### **1. Waterfall Model**

The first model utilized in the SDLC is the waterfall model. A different name for it is the linear sequential model.

In this paradigm, the output from one phase is used as the stimulus for the next. Only once the preceding phase has finished developing can the next phase begin.

The needs are obtained and examined first. The design phase can only start when the needs have been firmly established. The SRS Manual produced in this case serves as both an input and an output for the design process.

* Documents that serve as an input for the following phase—implementation and coding are prepared throughout the system design and software architecture phases.
* Programming is finished during the execution stage, and the resulting software is used as an input during the testing stage that comes next.
* All through the assessment process, the generated code is meticulously examined to look for any software defects. Up until the program is ready for usage, testing and debugging, additional testing, and defect recording are conducted.
* Following the customer's approval, the produced code is put into production during the deployment phase.
* The developers are responsible for fixing any problems that arise in the production environment and fall under maintenance.

#### **The Pros of the Waterfall Paradigm**

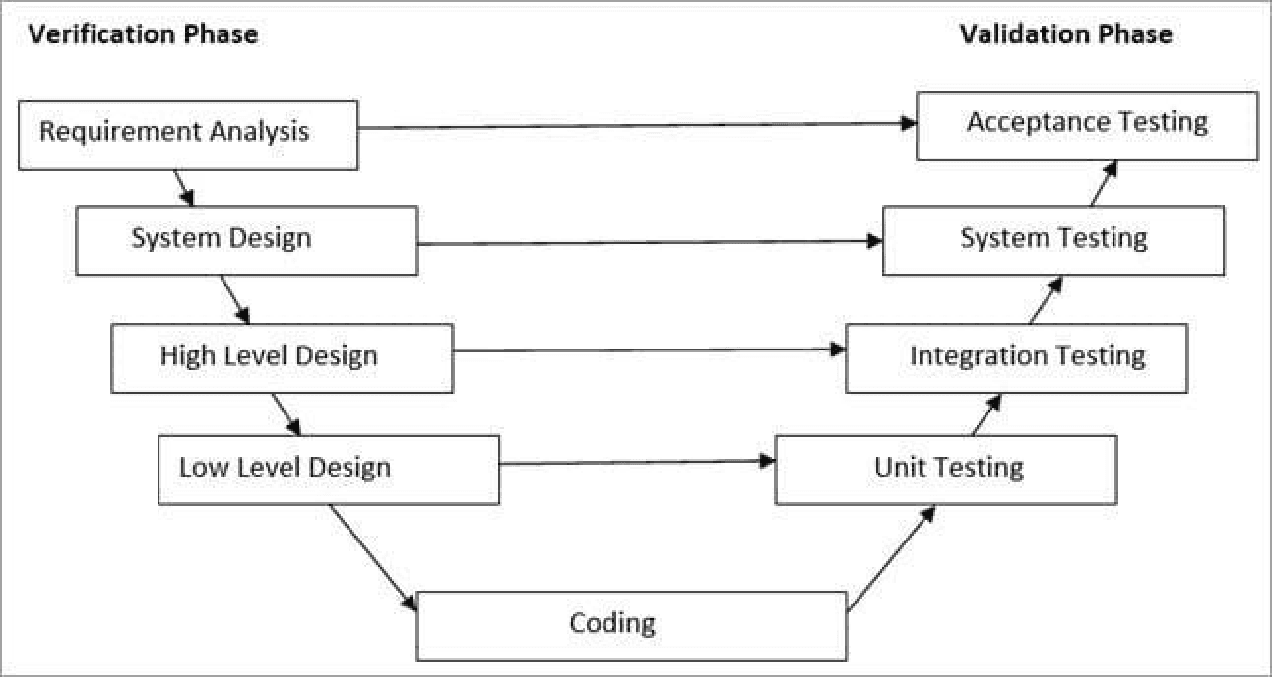
* The Waterfall Model is a straightforward, simply understandable model where each phase is completed one at a time.
* Each phase's deliverables are well specified, which prevents complexity and makes the project simple to manage.

#### **Shortfalls of the WaterFall Mode**

* A new phase cannot be begun in a waterfall model project until the current phase is finished, which makes it time-consuming and unsuitable for short-term projects.
* The waterfall model cannot be used for projects with ambiguous specifications or where the specifications are continually changing. This is because it relies on the needs being clear during the needs gathering and evaluation stages, and any changes made during the advanced stages would increase costs because they would need to be made during each phase.

### **2. V-Shaped Model**

Some people also call this model the "Validation" or "Verification" model. In this methodology, development and testing go concurrently, and validation and verification go together. The only difference between the V model and the waterfall model is that in the V-Model, testing and [test planning](https://www.knowledgehut.com/blog/software-testing/test-plan-in-software-testing) begin earlier.



