PORTFOLIO WEBSITE FOR NEONTECH

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Shrikanta Paul
Full Stack Web Developer

Contents

1. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

SDLC is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase. SDLC stands for Software Development Life Cycle and is also referred to as the Application Development life-cycle.

1.1 WATERFALL MODEL

The waterfall model is a linear, sequential approach to the software development lifecycle (SDLC) that is popular in software engineering and product development.

The waterfall model uses a logical progression of SDLC steps for a project, similar to the direction water flows over the edge of a cliff. It sets distinct endpoints or goals for each phase of development. Those endpoints or goals can't be revisited after their completion.

When to Use the waterfall Model?

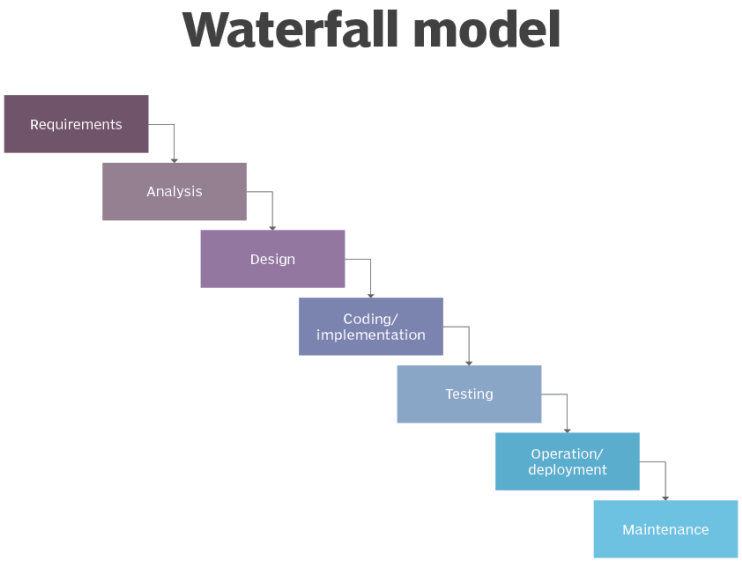
Projects based on the waterfall model are well defined, predictable and have specific documentation. They also have the following characteristics:

1. Fixed requirements;
2. Ample resources;
3. An established timeline;
4. Well-understood technology; and
5. Unlikely to require significant changes.

Phases of the waterfall model

When used for a software development process, the waterfall methodology has seven stages:

- **Requirements:** Potential requirements, deadlines and guidelines for the project are analyzed and placed into a formal requirements document, also called a functional specification. This stage of development defines and plans the project without mentioning specific processes.
- **Analysis:** The system specifications are analyzed to generate product models and business logic to guide production. This is also when financial and technical resources are audited for feasibility.
- **Design:** A design specification document is created to outline technical design requirements, such as the programming language, hardware, data sources, architecture and services.
- **Coding and implementation:** The source code is developed using the models, logic and requirement specifications designated in the prior phases. Typically, the system is coded in smaller components, or units, before being put together.
- **Testing:** This is when quality assurance, unit, system and beta tests identify issues that must be resolved. This may cause a forced repeat of the coding stage for debugging. If the system passes integration and testing, the waterfall continues forward.
- **Operation and deployment:** The product or application is deemed fully functional and is deployed to a live environment.
- **Maintenance:** Corrective, adaptive and perfective maintenance is carried out indefinitely to improve, update and enhance the product and its functionality. This could include releasing patch updates and new versions.



Advantages	Disadvantages
1. enables large or changing teams to move toward a common goal that's been defined in the requirements stage; 2. forces structured, disciplined organization; 3. simplifies understanding, following and arranging tasks; 4. facilitates departmentalization and managerial control based on the schedule or deadlines; 5. reinforces good coding habits to define before implementing design and then code; 6. enables early system design and specification changes to be easily done; and 7. clearly defines milestones and deadlines.	1. Design isn't adaptive; when a flaw is found, the entire process often needs to start over. 2. Method doesn't incorporate midprocess user or client feedback, and makes changes based on results. 3. Waterfall model delays testing until the end of the development lifecycle. 4. It doesn't consider error correction. 5. The methodology doesn't handle requests for changes, scope adjustments and updates well. 6. Waterfall doesn't let processes overlap for simultaneous work on different phases, reducing overall efficiency. 7. No working product is available until the later stages of the project lifecycle. 8. Waterfall isn't ideal for complex, high-risk ongoing projects.

