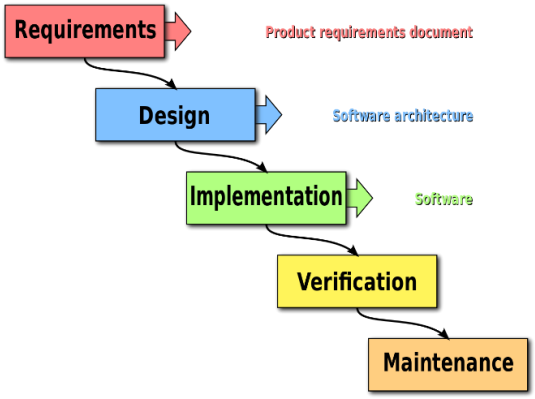
Traditional software development methodologies:

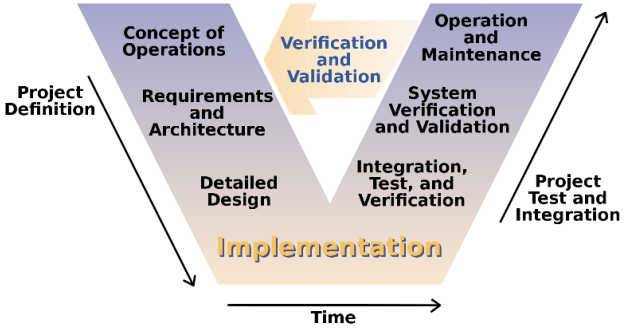
waterfall model,

The **waterfall model** is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. The approach is typical for certain areas of engineering design. In software development, it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction ("downwards" like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, deployment and maintenance.
The waterfall development model originated in the manufacturing and construction industries; where the highly structured physical environments meant that design changes became prohibitively expensive much sooner in the development process. When first adopted for software development, there were no recognised alternatives for knowledge-based creative work.[1].



V-model,

In software development, the **V-model**[2] represents a development process that may be considered an extension of the waterfall model, and is an example of the more general V-model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.



Spiral model

https://www.javatpoint.com/spiral-model
Spiral Model is a risk-driven software development process model. It is a combination of waterfall model and iterative model. Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.
Spiral Model: When to Use? Advantages & Disadvantages (guru99.com)

Prototyping,

A **prototype** is an early sample, model, or release of a product built to test a concept or process.[1] It is a term used in a variety of contexts, including semantics, design, electronics, and software programming. A prototype is generally used to evaluate a new design to enhance precision by system analysts and users.[2] Prototyping serves to provide specifications for a real, working system rather than a theoretical one.[3] In some design workflow models, creating a prototype (a process sometimes called **materialization**) is the step between the formalization and the evaluation of an idea.[4]

Iterative and incremental models

Iterative and incremental development is any combination of both iterative design or iterative method and incremental build model for development.
Usage of the term began in software development, with a long-standing combination of the two terms *iterative* and *incremental*[1] having been widely suggested for large development efforts. For example, the 1985 DOD-STD-2167[2] mentions (in section 4.1.2): "During software development, more than one iteration of the software development cycle may be in progress at the same time." and "This process may be described as an 'evolutionary acquisition' or 'incremental build' approach." In software, the relationship between iterations and increments is determined by the overall software development process.

Rapid application development methodology.

RAD methodology is a software design methodology that's designed to counter the rigidity of other traditional software development models–where you can't make changes easily after the initial development is complete.

RAD methodology is designed to be flexible to changes and to accept new inputs, like features and functions, at every step of the development process.

Rapid-application development (**RAD**), also called **rapid-application building** (**RAB**), is both a general term for adaptive software development approaches, and the name for James Martin's method of rapid development. In general, RAD approaches to software development put less emphasis on planning and more emphasis on an adaptive process. Prototypes are often used in addition to or sometimes even instead of design specifications. RAD is especially well suited for (although not limited to) developing software that is driven by user interface requirements. Graphical user interface builders are often called rapid application development tools.

Other approaches to rapid development include the adaptive, agile, spiral, and unified models.

Agile software development:

Agile is the ability to create and respond to change. Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

Agile software development refers to **software development methodologies** centered round the idea of iterative **development**, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. ... Scrum and Kanban are two of the most widely used **Agile methodologies**.

Fundamentals of agile software development methodologies,

Agile methodologies are methods that support Agile values and principles.

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

Agile manifesto,

Based on their combined experience of developing software and helping others do that, the seventeen signatories to the manifesto proclaimed that they value:[5]

- ***Individuals and interactions*** *over processes and tools*
- ***Working software*** *over comprehensive documentation*
- ***Customer collaboration*** *over contract negotiation*
- ***Responding to change*** *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

Presentation of a freely chosen agile methodology.

- **Scrum** is a framework utilizing an agile mindset for developing, delivering, and sustaining complex products,[1] with an initial emphasis on software development, although it has been used in other fields including research, sales, marketing and advanced technologies.[2] It is designed for teams of ten or fewer members, who break their work into goals that can be completed within time-boxed iterations, called *sprints*, no longer than one month and most commonly two weeks. The Scrum Team assess progress in time-boxed daily meetings of 15 minutes or less, called daily scrums. At the end of the sprint, the team holds two further meetings: the sprint review which demonstrates the work done to stakeholders to elicit feedback, and sprint retrospective which enables the team to reflect and improve.

- **Kanban** (Japanese: 看板, meaning signboard or billboard) is a lean method to manage and improve work across human systems. This approach aims to manage work by balancing demands with available capacity, and by improving the handling of system-level bottlenecks.
  Work items are visualized to give participants a view of progress and process, from start to finish—usually via a Kanban board. Work is pulled as capacity permits, rather than work being pushed into the process when requested.
  In knowledge work and in software development, the aim is to provide a visual process management system which aids decision-making about what, when, and how much to produce. The underlying Kanban method originated in lean manufacturing,[1] which was inspired by the Toyota Production System.[2] It has its origin in the late 1940s when the Toyota automotive company implemented a production system called just-in-time; which had the objective of producing according to customer demand and identifying possible material shortages within the production line. But it was Microsoft engineer David J. Anderson who realized how this method devised by Toyota could become a process applicable to any type of company that needs organization. Kanban is commonly used in software development in combination with other methods and frameworks such as Scrum.[3]