**Part I. Software Process**

1. **Specification**

* Identify the **purpose** of software (what the software needs to do for customers?).
* Identify the **constraints** on the software's quality and the user’s experience.

1. **Development**

* Software **design**
* Software **implementation**

1. **Validation**

* **Verification**: check if the software/system conforms to its specifications.
* **Validation**: check if the software/system meets the requirements of the system customer.

1. **Evolution**

* Changing existing software/system to meet new requirements of customer and market.

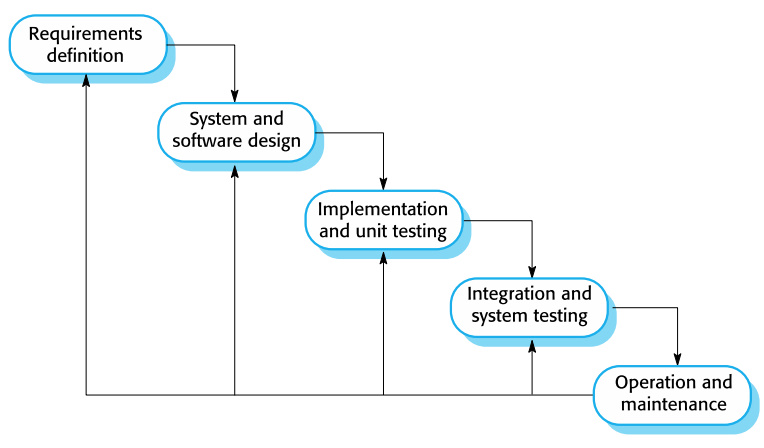
**Short paragraph to introduce these 4 stages:**

Making software involves four main stages: specification, development, validation, and evolution. In the **specification** stage, we identify what the software needs to do for customers and any constraints on its quality and user experience. This is where we figure out the purpose and requirements of the software. During the **development** stage, we design and build the software. This includes planning the structure and writing the code to create the program. The **validation** stage involves checking the software to ensure it works correctly and meets the customer’s needs. This includes verification to see if it matches the specifications and validation to ensure it fulfills customer requirements. Finally, in the **evolution** stage, we update and change the software to meet new customer and market needs. This involves making improvements and adding new features to keep the software useful and effective.

**Part II. Software Process Models**

1. **Waterfall Model**

* **Definition:** The waterfall model is a plan-driven model in which all process activities are planned in advance, and progress is measured based on this plan.
* **Principle:** a phase has to be completed before moving on to the next phase.
* The waterfall model is mostly used for large systems engineering projects in which the project is easily divided into separate modules with clear requirements.
* **Drawback:** difficult to respond to changing customer requirements.
* **Main stages**



**Short paragraph to explain why we use the waterfall model:**

The Waterfall Model is used for several important reasons. It is a plan-driven approach, which means all project activities are planned in advance. This helps ensure that everyone involved understands what needs to be done and when. The model follows a clear sequence of phases, where each phase must be completed before moving to the next. This structured approach makes it easier to manage and track the project’s progress. The Waterfall Model is particularly useful for large systems engineering projects that can be divided into separate modules with clear requirements. Since everything is planned out beforehand, it is easier to allocate resources and set timelines. Moreover, thorough documentation is created at each phase, which can be helpful for future reference and maintenance. While the Waterfall Model is not very flexible and can struggle to respond to changing customer requirements, it works well when the project requirements are stable and unlikely to change. Its predictability and clear structure make it a suitable choice for projects with well-understood needs.

1. **Agile Method**

* **How did the agile method come up?**

Traditional methodologies like Waterfall involved a sequential design process, where each phase needed to be completed before moving on to the next. This often led to inflexibility, long development cycles, and a final product that might not meet users' needs due to changing requirements during the development process.

Agile development methods emerged in the late 1990s. The purpose of the agile method is to respond quickly to changes in customer’s requirements. Another purpose of this method is to reduce the delivery time for working software systems (that means the software will be released frequently and quickly to the customer).

The idea of the agile method is that we divide a big project into smaller modules so that we reduce the complexity of the entire problem. Then, we apply the waterfall method to do these modules.

