint.com/software-engineering-big-bang-model)

* Big bang model is focusing on all types of resources in software development and coding, with no or very little planning.
* The requirements are understood and implemented when they come.
* This model works best for small projects with smaller size development team which are working together.
* It is also useful for academic software development projects.
* It is an ideal model where requirements are either unknown or final release date is not given.

### [**Iterative Model**](https://www.javatpoint.com/software-engineering-iterative-model)

* It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete.
* In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

# **Incremental Model**

* Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.
* In this model, each module goes through the requirements, design, implementation and testing phases.
* Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.



# **Agile Model**

* "**Agile process model**" refers to a software development approach based on iterative development.
* Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process.
* Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
* Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks.
* The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
* Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.



Scrum

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

There are three roles in it, and their responsibilities are:

* **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
* **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
* **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.