SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

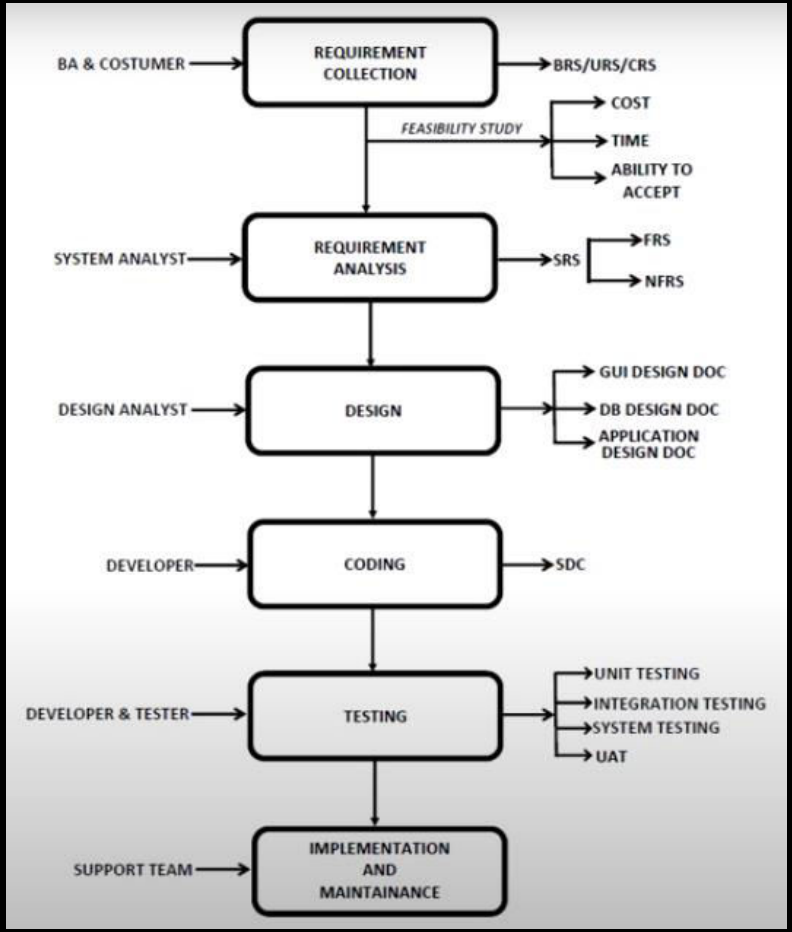
- ***Software development life cycle (SDLC)*** - a structured step-by-step approach for developing information systems.
- It is framework that describes the activities performed at each stage of a software development project.
- Typical activities include:
    - Determining budgets
    - Gathering business requirements
    - Designing models
    - Writing user documentation

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

SDLC has 7 phases:

1. Planning
2. Analysis
3. Design
4. Development
5. Testing
6. Implementation
7. Maintenance

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)



Phase 1: Planning

Planning phase - involves determining a solid plan for developing your Information System / Software / Project

Three primary planning activities:

1. Define the system to be developed
   - ***Critical success factor (CSF)*** - a factor simply critical to your organization's success

Phase 1: Planning

2. Set the project scope
   - ***Project scope*** **-** clearly defines the high-level system requirements
   - ***Scope creep*** **-** occurs when the scope of the project increases
   - ***Feature creep*** **-** occurs when developers add extra features that were not part of the initial requirements
   - ***Project scope document*** **-** a written definition of the project scope, and is usually no longer than a paragraph

