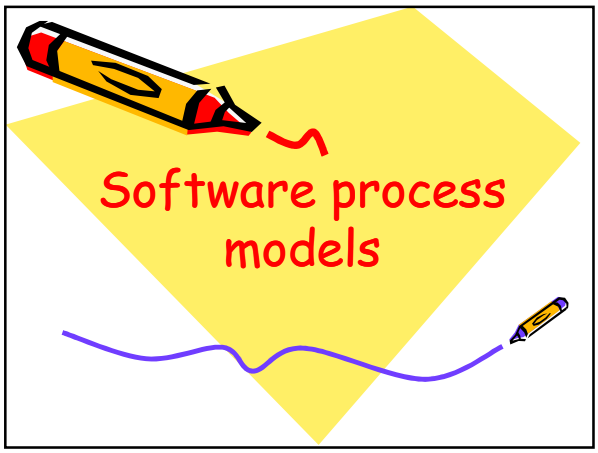


Prescriptive Process Models

- Originally proposed to bring order to chaos.
- Prescriptive process models advocate an orderly approach to software engineering.
  - However, will some extent of chaos (less rigid) be beneficial to bring some creativity?

2

Prescriptive Process Models

That leads to a few questions …

- If prescriptive process models strive for structure and order (prescribe a set of process elements and process flow), are they inappropriate for a software world that thrives on change?

Prescriptive Process Models

- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

The Waterfall Model

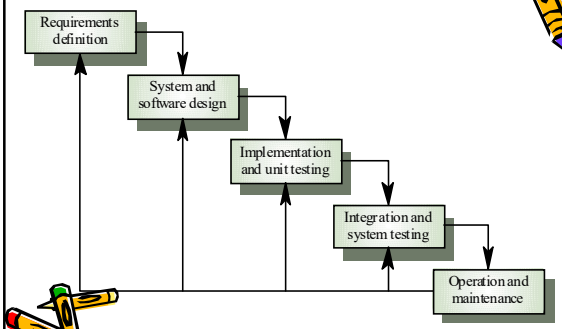
- It is the oldest paradigm for SE. When requirements are well defined and reasonably stable, it leads to a linear fashion.
- The classic life cycle suggests a systematic, sequential approach to software development.

5

The Waterfall Model

- The waterfall model emphasizes that a logical progression of steps be taken throughout the software development life cycle (SDLC), much like the cascading steps down an incremental waterfall.
- While the popularity of the waterfall model has waned over recent years,
- the logical nature of the sequential process used in the waterfall method cannot be denied, and it remains a common design process in the industry

Waterfall Model



Waterfall model phases

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

The drawback of the waterfall model is the difficulty of accommodating change after the process is underway

Requirements:

- During this initial phase, the potential requirements of the application are methodically analyzed and written down in a specification document that serves as the basis for all future development.
- The result is typically a requirements document that defines *what* the application should do, but not *how* it should do it

Analysis:

- During this second stage, the system is analyzed in order to properly generate the models and business logic that will be used in the application.

Design:

- This stage largely covers technical design requirements, such as programming language, data layers, services, etc.
- A design specification will typically be created that outlines how exactly the business logic covered in analysis will be technically implemented.

Coding:

- The actual source code is finally written in this fourth stage, implementing all models, business logic, and service integrations that were specified in the prior stages

Testing:

- During this stage, QA, beta testers, and all other testers systematically discover and report issues within the application that need to be resolved.
- It is not uncommon for this phase to cause a "necessary repeat" of the previous coding phase, in order for revealed bugs to be properly squashed.

Operations:

- Finally, the application is ready for deployment to a live environment.
- The operations stage entails not just the deployment of the application, but also subsequent support and maintenance that may be required to keep it functional and up-to-date

Advantages of Waterfall

- **Adapts to Shifting Teams**: While not necessarily specific to the waterfall model only, using a waterfall method does allow the project as a whole to maintain a more detailed, robust scope and design structure due to all the upfront planning and documentation stages.