Source: softwaretestinghelp.com

#### **The Pros of the V-Model**

* It is clear and concise.
* Smaller projects with stated requirements that freeze in the early stages benefit from the V-model approach.
* It is a methodical, disciplined approach that yields a high-quality end product.

#### **The cons of the V-Model**

* The V Model is rigid. It is challenging in the case of utilization. The execution and methods are pretty complex compared to other design environments or models.
* The V model is not very flexible. The architecture is straightforward, but it has limited flexibility in terms of design. The model design does not advocate highly complex calculations.
* The V Model software is designed during the implementation phase, so no initial prototypes are produced. Due to the absence of prototype guidance, it is complicated to develop professional software that requires a clear, concrete, and proper execution design.
* The drawback of the V Model is its update feature. The most highly professional software uses V Model for massive updates of both the test and requirement documents if something goes wrong midway.
* V Model is not suitable for building object-oriented software due to its uncertainties in the design.

### **3. Prototype Model**

In this model, the team first comes up with the prototype of the software that is to be developed.

* Comparing prototype models to the actual software reveals that they perform poorly and have limited functional capabilities. Prototypes are built with dummy functionality. This is a useful tool for figuring out what the customers want.
* Prototypes are developed before the finished version to get relevant user feedback. After taking into account user feedback, the prototype is once more examined by the client. Up until the consumer accepts the model, this process continues.
* After gathering requirements, a quick design is created, followed by the creation of a prototype, which is then sent to the client for evaluation.
* The prototype is modified in response to client comments and the clarified need, then re-presented to the client for review. The customer must accept the prototype before the genuine program can be developed. The Waterfall model technique is used to construct the real software.

#### **The Pros of a Prototype Model**

* Flaws are discovered considerably sooner using prototype models, and development costs and time are reduced.
* A change in need, a missing feature, or a lack of functionality can all be found during the assessment process and included in the finalized prototype.
* A customer's involvement from the beginning clears up any misunderstandings about the functionality's requirements.

#### **The Cons of the Prototype Model**

* The client can alter the requirements for the finished product since they are involved in every stage of the process, which makes the scope more difficult and could extend the time it takes to produce the product.

### **4. The Spiral Model**

A combination of both prototype and iterative approaches forms the spiral model.

The iterations adhere to the stages of the spiral model. The innermost loop in the model is for requirement collection and analysis, which is followed by planning, risk assessment, and production. The loops in the model are the different phases of the SDLC process. Design is the second loop, followed by system development and testing.

Planning, engineering evaluation, and risk analysis are the four phases of the spiral model.

* **A. Planning:** The requirement collection stage of the planning process involves getting all the necessary data from the client and documenting it. The writing of the System Requirement Specifications manual marks the start of the next step.
* **B. Risk Analysis:** A prototype is built to conduct the analysis and choose the best solution for the risks present.
* **C. Engineering:** Coding and testing are completed after the risk analysis.
* **D. Evaluation:** The customer assesses the system that was created and makes preparations for the next iteration.

#### **Advantages of the Spiral Model**

* The prototype models are extensively used for risk analysis.
* The following iteration can include any functionality changes or improvements.

#### **Disadvantages of the Spiral Model**

* Only huge projects are the greatest candidates for the spiral model. As it may require several iterations and take a long time to produce the desired result, the cost may be substantial.

### **5. Iterative Incremental Model**

The product is divided into manageable pieces using the iterative incremental methodology.

After each iteration is finished, the product is confirmed and sent to the client for review and feedback. The newly incorporated functionality is combined with customer reviews in the following iteration.

As a result, the product gains features as iterations go by, and the final version has all of the product's features.

Phases of the incremental and iterative model:

* **A. Inception Phase:** The project's requirements and scope are outlined in this phase.
* **B. Elaboration Phase:** During the elaboration phase, a product's workable architecture is supplied. This design addresses any risks found during the conception phase and satisfies any non-functional needs as well.
* **C. Construction Phase:** During the construction phase, the architecture is completed with code that is ready for deployment and is written through operational requirement analysis, layout, execution, and testing.
* **D. Transition Phase:** The product is introduced to the manufacturing environment during the Transition Phase.

#### **Pros of Iterative & Incremental Model**

Any requirement modification is possible and won't cost anything because the new need may be included in the following iteration.

* The iterations examine and identify risk.
* Early defect detection is achieved.
* The product is simple to manage because it has been broken up into tiny pieces.

#### **The Drawback of the Iterative Model**

* To break down and create gradually, a thorough grasp of the requirements and the product is necessary.

### **6. The Big Bang Model**

The Big Bang Model has no established procedure. Input and output consist of money and labor, and the result is a created good that may or may not be what the client requires.

The Big Bang Model doesn't need a lot of scheduling or planning. The developer analyzes the requirements, writes the code, and creates the product in accordance with his knowledge. This strategy is only applied to modest projects. The absence of a testing team and the absence of formal testing might lead to the project's failure.