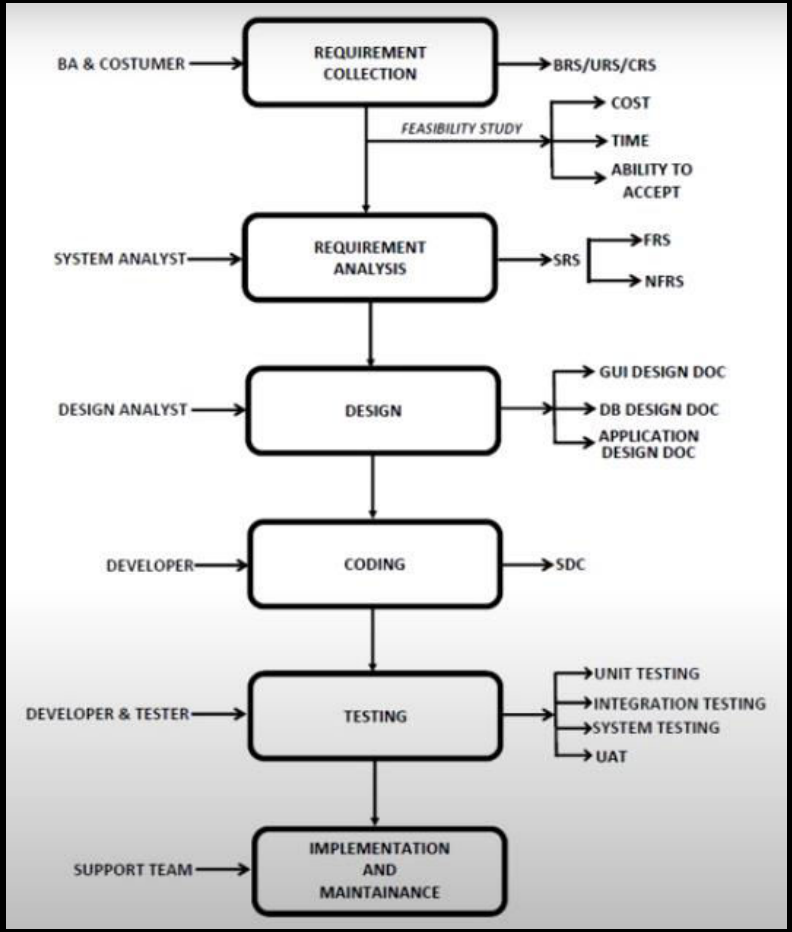
Phase 1: Planning

3. Develop the project plan including tasks, resources, and timeframes
   - ***Project plan*** - defines the what, when, and who questions of system development
   - ***Project manager*** **-** an individual who is an expert in project planning and management, defines and develops the project plan and tracks the plan to ensure all key project milestones are completed on time
   - ***Project milestones*** - represent key dates for which you need a certain group of activities performed

Phase 1: Planning

- Business Analyst and Customer will be involved in this phase
- BRS / URS / CRS document will be prepared in this phase
- There will be a feasibility study in the document prepared
- Before going the next phase there will be kick of meeting, in this both the BA & customer will discuss about the objective of the project

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)



Phase 2: Analysis

- ***Analysis phase*** **-** involves end users and IT specialists working together to gather, understand, and document the business requirements for the proposed system.

- System Analyst will be playing a major role in this phase

Phase 2: Analysis

Two primary analysis activities:

1. Gather the business requirements
   - ***Business requirements*** - the detailed set of knowledge worker requests that the system must meet in order to be successful.
   - ***Joint application development*** (***JAD***) - knowledge workers and IT specialists meet, sometimes for several days, to define or review the business requirements for the system

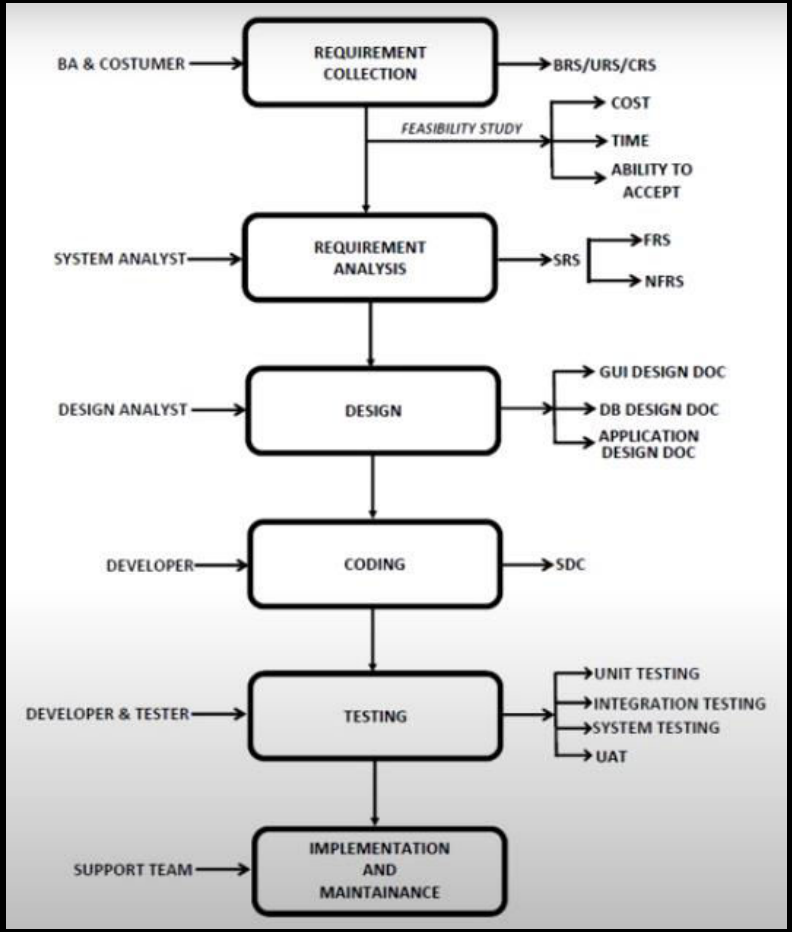
Phase 2: Analysis

2. Prioritize the requirements
   - ***Requirements definition document*** – prioritizes the business requirements and places them in a formal comprehensive document