Advantages of Waterfall

- This is particularly well suited to large teams that may see members come and go throughout the life cycle of the project, allowing the burden of design to be placed on the core documentation and less on any individual team member.

Advantages of Waterfall

- **Forces Structured Organization**: the waterfall model *forces* the project, and even the organization building said project, to be extraordinarily disciplined in its design and structure.
- Projects will include detailed procedures to manage every aspect of the project, from design and development to testing and implementation.

Advantages of Waterfall

- **Allows for Early Design Changes**: While it can be difficult to make design changes later in the process, the waterfall approach lends itself well to alterations early in the life cycle.

Advantages of Waterfall

- This is great when fleshing out the specification documents in the first couple stages with the development team and clients, as alterations can be made immediately and with minimal effort, since no coding or implementation has actually taken place up to that point.

Advantages of Waterfall

- **Suited for Milestone-Focused Development**: Due to the inherent linear structure of a waterfall project, such applications are always well-suited for organizations or teams that work well under a milestone- and date-focused paradigm.

Advantages of Waterfall

- With clear, concrete, and well understood stages that everyone on the team can understand and prepare for, it is relatively simple to develop a time line for the entire process and assign particular markers and milestones for each stage and even completion.

DisAdvantages of Waterfall

- **Nonadaptive Design Constraints:** inherent lack of adaptability across all stages of the development life cycle.
- When a test in stage five reveals a fundamental flaw in the design of the system, it not only requires a dramatic leap backward in stages of the process, but in some cases, can be often lead to a devastating realization regarding the legitimacy of the entire system.

DisAdvantages of Waterfall

- **Ignores Mid-Process User/Client Feedback**: Due to the strict step-by-step process that the waterfall model enforces, another particularly difficult issue to get around is that user or client feedback that is provided late into the development cycle can often be too little, too late..

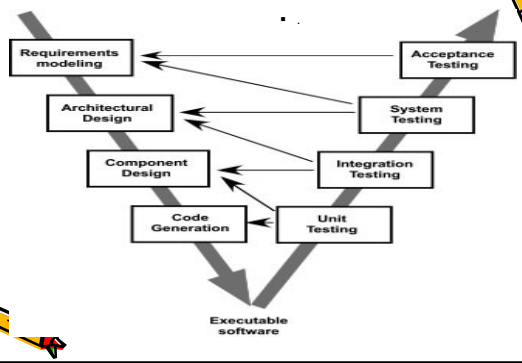
DisAdvantages of Waterfall

- While project managers can obviously enforce a process to step back to a previous stage due to an unforeseen requirement or change coming from a client, it will be both costly and time-consuming, for both the development team and the client

DisAdvantages of Waterfall

- **Delayed Testing Period**:the waterfall model largely shies away from testing until quite late into the life cycle.
- This means that most bugs or even design issues won't be discovered until very late into the process,

The V-Model



The V-Model

- Much like the traditional waterfall model, the V-Model specifies a series of linear stages that should occur across the life cycle, one at a time, until the project is complete.
- For this reason V-Model is not considered an agile development method,

The V-Model

- **V-Model** is an SDLC model that has a testing phase corresponding to every development stage in the waterfall model.
  - The V-model is an extension of the waterfall model.
  - V model Testing is done in parallel to development.
  - It is also called a Validation and Verification Model.

The V-Model

- The left side of the model is Software Development Life Cycle - **SDLC**
- The right side of the model is Software Test Life Cycle - **STLC**
- The entire figure looks like a V, hence the name **V - model**

Requirements

- During this initial phase, system requirements and analysis are performed to determine the feature set and needs of users.
- During the requirements phase, acceptanc e tests are designed

System Design

- Utilizing feedback and user requirement documents created during the requirements phase, this next stage is used to generate a specification document that will outline all technical components such as the data layers, business logic, and so on.
- System Tests are also designed during this stage for later use.