1.2 ITERATIVE MODEL

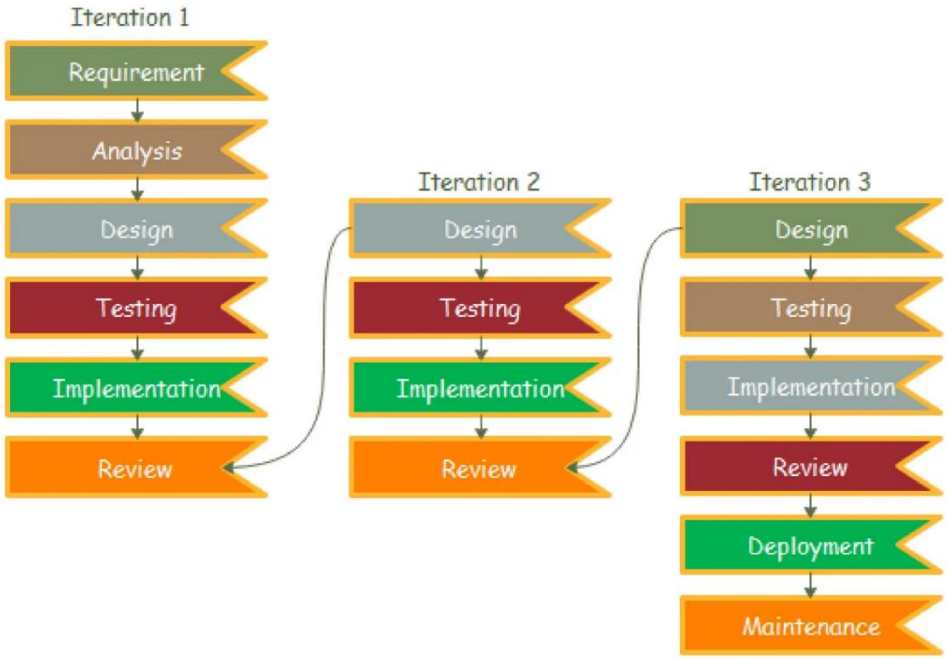
The iterative model is a software development life cycle (SDLC) approach in which initial development work is carried out based on well-stated basic requirements, and successive enhancements are added to this base piece of software through iterations until the final system is built. We get a working piece of software very early in the lifecycle because the iterative model begins with a simple execution of a small collection of software requirements, which iteratively improves the evolving variants until the entire system is executed and ready to be redistributed. Every Iterative model release is created over a certain and predetermined time period known as iteration. Bugs and errors from the previous iteration do not propagate to the next iteration, and this model is flexible enough to incorporate customer feedback in every iteration.

When to Use the Iterative Model?

1. The iterative model is suitable for the following use cases:
2. When the project is huge, it can be broken down into smaller pieces and developed by adhering to the iterative paradigm.
3. When the requirements can be understood and defined clearly at the beginning of the project.
4. When there is a need to incorporate customer feedback at every stage - The major requirements are laid down initially; however, as the development process progresses, some functionalities are altered, and additions are suggested.
5. While working on the project, the development team is experimenting with and learning new technology.

Phases of Iterative Model

- **Requirement Gathering & Analysis:** The business requirements are gathered during this phase of the iterative model. Then, an analyst determines whether they can be met within the financial constraints. This phase details the business needs, and system information (hardware or software) is acquired and assessed for viability.
- **Design:** During this phase of the iterative model, the project team receives the complete list of criteria for starting work in a specific direction. Then, they use various diagrams, like a data flow diagram, class diagram, activity diagram, state transition diagram, and so on, to gain explicit knowledge of the program design and to help them progress with development. Based on their investigation, developers provide viable solutions. Furthermore, the project's scale and criticality are crucial factors in deciding the complexity of the design for the project.
- **Implementation:** At this point in the project, according to the iterative model, the actual coding of the system begins. This stage will be influenced by the Design Stage's analysis and design. All needs, planning, and design plans have been carried out. The chosen design will be implemented by the developer using predefined coding and metrics standards. They must implement a unit test at each stage of code development and should strive to produce a fully functional, testable system for that iteration. The complexity of work and time spent on this iteration will vary depending on the project.
- **Testing:** This stage entails comparing the current build iteration to a set of rules and norms to determine whether or not it fits them. This sort of testing includes performance testing, stress testing, security testing, requirements testing, usability testing, multi-site testing, disaster recovery testing, and so on. The tester can create new test cases or reuse those from previous releases, but testing is a key priority because any failures would affect the software's specification, affecting the business. We can also check in with the project stakeholders to perform some tests and get their input. A developer or tester must guarantee that correcting one bug does not result in the appearance of new bugs in the system.
- **Deployment:** After completing all the phases, the software is deployed to its work environment.
- **Review:** In this phase, after the product deployment, we check the behavior and validity of the deployed product. And if any errors are found, the process starts again from requirement gathering.
- **Maintenance:** In the maintenance phase, after software deployment in the working environment, there may be some bug fixes or new updates required.