Phase 2: Analysis

- System Analyst will be preparing SRS document
- SRS document will include both FRS and NFRS
  - FRS – checks the behavior of the system
  - NFRS – checks the behavior on certain conditions such as hardware, software, browser, multiuser, etc.,

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)



Phase 3: Design

Design phase - build a technical blueprint of how the proposed system will work

Two primary design activities:

1. Design the technical architecture
   - ***Technical architecture*** **-** defines the hardware, software, and telecommunications equipment required to run the system

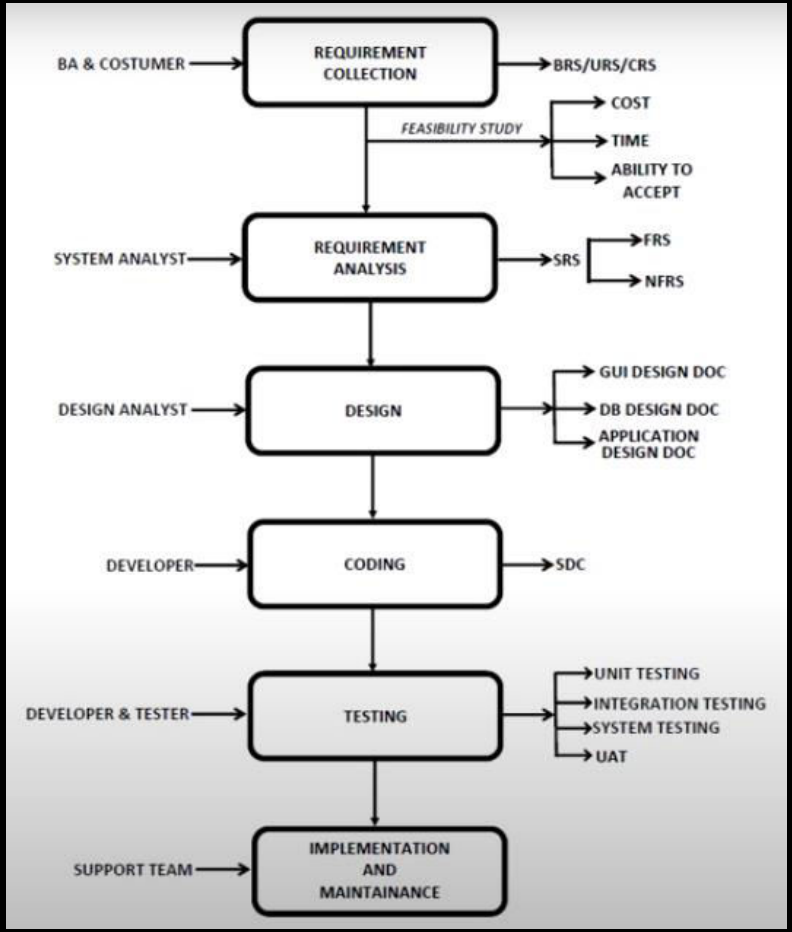
Phase 3: Design

2. Design system models
   - ***Modeling*** - the activity of drawing a graphical representation of a design
   - ***Graphical user interface (GUI)*** - the interface to an information system
   - ***GUI screen design*** - the ability to model the information system screens for an entire system

Phase 3: Design

- Design Architect will prepare 3 documents
  - GUI Design document
  - DB Design document
  - Application Design document

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)