Architecture Design

- During this stage, specifications are drawn up that detail how the application will link up all its various components, either internally or via outside integrations.
- Often this is referred to as high-level design.
- Integration tests are also developed during this time.

Module Design

This phase consists of all the low-level design for the system, including detailed specifications for how all functional, coded business logic will be implemented, such as models, components, interfaces, and so forth.

Unit tests should also be created during the module design phase.

Implementation/Coding

- At this point, halfway through the stages along the process, the actual coding and implementation occur.
- This period should allot for as much time as is necessary to convert all previously generated design and specification docs into a coded, functional system.
- This stage should be fully complete once the testing phases begin.

Unit Testing

Now the process moves back up the far side of the V-Model with inverse testing, starting with the unit tests developed during the module design phase.

Ideally, this phase should eliminate the vast majority of potential bugs and issues, and thus will be the lengthiest testing phase of the project.

Integration Testing

- Testing devised during the architecture design phase are executed here, ensuring that the system functions across all components and third-party integrations.

System Testing

- The tests created during system design are next executed, largely focusing on performance and regression testing.

Acceptance Testing

- Lastly, acceptance testing is the process of implementing all tests created during the initial requirements phase and should ensure that the system is functional in a live environment with actual data, ready for deployment.

Advantages of V-Model

Suited for Restricted Projects: Due to the stringent nature of the V-Model and its linear design, implementation, and testing phases, it's perhaps no wonder that the V-Model has been heavily adopted by the medical device industry in recent years.

Advantages of V-Model

- In situations where the project length and scope are well-defined, the technology is stable, and the documentation & design specifications are clear, the V-Model can be a great method.

Advantages of V-Model

- Ideal for Time Management: Along the same vein, V-Model is also well-suited for projects that must maintain a strict deadline and meet key milestone dates throughout the process.

Advantages of V-Model

- With fairly clear and well understood stages that the whole team can easily comprehend and prepare for, it is relatively simple to create a time line for the entire development life cycle, while generating milestones for each stage along the way.

Disadvantages of V-Model

Lacks Adaptability: Similar to the issues facing the traditional waterfall model on which the V-Model is based, the most problematic aspect to the V-Model is its inability to adapt to any necessary changes during the development life cycle.

Disadvantages of V-Model

- Timeline Restrictions: While not an inherent problem with the V-Model itself, the focus on testing at the end of the life cycle means that it's all too easy to be pigeonholed at the end of the project into performing tests in a rushed manner to meet a particular deadline or milestone.

Disadvantages of V-Model

- Ill-Suited for Lengthy Life Cycles: Like the waterfall model, the V-Model is completely linear and thus projects cannot be easily altered once the development train has left the station.
- V-Model is therefore poorly suited to handle long-term projects that may require many versions or constant updates/patches.

Disadvantages of V-Model

- Encourages 'Design-by-Committee' Development: While V-Model is certainly not the only development model to fall under this criticism, it cannot be denied that the strict and methodical nature of the V-Model and its various linear stages tend to emphasize a development cycle befitting managers and users, rather than developers and designers.

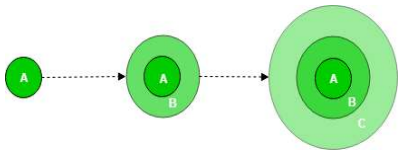
Disadvantages of V-Model

With a method like V-Model, it can be all too easy for project managers or others to overlook the vast complexities of software development in favor of trying to meet deadlines, or to simply feel overly confident in the process or current progress, based solely on what stage in the life cycle is actively being developed.

Incremental Model

- Incremental process model is also known as Successive version model.
- First, a simple working system implementing only a few basic features is built and then that is delivered to the customer.
- Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.

Incremental Model



A, B, C are modules of Software Product that are incrementally developed and delivered.

Life cycle activities –

- Requirements of Software are first broken down into several modules that can be incrementally constructed and delivered.
- At any time, the plan is made just for the next increment and not for any kind of long term plans.
- Therefore, it is easier to modify the version as per the need of the customer.