#### **The Pros of the Big Bang Model**

* Its simplicity
* Planning and scheduling are not as necessary
* The developer is free to design their own software

#### **The Cons of the Big Bang Model**

* Large, ongoing, and sophisticated projects cannot be handled with big-bang models.
* high uncertainty and danger

#### **7. Agile Model**

The incremental and iterative models are combined to create the agile paradigm. This paradigm emphasizes flexibility throughout product development more so than requirements.

A product is divided into manageable incremental builds in Agile. It is not created in its entirety in one sitting. The number of features increases with each build.

The upcoming update expands upon earlier features.

* Sprints are the terms used for iterations in agile. A sprint lasts two to four weeks. Each sprint ends with the product owner verifying it before it is provided to the client with his or her permission.
* Customer input is used to make improvements, and the following sprint will focus on his recommendations and enhancements. Each sprint includes testing to reduce the possibility of any failures.

#### **The Pros of the Agile Model**

It gives one greater freedom to adjust to the changes.

* The new functionality is simple to implement.
* Customer satisfaction because we consider their comments and recommendations at every level.

#### **Disadvantages of Agile Model**

* inadequate documentation
* Agile requires highly qualified and experienced personnel.
* The project would fail if the customer was unsure of exactly what they wanted the final result to be.

## **Types of Testing in SDLC (Software Development Lifecycle)**

Understanding the role of testing in SDLC is pivotal for ensuring the reliability of software throughout its development. In this context, let's explore the different types of testing in SDLC and their significance in delivering high-quality software products.

### **1. Unit Testing**

The smallest testable components of an application, known as units, are separately and separately examined for appropriate operation as part of the unit testing phase of software development. Software engineers and occasionally QA employees use this [testing approach](https://www.knowledgehut.com/blog/devops/testing-in-devops) when the software is still in the development stage. Unit testing's primary goal is to separate written code for testing to see if it functions as intended.

Unit testing is a crucial stage in the development process because, when done properly, it may aid in finding early code issues that could be more challenging to identify in subsequent testing phases. and successfully!

### **2. Integration Testing**

Integration testing is the process of testing several software development components collectively. Whether they work together effortlessly as a unit or as a system depends on this.

Some people misunderstand it with unit testing, which likewise evaluates components of the whole, but there is a distinction in when and how the test is run.

Modules or parts of the project are tested independently to ensure they operate as intended. Groups of these modules are subjected to integration testing to ensure optimal interoperability and cooperation. While integration testing is typically done later in the project schedule, unit testing is one of the first tests carried out in a project.

### **3. System Testing**

The method through which a quality assurance (QA) team assesses how the different aspects of a software relate with one another in the entire, integrated system or application is known as system testing, also known as system-level testing or systems integration testing.

System testing makes sure the software works as intended. This procedure, which functions as a form of black box testing, focuses on an application's performance. For instance, system testing may examine whether each type of user input results in the desired output across the application.

UAT refers to the last phase of the [software testing process](https://www.knowledgehut.com/blog/devops/testing-in-devops). Before releasing the real product, this UAT procedure serves a significant and important function in validating whether all business criteria have been met.

Additionally, the definition of UAT testing may be described as the user technique, in which the generated software is tested by the business user to see whether it is operating in accordance with the established requirements. Beta testing, [application testing](https://www.knowledgehut.com/blog/devops/testing-in-devops), and end-user testing are other names for this kind of testing. Before deciding to release the program onto the market, the UAT testing procedure is undertaken.

### **4. User Acceptance Testing**

UAT refers to the last phase of the software testing process. Before releasing the real product, this UAT procedure serves a significant and important function in validating whether all business criteria have been met.

Additionally, the definition of UAT testing may be described as the user technique, in which the generated software is tested by the business user to see whether it is operating in accordance with the established requirements. Beta testing, application testing, and end-user testing are other names for this kind of testing. Before deciding to release the program onto the market, the UAT testing procedure is undertaken.

## **Benefits of the SDLC**

Using a safe software development life cycle has a lot of benefits. Here are a few of the most important advantages one should be aware of.

* Early vulnerability detection lowers the cost of establishing security measures and vulnerability mitigation procedures. Instead of releasing patched software, which is far more expensive than fixing the issue in real-time during the SDLC, the security flaws are corrected during the development cycle.
* A safe SDLC also has the benefit of fostering a security culture that is more likely to identify problems in other departments within an organization in addition to production.
* Essential security choices are recorded before development starts because security is included into the design phase of a safe SDLC. The development and management team are both aware of the project's security dangers and worries. As a result, the development approach may be adjusted to guarantee secure code is created as the SDLC moves forward.
* One of the main benefits of a secure SDLC is that it aids in the company's overall reduction of inherent business risks. Businesses that suffer from cybersecurity assaults typically lose a lot more money than they had planned, whether it be from minor security threats like SQL or XML intrusions or serious security problems like DoS (denial of service).