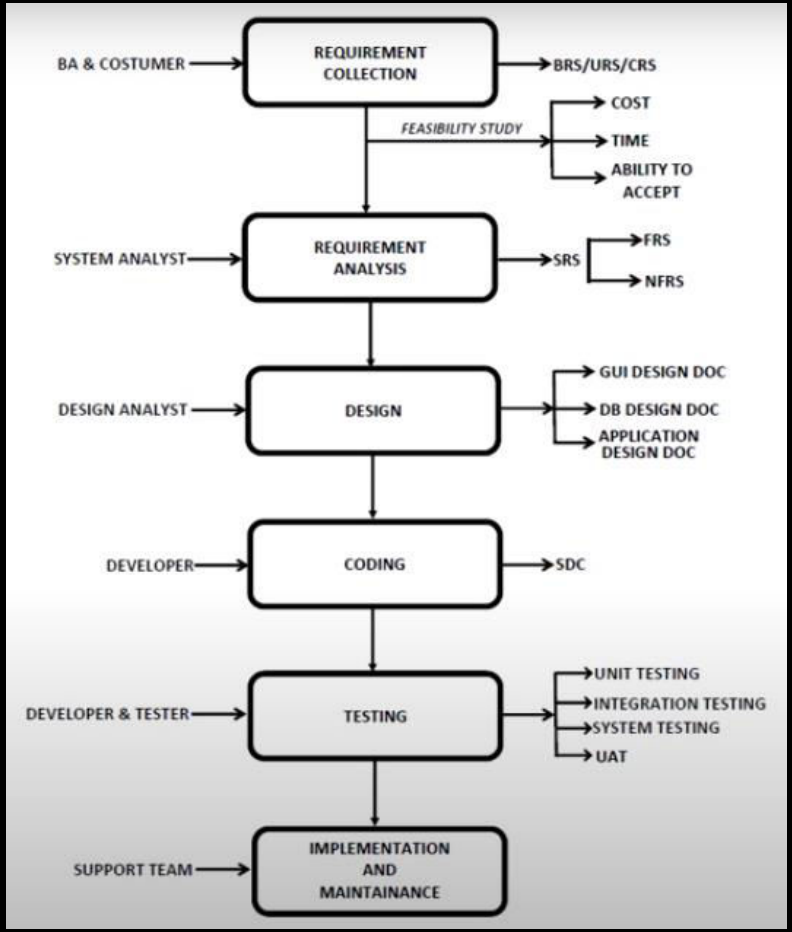
Phase 4: Development

Development phase **-** take all of your detailed design documents from the design phase and transform them into an actual system

Two primary development activities:

1. Build the technical architecture
2. Build the database and programs
   - Both of these activities are mostly performed by IT specialists

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)



Phase 5: Testing

Testing phase – verifies and validates that the system works and meets all of the business requirements defined in the analysis phase

Two primary testing activities:

1. Write the test conditions
2. Perform the testing of the system

Types of Testing

- There are two types of testing
  - Static Testing
  - Dynamic Testing

Static Testing

- Review
  - Management Review
  - Technical Review
  - Formal Review
  - Informal Review

Objective of Review

- To find defects in requirements
- To find defects in design
- To identify deviations in process
- To provide valuable suggestions to improve the process

Types of Reviews

- Management Review
- Technical Review
- Formal Review
- Informal Review

Types of Reviews

Management Review

- Conducted by the high level or mid level management to identify deviations between “Planned work” and the “actual work”, this is called Slippage

Technical Review

- This will be conducted by the team of technical experts to check the best approach for implementing a task

Types of Reviews

Formal Review

- This will be conducted with the help of Inspection and Audit
  - Inspection is conducted while the task is under execution
  - Audit is conducted after completing the task

Informal Review

- **This will be conducted without following any predefined procedure**
  - Code review is done by the developer on the coding standards
  - Peer review is done by colleagues

Dynamic Testing

- Dynamic Testing is conducted in 4 levels
  - Unit Testing
  - Integration Testing
  - System Testing
  - User Acceptance Testing

White Box Testing

Black Box Testing

4 levels of Dynamic Testing

- **Unit Testing** – tests individual unit of code, this is also know as component testing or module testing
- ***Integration testing*** – verifies that the units of code function correctly when integrated. The purpose of this **testing** is to expose faults in the interaction between **integrated** units.
- ***System testing* –** *validates both functional and non functional behaviour of the system*
- ***User Acceptance Testing(UAT)* –** is performed by the end user or the client to validate/accept the software system before moving the software application to the production environment. Alpha & Beta Testing are types of UAT

Unit Testing

- Individual units/components of a software are tested.
- Purpose is to validate that each unit of the software perform as designed. It usually has one or a few inputs and usually a single output.
- Unit Testing of software applications is done during the development of an application.
- Goal of Unit Testing is to isolate each part of the program and show that the individual parts are correct.
- Unit Testing is usually performed by the developer who writes the code.

Integration Testing

- Individual units are combined and tested as a group.
- The purpose of this testing is to expose faults in the interaction between integrated units.
- Test drivers and test stubs are used to assist in Integration Testing.
- Integration testing can be done at UI level, component level and system level..