* **Features**
  + Program specification, design, and implementation are inter-leaved.
  + Rapid Application Development (which ensures faster product delivery), but not complete version: ra sản phẩm nhanh, nhưng không phải phiên bản hoàn chỉnh
  + Frequent releases of the software: phần mềm được phát hành thường xuyên
  + Reducing process overheads by minimizing documentation and producing high-quality code.
* **Pros and Cons**
  + Agile method is suitable for small or medium-sized products.
  + Agile method is useful when user requirements are not clear.
  + Customer participation in the development team.
  + Scaling agile methods to large systems is an issue.
* **4 Agile Manifestor**
  + Individuals and interactions over processes and tools. (**Các cá nhân và tương tác** hơn là các quy trình và công cụ**)**
  + Working software over comprehensive documentation. (**Phần mềm chạy được** hơn là dựa trên tài liệu toàn diện**)**
  + Customer collaboration over contract negotiation. (**Sự hợp tác của khách hàng**hơn là dựa vào hợp đồng)
  + Responding to change over following a plan. (**Đáp ứng sự thay đổi** hơn làso với việc tuân theo kế hoạch)
* **12 Principles**
  + Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. (**Ưu tiên cao nhất** là làm **hài lòng khách hàng** thông qua việc **phân phối sớm** và liên tục các phần mềm có giá trị.)
  + Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. (Hoan nghênh các yêu cầu thay đổi, **ngay cả trong giai đoạn muộn** của việc phát triển. Các quy trình nhanh khai thác sự thay đổi vì lợi thế cạnh tranh của khách hàng.)
  + Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. (**Cung cấp sản phẩm** phần mềm **thường xuyên**, từ vài tuần đến vài tháng, ưu tiên khoảng thời gian ngắn hơn.)
  + Business people and developers must work together daily throughout the project. (**Người kinh doanh và nhà phát triển** phải làm việc cùng nhau hàng ngày trong **suốt dự án**.)
  + Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. (Xây dựng các dự án xung quanh những **cá nhân có động lực**. Cung cấp cho họ **môi trường và sự hỗ trợ mà họ cần**, và tin tưởng để họ hoàn thành công việc.)
  + The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. (**Phương pháp hiệu quả** nhất để truyền tải thông tin đến và trong nhóm phát triển là **trò chuyện trực tiếp.**)
  + Working software is the primary measure of progress. (**Phần mềm hoạt động** là **thước đo chính của sự tiến bộ**.)
  + Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. (Các quy trình Agile thúc đẩy sự phát triển bền vững. Các nhà tài trợ, nhà phát triển và người dùng sẽ có thể duy trì một tốc độ phát triển liên tục.)
  + Continuous attention to technical excellence and good design enhances agility. (Sự quan tâm **liên tục**, sự xuất sắc về kỹ thuật và thiết kế tốt giúp tăng cường sự nhanh nhẹn.)
  + Simplicity--the art of maximizing the amount of work not done--is essential. (Sự **đơn giản** - nghệ thuật tối đa hóa khối lượng công việc chưa hoàn thành - là **điều cần thiết**.)
  + The best architectures, requirements, and designs emerge from self-organizing teams. (Các kiến ​trúc, yêu cầu và thiết kế tốt nhất xuất hiện từ các **nhóm dự án**.)
  + At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. (Theo định kỳ, các nhóm phản ánh về cách trở nên hiệu quả hơn, sau đó điều chỉnh hoạt động của mình sao cho cho phù hợp.)

**Part II. Software Specification**

1. **Functional Requirements & Non-Functional Requirements**

* **Functional Requirements:** 
  + Định nghĩa: là câu phát biểu về chức năng, tính năng, tên màn hình mà user sẽ dùng cho công việc của họ.
  + Dấu hiệu nhận biết: động từ liên quan đến xử lý thông tin, lưu trữ thông tin
  + Ví dụ:
    - App cho phép user quản lí giỏ hàng: thêm, xóa, cập nhật giỏ hàng
    - App cho phép search sản phâm theo keyword: tên, nhãn hiệu
    - App cho phép user lọc sản phâm theo: thương hiệu, giá
    - App cho phép xem lịch sử đơn hàng (đã mua và hoàn tất), trạng thái đơn hàng
* **Non-Functional Requirements**
  + Định nghĩa: là câu phát biểu về tính chất, chất lượng, nói về trải nghiệm, cảm xúc khi dùng áp. Không đề cập chi tiết đến 1 tính năng nào cụ thể mà nó nói chung cho toàn bộ app.
  + Dấu hiệu nhận biết: dùng tính từ mô tả tính chất của áp
  + Ví dụ:
    - App thiết kế đẹp, màu sắc hài hòa
    - App chạy nhanh
    - App dễ dùng
    - App có bảo mật tốt