Advantages	Disadvantages
1. Rapid Issue Identification: Quickly identifies design or functionality faults, allowing for prompt corrective actions within budget constraints. 2. Early Working Product: Unlike the waterfall model, a working product is available early in the lifecycle. 3. Bugs Prevention: Detects errors and prevents their propagation to subsequent iterations by thoroughly testing each iteration's output. 4. Flexible Requirements: Allows for changes in requirements with minimal cost, though structural constraints may limit some accommodations. 5. Customer Feedback: Incorporates customer feedback in every iteration, facilitating swift implementation. 6. Efficiency: Emphasizes design and development over extensive documentation, saving time.	1. System Structure Issues: Incomplete requirement gathering upfront may lead to system architecture problems. Repeated design changes may occur due to faulty initial requirements. 2. Limited for Frequent Changes: Not suitable for projects with frequently shifting requirements, as it may disrupt the iterative process. 3. Not Ideal for Small Projects: Impractical for small projects, making it challenging to break them down into smaller iterations. 4. Resource-Intensive: Requires highly trained resources for analysis, making it more resource-intensive than the waterfall model. 5. Complex Management: Managing the entire iterative procedure can be challenging.

1.3 SPIRAL MODEL

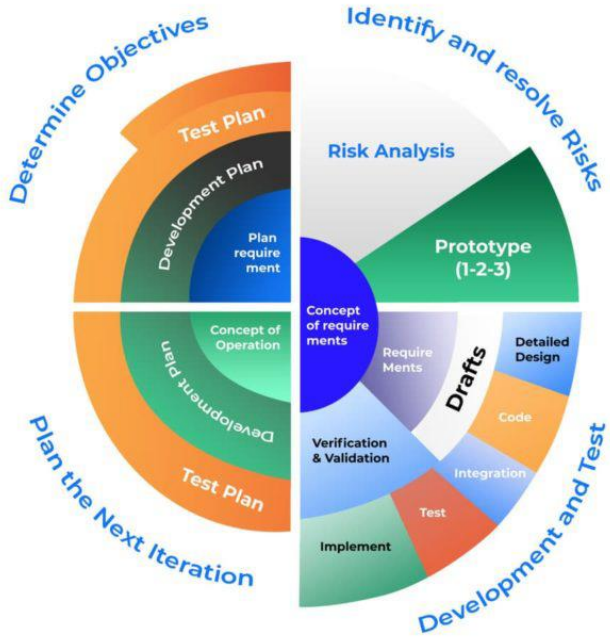
Spiral Model is a risk-driven software development process model. It is a combination of waterfall model and iterative model. Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.

When to use Spiral Model?

1. A Spiral model in software engineering is used when project is large
2. When releases are required to be frequent, spiral methodology is used
3. When creation of a prototype is applicable
4. When risk and costs evaluation is important
5. Spiral methodology is useful for medium to high-risk projects
6. When requirements are unclear and complex, Spiral model in SDLC is useful
7. When changes may require at any time
8. When long term project commitment is not feasible due to changes in economic priorities
9.

Phases of Spiral Model

- **Planning:** It includes estimating the cost, schedule and resources for the iteration. It also involves understanding the system requirements for continuous communication between the system analyst and the customer
- **Risk Analysis**: Identification of potential risk is done while risk mitigation strategy is planned and finalized
- **Engineering:** It includes testing, coding and deploying software at the customer site
- **Evaluation:** Evaluation of software by the customer. Also, includes identifying and monitoring risks such as schedule slippage and cost overrun



Advantages	Disadvantages
1. Additional functionality or changes can be done at a later stage 2. Cost estimation becomes easy as the prototype building is done in small fragments 3. Continuous or repeated development helps in risk management 4. Development is fast and features are added in a systematic way in Spiral development 5. There is always a space for customer feedback	1. Risk of not meeting the schedule or budget 2. Spiral development works best for large projects only also demands risk assessment expertise 3. For its smooth operation spiral model protocol needs to be followed strictly 4. Documentation is more as it has intermediate phases 5. Spiral software development is not advisable for smaller project, it might cost them a lot