System Testing

- It is where the complete and integrated software is tested.
- Purpose of this test is to evaluate the system's compliance with the specified requirements.
- System Testing is the third level of software testing performed after Integration Testing
- Normally, independent Testers perform System Testing.
- Also known as End-To-End testing.

User Acceptance Testing

- Alpha
- Beta

Alpha Testing

- Alpha testing is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or the public.
- The aim is to carry out the tasks that a typical user might perform.
- Alpha testing is carried out in a lab environment and usually, the testers are internal employees of the organization.

Beta Testing

- It is where a system is tested for acceptability.
- Purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
- Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.
- Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external User Acceptance Testing.

Alpha Testing	Beta Testing
Alpha testing performed by Testers who are usually internal employees of the organization	Beta testing is performed by Clients or End Users who are not employees of the organization
Alpha Testing performed at developer's site	Beta testing is performed at a client location or end user of the product
Reliability and Security Testing are not performed in-depth Alpha Testing	Reliability, Security, Robustness are checked during Beta Testing
Alpha testing requires a lab environment or testing environment	Beta testing doesn't require any lab environment or testing environment. The software is made available to the public and is said to be real time environment
Long execution cycle may be required for Alpha testing	Only a few weeks of execution are required for Beta testing
Critical issues or fixes can be addressed by developers immediately in Alpha testing	Most of the issues or feedback is collected from Beta testing will be implemented in future versions of the product
Alpha testing is to ensure the quality of the product before moving to Beta testing	Beta testing also concentrates on the quality of the product, but gathers users input on the product and ensures that the product is ready for real time users.

Regression Testing

- Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- It is a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
- This testing is done to make sure that new code changes should not have side effects on the existing functionalities.
- It ensures that old code still works once the new code changes.

Phase 6: Implementation

***Implementation phase* -** distribute the system to all of the knowledge workers and they begin using the system to perform their everyday jobs

Two primary implementation activities

1. Write detailed user documentation
   - ***User documentation*** - highlights how to use the system

Phase 6: Implementation

2. Provide training for the system users
   - ***Online training*** - runs over the Internet or off a CD-ROM
   - ***Workshop training*** **-** is held in a classroom environment and lead by an instructor

Phase 6: Implementation

Choose the right implementation method

- ***Parallel implementation*** – use both the old and new system simultaneously
- ***Plunge implementation*** – discard the old system completely and use the new
- ***Pilot implementation*** – start with small groups of people on the new system and gradually add more users
- ***Phased implementation*** – implement the new system in phases

Phase 7: Maintenance

Maintenance phase - monitor and support the new system to ensure it continues to meet the business goals

Two primary maintenance activities:

1. Build a help desk to support the system users
   - ***Help desk*** - a group of people who responds to knowledge workers' questions
2. Provide an environment to support system changes