Life cycle activities –

- Development Team first undertakes to develop core features (these do not need services from other features) of the system.

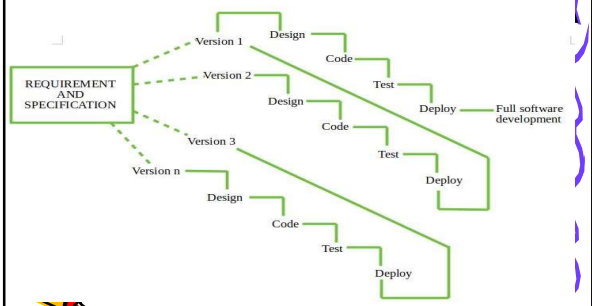
Life cycle activities –

- Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions.
- Each incremental version is usually developed using an iterative waterfall model of development.

Life cycle activities

- As each successive version of the software is constructed and delivered, now the feedback of the Customer is to be taken and these were then incorporated in the next version.
- Each version of the software have more additional features over the previous ones.

Life cycle activities



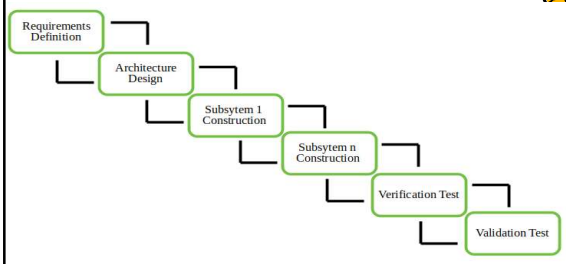
Life cycle activities

- After Requirements gathering and specification, requirements are then spitted into several different versions starting with version-1, in each successive increment, next version is constructed and then deployed at the customer site.
- After the last version (version n), it is now deployed at the client site.

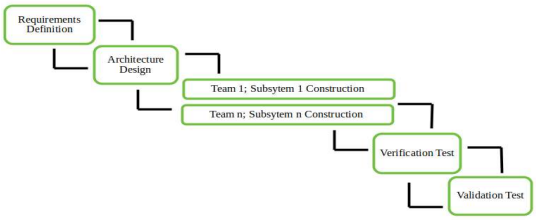
Types of Incremental model

- **Staged Delivery Model** – Construction of only one part of the project at a time.
- **Parallel Development Model** – Different subsystems are developed at the same time. It can decrease the calendar time needed for the development, i.e. TTM (Time to Market), if enough Resources are available.

Staged Delivery Model



Parallel Development Model

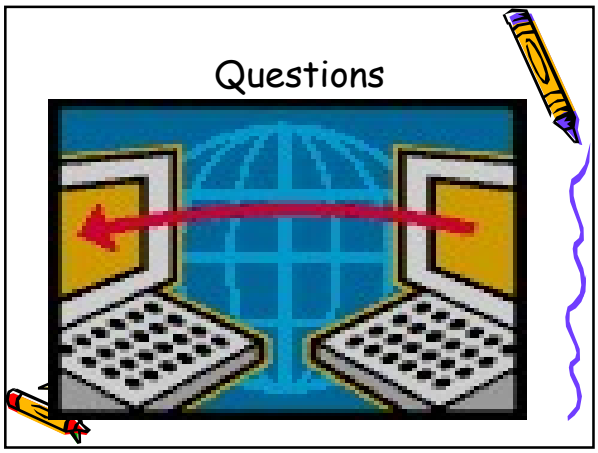


Advantages –

- **Customer value** can be delivered with each increment so system functionality is available earlier
- **Early increments** act as a prototype to help elicit requirements for later increments
- Lower risk of overall project failure
- **The highest priority system services** tend to receive the most testing

Disadvantages –

- Requires good planning and design.
- Total cost is not lower.
- Well defined module interfaces are required.


Questions