1.4 V-MODEL

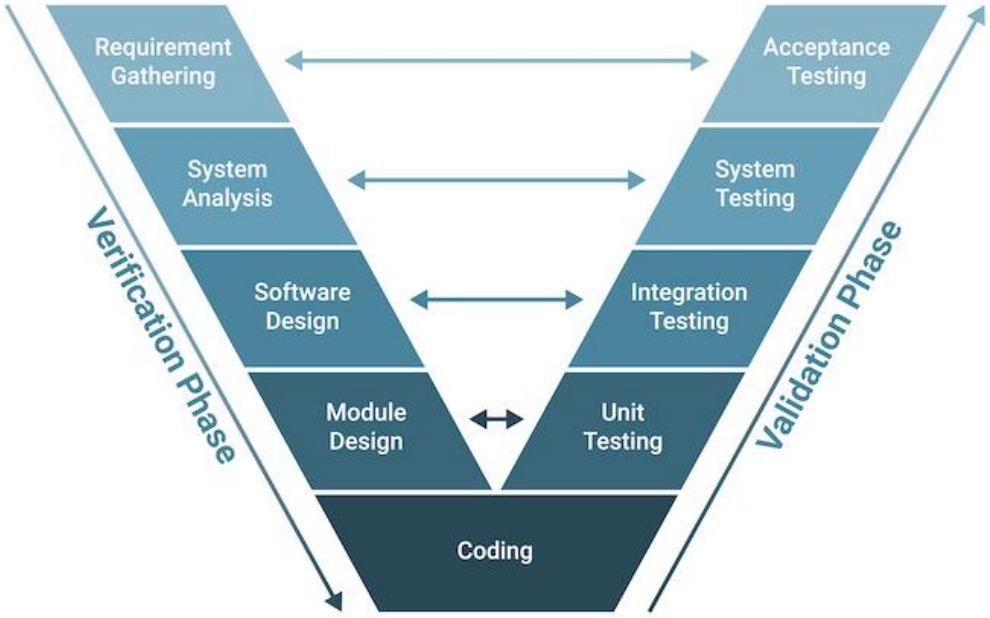
The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

When to use V-Model?

1. Requirements are well defined, clearly documented and fixed.
2. Product definition is stable.
3. Technology is not dynamic and is well understood by the project team.
4. There are no ambiguous or undefined requirements.
5. The project is short.

Phases of V-Model

- **Business Requirement Analysis:** Understand customer expectations and detailed requirements. Plan acceptance tests based on these requirements.
- **System Design:** Design the complete system including hardware and communication setup. Develop the system test plan.
- **Architectural Design:** Specify architectural specifications, module breakdown, and data transfer mechanisms. Prepare for integration tests.
- **Module Design:** Create detailed internal designs for system modules. Design unit tests for early bug detection.
- **Coding Phase:** Write code for system modules following coding guidelines and standards. Conduct code reviews and optimize for performance.
- **Validation Phases:**
  - ✓ Unit Testing: Execute unit tests designed in the module design phase to catch code-level issues.
  - ✓ Integration Testing: Test module coexistence and communication.
  - ✓ System Testing: Verify the entire system's functionality and external communication.
  - ✓ Acceptance Testing: Test the product in the user environment, uncovering compatibility and non-functional issues.



Advantages	Disadvantages
1. This is a highly-disciplined model and Phases are completed one at a time. 2. Works well for smaller projects where requirements are very well understood. 3. Simple and easy to understand and use. 4. Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.	1. High risk and uncertainty. 2. Not a good model for complex and object-oriented projects. 3. Poor model for long and ongoing projects. 4. Not suitable for the projects where requirements are at a moderate to high risk of changing. 5. Once an application is in the testing stage, it is difficult to go back and change a functionality. 6. No working software is produced until late during the life cycle.