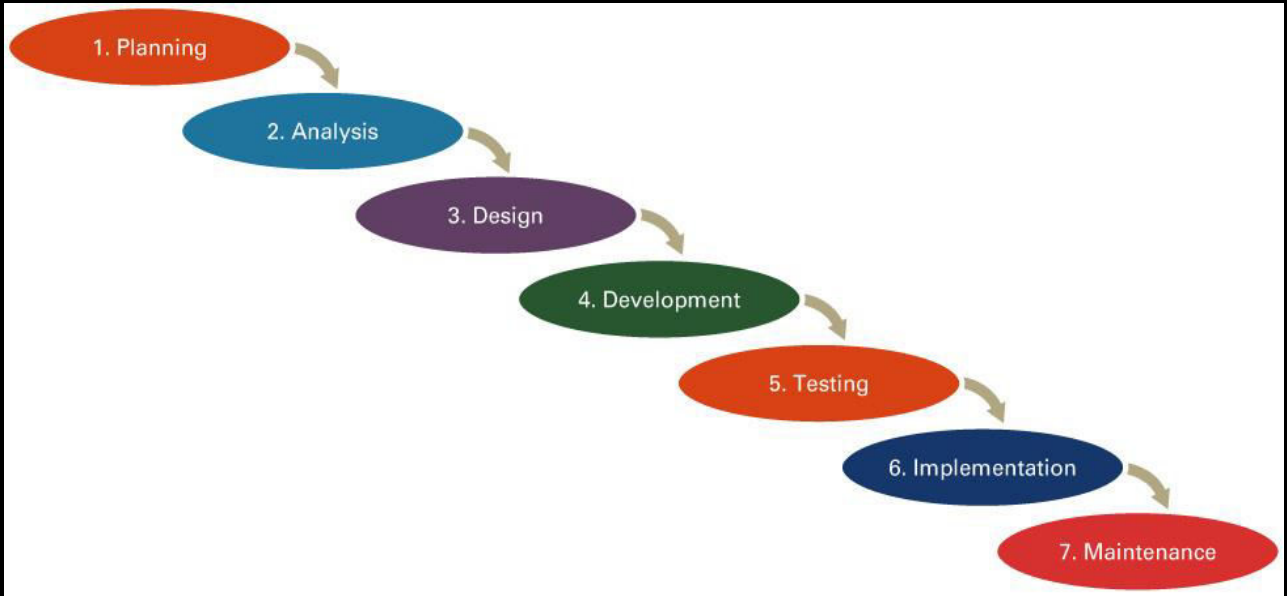
SOFTWARE DEVELOPMENT METHODOLOGIES

Developers have different development methodologies:

- Waterfall Methodology
- V-Shaped Model
- Spiral Model
- Rapid application development (RAD)
- Incremental Model
- Agile Methodology

Waterfall Methodology

Waterfall methodology - a sequential, activity-based process in which each phase in the SDLC is performed sequentially from planning through implementation



Waterfall Model

- **Planning** – **d**efines the system to be developed
- **Analysis** – defines needed information, function, behavior, performance and interfaces.
- **Design** – data structures, software architecture, interface representations, algorithmic details.
- **Development** – Developing program and database
- **Testing** – Testing for both functional and non functional behaviour
- **Implementation** – source code, database, user documentation, testing.
- **Maintenance** – Maintaining and providing support to the project

Waterfall Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

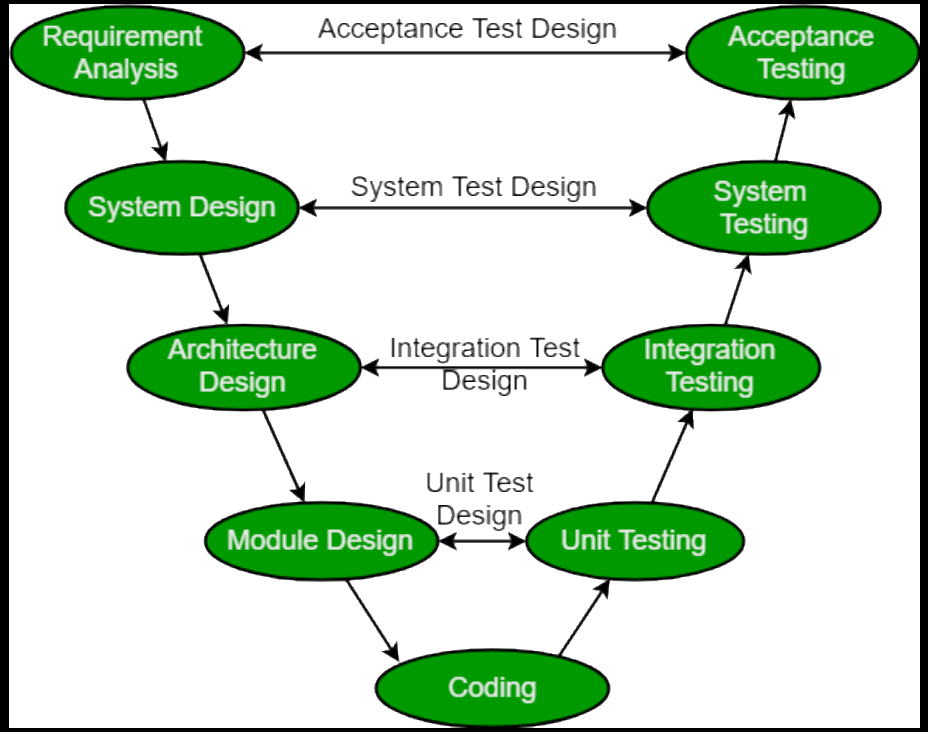
Waterfall Weakness

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

When to use the Waterfall Model

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

V-Shaped SDLC Model



- A variant of the Waterfall that emphasizes the verification and validation of the product.
- Testing of the product is planned in parallel with a corresponding phase of development

V-Shaped Strengths

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

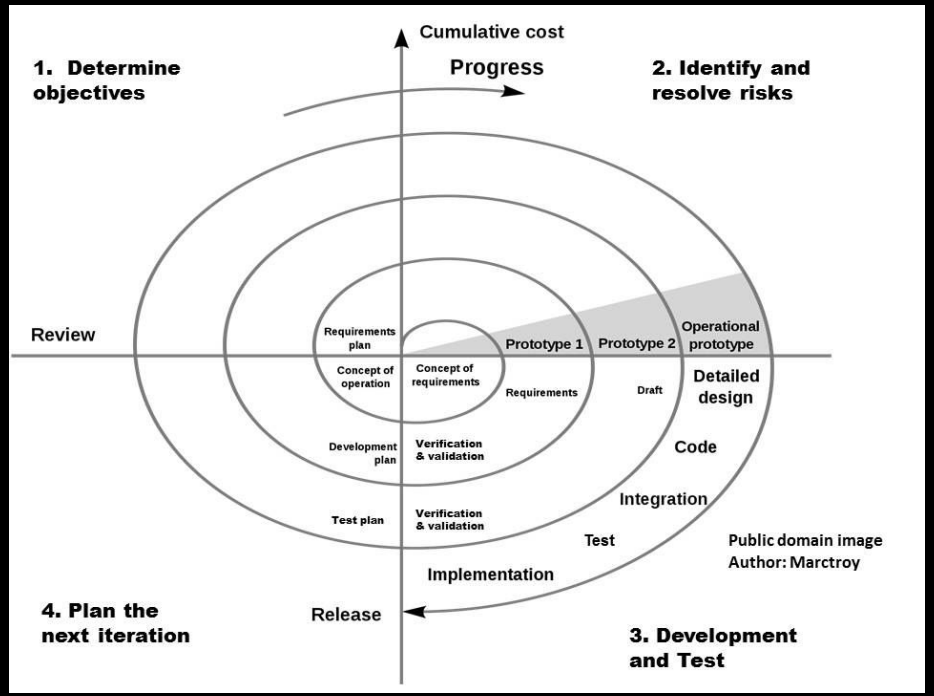
V-Shaped Weaknesses

- Does not easily handle concurrent events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

When to use the V-Shaped Model

- Excellent choice for systems requiring high reliability – hospital patient control applications
- All requirements are known up-front
- Solution and technology are known

Spiral SDLC Model



- Each cycle involves the same sequence of steps as the waterfall process model

Spiral Model Strengths

- Provides early indication of insurmountable risks, without much cost
- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

Spiral Model Weaknesses

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

When to use Spiral Model

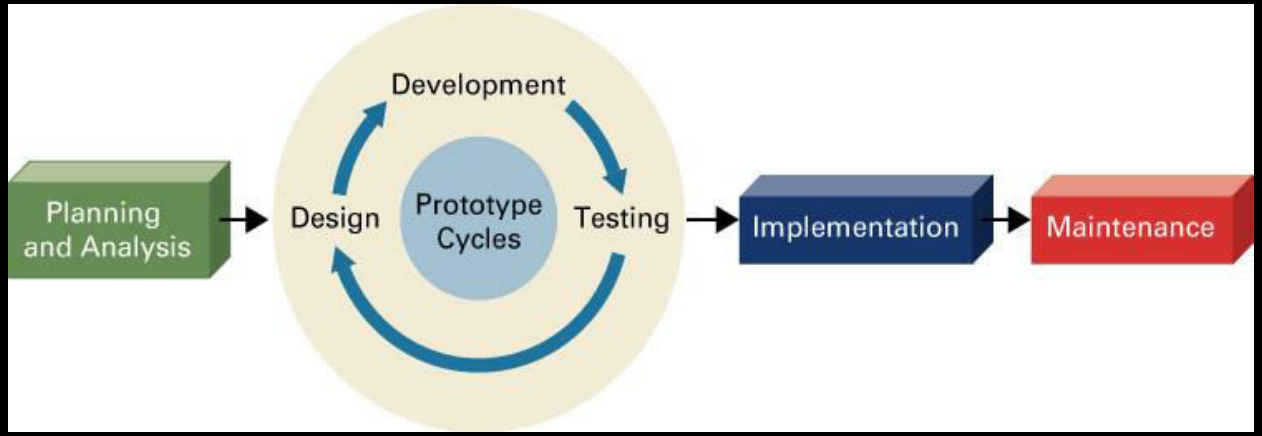
- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- Significant changes are expected (research and exploration)

Rapid Application Development (RAD)

Rapid application development (RAD) (also called ***rapid prototyping***) - emphasizes extensive user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the systems development process

- ***Prototype*** - a smaller-scale, representation, or working model of the user's requirements or a proposed design for an information system

Rapid Application Development (RAD)



RAD Strengths

- Reduced cycle time and improved productivity with fewer people means lower costs
- Time-box approach mitigates cost and schedule risk
- Customer involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs
- Focus moves from documentation to code (WYSIWYG).
- Uses modeling concepts to capture information about business, data, and processes.