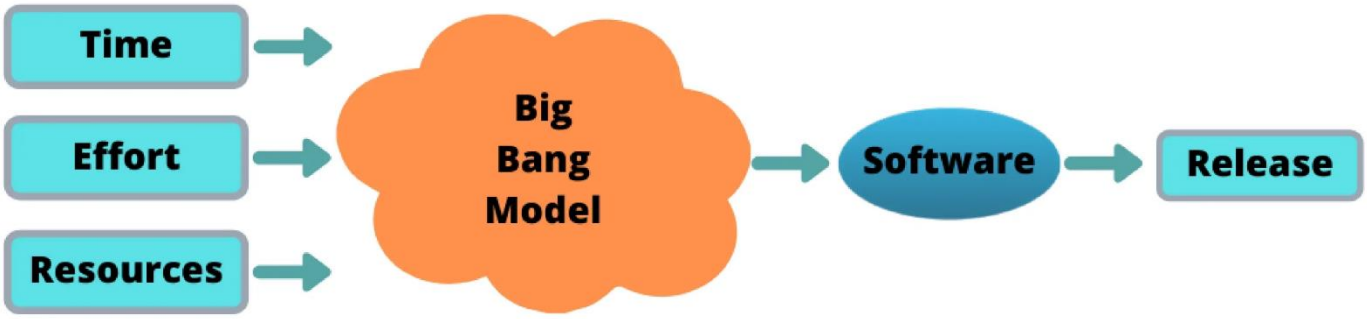
1.5 BIG BANG MODEL

In Big Bang Model developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

This model is ideal for small projects like aca demic projects or practical projects. One or two developers can work together on this model.

When to use Big Bang Model?

As we discussed above, this model is required when this project is small like an academic project or a practical project. This method is also used when the size of the developer team is small and when requirements are not defined, and the release date is not confirmed or given by the customer.



Advantages	Disadvantages
1. There is no planning required. 2. Simple Model. 3. Few resources required. 4. Easy to manage. 5. Flexible for developers.	1. There are high risk and uncertainty. 2. Not acceptable for a large project. 3. If requirements are not clear that can cause very expensive.

1.6 AGILE MODEL

The meaning of Agile is swift or versatile."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.

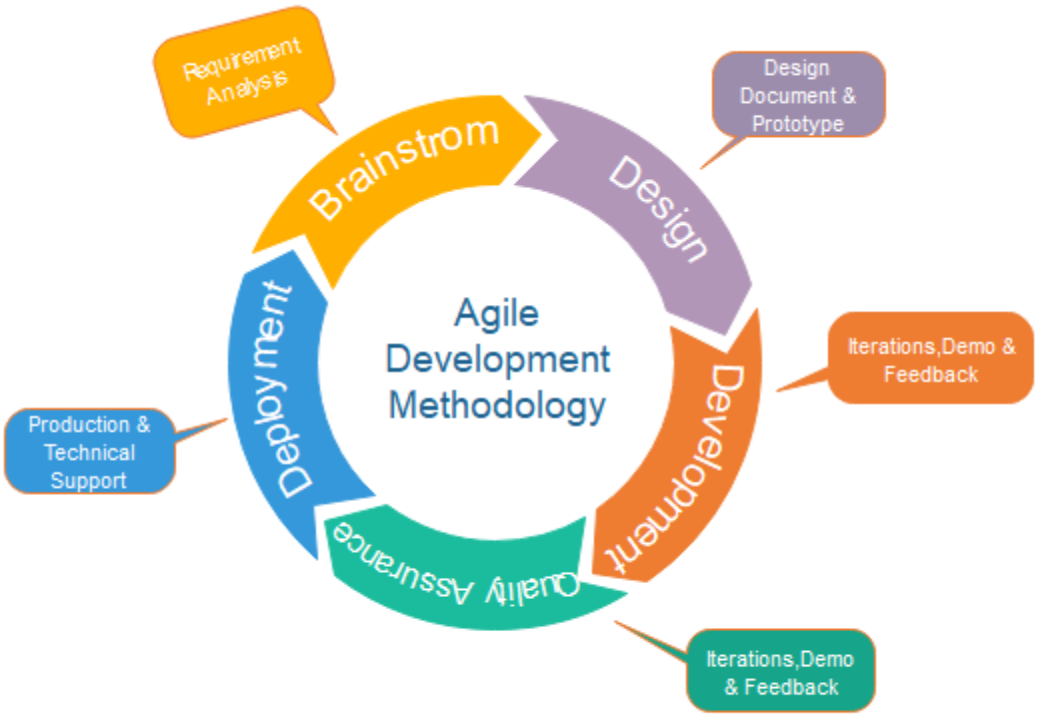
When to use the Agile Model?

1. When frequent changes are required.
2. When a highly qualified and experienced team is available.
3. When a customer is ready to have a meeting with a software team all the time.
4. When project size is small.

Phases of Agile Model

Following are the phases in the Agile model are as follows:

- **Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
- **Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.
- **Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.
- **Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
- **Deployment:** In this phase, the team issues a product for the user's work environment.
- **Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.



Advantages	Disadvantages
1. Frequent Delivery 2. Face-to-Face Communication with clients. 3. Efficient design and fulfils the business requirement. 4. Anytime changes are acceptable. 5. It reduces total development time.	1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members. 2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

1.7 RAD MODEL

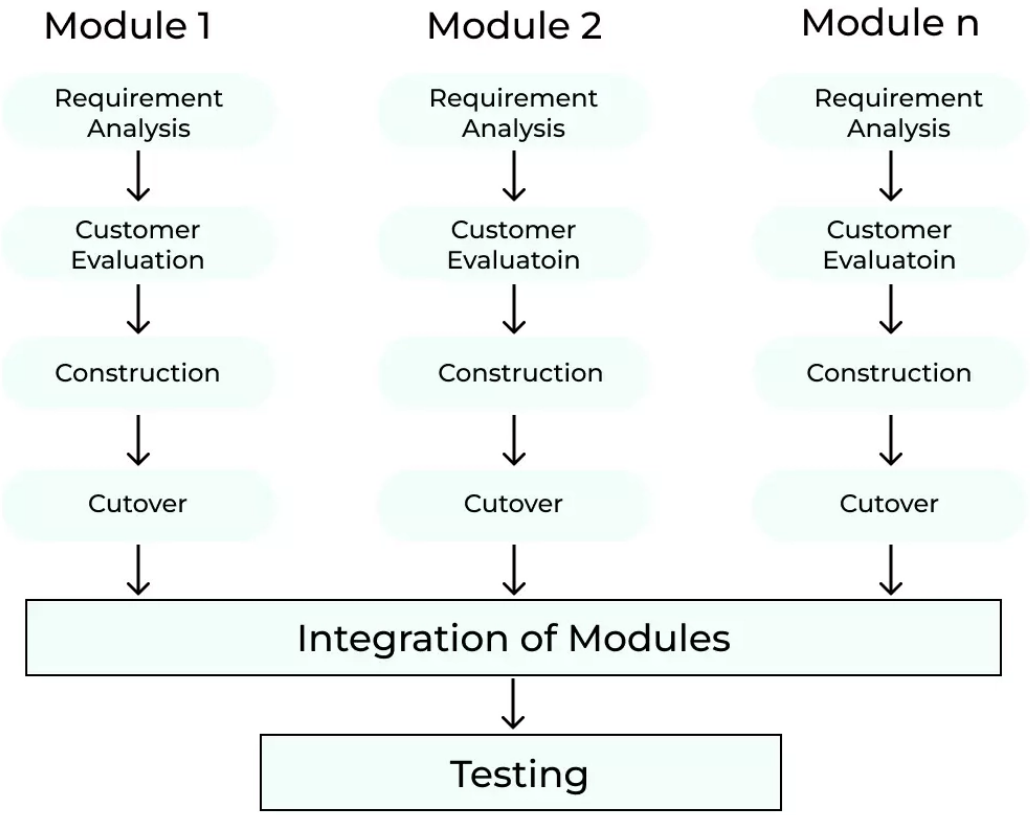
RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element-based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

When to use RAD Model?

1. When the system should need to create the project that modularizes in a short span time (2-3 months).
2. When the requirements are well-known.
3. When the technical risk is limited.
4. When there's a necessity to make a system, which modularized in 2-3 months of period.
5. It should be used only if the budget allows the use of automatic code generating tools.

Phases of RAD Model

- **Business Modelling:** The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.
- **Data Modelling:** The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.
- **Process Modelling:** The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
- **Application Generation:** Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.
- **Testing & Turnover:** Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.



Advantages	Disadvantages
1. This model is flexible for change. 2. In this model, changes are adoptable. 3. Each phase in RAD brings highest priority functionality to the customer. 4. It reduced development time. 5. It increases the reusability of features.	1. It required highly skilled designers. 2. All application is not compatible with RAD. 3. For smaller projects, we cannot use the RAD model. 4. On the high technical risk, it's not suitable. 5. Required user involvement.