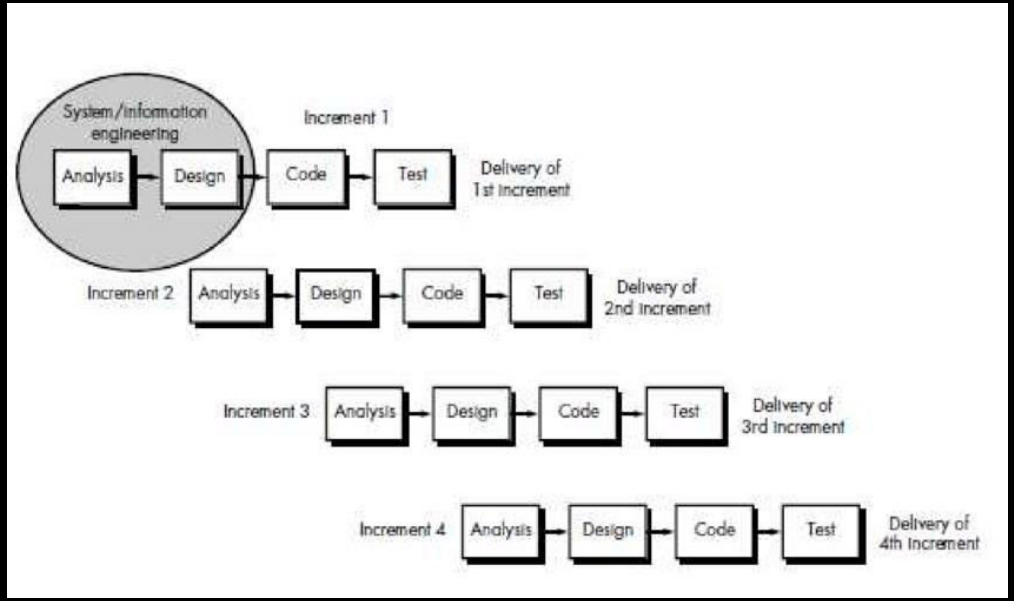
RAD Weaknesses

- Accelerated development process must give quick responses to the user
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

When to use RAD

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- Functionality delivered in increments
- High performance not required
- Low technical risks
- System can be modularized

Incremental SDLC Model



- Construct a partial implementation of a total system
- Then slowly add increased functionality
- The incremental model prioritizes requirements of the system and then implements them in groups.
- Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented.

Incremental Model Strengths

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses "divide and conquer" breakdown of tasks
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

Incremental Model Weaknesses

- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower

When to use the Incremental Model

- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology

Agile Method

- AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project.
- In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model
- This methodology is one of the simplest and effective processes to turn a vision for a business need into software solutions
- Agile is a term used to describe software development approaches that employ continual planning, learning, improvement, team collaboration, evolutionary development, and early delivery.

Agile Methods

The agile software development emphasizes on four core values.

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

Agile Methodology

- Customer gives outline to the project requirement usually one page document, called Epic
- The Product Manager will create “User Story” from the customer
- The “User Story” will be provided to the development team
- The development team will create Project Plan based on the User Story
- This model doesn’t have BRS
- This is mostly used for short term project

Agile Methodology

- Project will be divided in to modules for a short duration
- This short duration of project is know as Sprint
- There will be a Interim Project Release at the end of every Sprint
- Sprint duration can be 2weeks to 2 months based on the project

Members

- Scrum Master
  - Master is responsible for setting up the team, sprint meeting and removes obstacles to progress
- Product Owner
  - The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
- Scrum Team
  - Team manages its own work and organizes the work to complete the sprint or cycle

Sprint Meeting

- Sprint Planning Meeting
- Sprint Demo Meeting
- Sprint Retrospective Meeting
- Daily Standup call Meeting

Sprint Planning Meeting

- Task will be listed
- Everyone will take their own task
- Task will be allotted based on the effort card

  Eg: Fibonacci , T shirt Size

Sprint Demo Meeting

- Demo is provided to the customer
- This will happen at the Interim release

Sprint Retrospective Meeting

- Team cooperation will be discussed
- SWOT analysis will be made based on team work
- Based on this the team will be team may be changed or re shuffled

Daily Standup Call Meeting

- It must be a 10 minutes meeting
- Following things will be discussed:
  - What I did yesterday ?
  - Do I have any issue ?
  - What is my plan today?

Agile Disadvantages

- Very Less documentation
- Suitable only for short term projects