1.8 PROTOTYPE MODEL

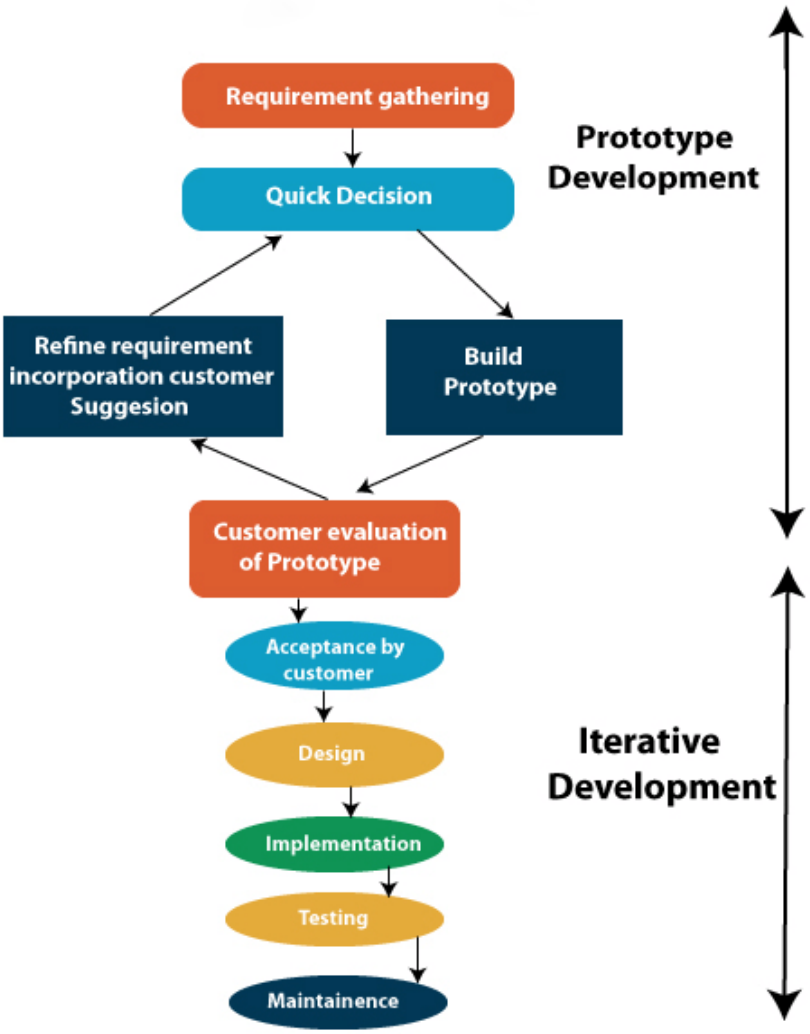
The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.

When to use Prototype Model?

1. The Prototyping Model should be used when the requirements of the product are not clearly understood or are unstable.
2. The prototyping model can also be used if requirements are changing quickly.
3. This model can be successfully used for developing user interfaces, high-technology software-intensive systems, and systems with complex algorithms and interfaces.
4. The prototyping Model is also a very good choice to demonstrate the technical feasibility of the product.

Phases of Prototype Model

- Requirement Gathering and Analyst
- Quick Decision
- Build a Prototype
- Assessment or User Evaluation
- Prototype Refinement
- Engineer Product



Advantages	Disadvantages
1. Reduce the risk of incorrect user requirement 2. Good where requirement is changing/uncommitted 3. Regular visible process aids management 4. Support early product marketing 5. Reduce Maintenance cost. 6. Errors can be detected much earlier as the system is made side by side.	1. An unstable/badly implemented prototype often becomes the final product. 2. Require extensive customer collaboration 3. Costs customer money 4. Needs committed customer 5. Difficult to finish if customer withdraw 6. May be too customer specific, no broad market 7. Difficult to know how long the project will last. 8. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback. 9. Prototyping tools are expensive. 10. Special tools & techniques are required to build a prototype. 11. It is a time-consuming process.

2. THE SUITABLE SDLC MODEL FOR NEONTECH

2.1 THE MODEL USED

A Hybrid SDLC (Software Development Life Cycle) model can be a suitable choice for NeonTech's website development, especially as they have specific project requirements that could benefit from a combination of different SDLC approaches. Here's a suggested Hybrid Model tailored for NeonTech's website development:

- **Requirements Gathering (Waterfall Phase)**
  - Begin with a requirements gathering phase following a Waterfall approach. Engage with the client to comprehensively gather and document the website's initial requirements, goals, and constraints.
  - This phase will result in a detailed project scope, requirements document, and a clear understanding of the website's primary features and functionalities.
- **Agile Development Phase (Iterative and Incremental)**
  - After the initial requirements are gathered, transition into an Agile development phase, specifically using an iterative and incremental approach.
  - Divide the project into smaller, manageable iterations (sprints), typically lasting 2-4 weeks each.
  - Develop and deliver functional increments of the website in each iteration, focusing on high-priority features first.
  - Engage with the client or stakeholders regularly to gather feedback and make adjustments as needed.
  - Prioritize features based on user needs and market trends, allowing for flexibility in feature development.
- **Integration and Testing (Hybrid Phase)**
  - During the Agile development phase, ensure continuous integration and testing to maintain the website's stability and quality.
  - Integrate the newly developed features and conduct integration testing to verify that all components work together seamlessly.
  - Use automated testing tools to streamline the testing process and identify issues early.
- **User Acceptance Testing (Waterfall Phase)**
  - Once all major features are developed and integrated, transition back to a Waterfall-like phase for User Acceptance Testing (UAT).
  - Allow the client or end-users to thoroughly test the website's functionality in a production-like environment.
  - Ensure that the website meets the client's expectations and resolves any issues or bugs discovered during UAT.
- **Deployment and Maintenance (Hybrid Phase)**
  - After successful UAT, deploy the website to a live environment.
  - Continue with Agile practices for ongoing maintenance and updates, with regular iterations to address bug fixes, improvements, and new feature requests.
  - Keep the website aligned with evolving user needs and market trends.
- **Documentation and Knowledge Transfer (Throughout)**
  - Maintain documentation throughout the project, covering requirements, design decisions, and technical details.
  - Ensure knowledge transfer among team members for effective collaboration and support.

A Hybrid Model combining Waterfall and Agile elements offers NeonTech the advantage of structured planning and comprehensive initial requirements (Waterfall), while also providing flexibility and adaptability to evolving project needs (Agile). This approach allows for a balance between detailed planning and the ability to respond to changes, making it suitable for website development projects with varying degrees of complexity and uncertainty.

2.2 WHY HYBRID SDLC IS GOOD FOR NEONTECH

- **Tailored Approach:** NeonTech can benefit from a tailored approach that combines the strengths of both Waterfall and Agile methodologies to address their specific project requirements.
- **Initial Clarity (Waterfall Phase):** The Waterfall phase at the beginning of the project ensures that NeonTech gathers and documents detailed requirements comprehensively. This initial clarity is crucial for establishing a solid foundation and understanding the scope of the website.
- **Flexibility (Agile Phase):** Transitioning into the Agile phase allows NeonTech to adapt to changing requirements, which is common in web development projects. It enables them to respond quickly to client feedback and evolving market demands, resulting in a more customer-centric website.
- **Incremental Development:** The Agile phase's iterative and incremental approach lets NeonTech deliver functional parts of the website early, ensuring that they have a working product sooner. This is beneficial for showcasing progress to stakeholders and addressing critical features first.
- **Risk Mitigation:** By integrating testing throughout the development process, the Hybrid model reduces the risk of critical defects being discovered late in the project. This leads to better quality control.
- **User Involvement:** Agile iterations encourage client and user involvement throughout the development process, fostering a sense of ownership and collaboration. This helps NeonTech ensure that the website aligns closely with user expectations.
- **Structured Testing (UAT):** The transition back to a Waterfall-like phase for User Acceptance Testing (UAT) ensures a comprehensive evaluation of the website in a controlled environment. This minimizes the chances of major issues being discovered after deployment.
- **Balanced Documentation:** The model maintains documentation throughout the project, combining detailed documentation from the Waterfall phase with Agile's emphasis on collaboration and frequent communication.
- **Adaptive Maintenance:** After deployment, NeonTech can continue with Agile practices for ongoing maintenance and updates. This allows them to address user feedback, make improvements, and stay competitive in a rapidly changing digital landscape.
- **Resource Efficiency:** The Hybrid model optimizes resource allocation by focusing Waterfall's structured planning on the initial phase while using Agile's iterative development for the rest of the project.

Overall, the Hybrid Model offers NeonTech a balanced approach that harnesses the strengths of both Waterfall and Agile methodologies, making it suitable for website development projects that require a combination of structured planning and adaptability to changing circumstances. It can help NeonTech create a high-quality, customer-centric website that meets their business objectives while remaining responsive to evolving user needs and market trends.

3. CONTACT ME

You can get in touch with us in any of the below ways:

By Phone:
+8801985064618
Shrikanta Paul

By Email:
shrikantapaul571@gmail.com