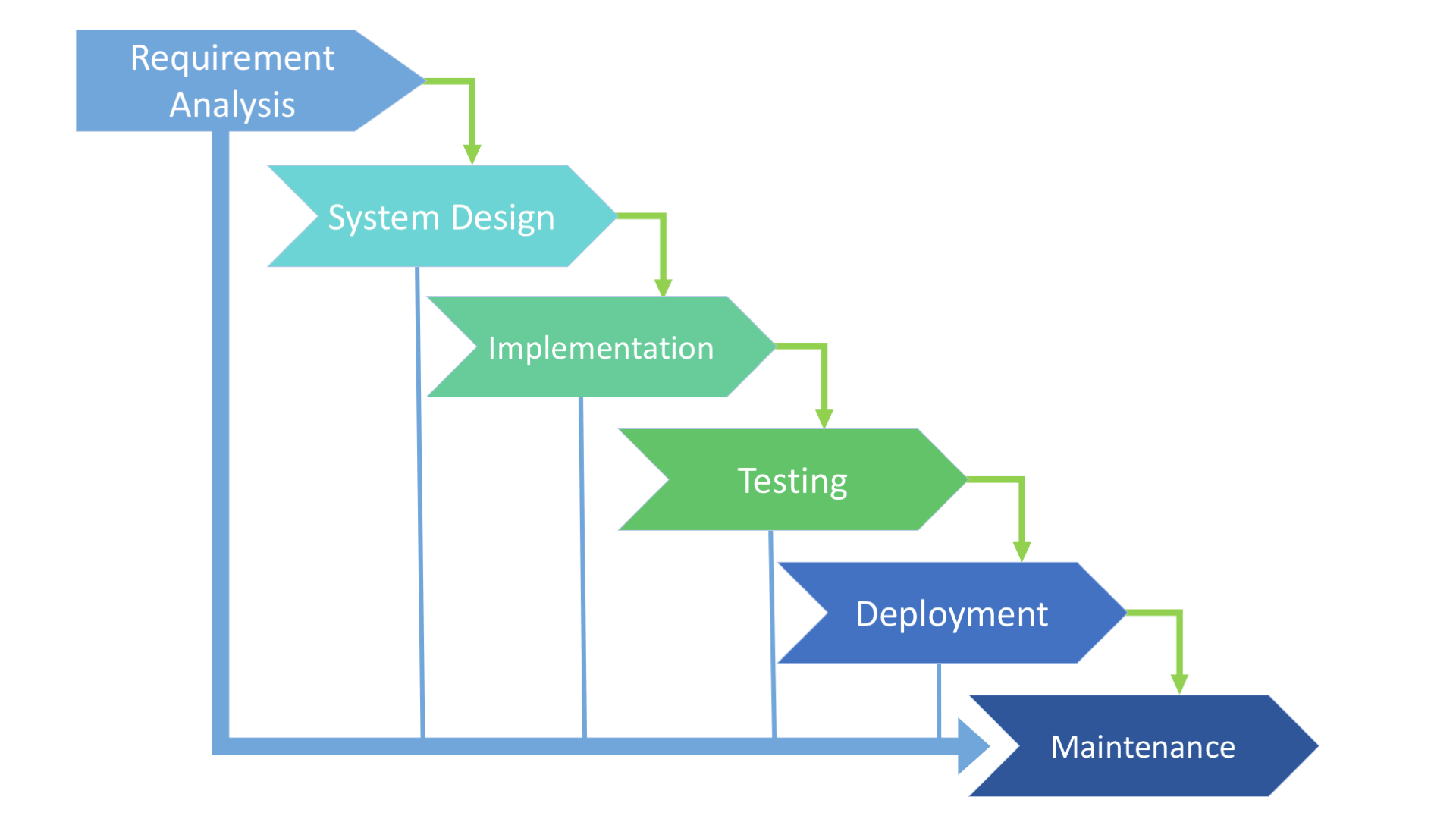
**1.What is SDLC Waterfall Model?**

**Waterfall model is an example of a Sequential model. In this model, the software development activity is divided into different phases and each phase consists of a series of tasks and has different objectives. which is also known as the linear-sequential life cycle model**

Waterfall model is the pioneer of the [SDLC processes](https://en.wikipedia.org/wiki/Software_development_process). In fact, it was the first model which was widely used in the software industry. It is divided into phases and output of one phase becomes the input of the next phase. It is mandatory for a phase to be completed before the next phase starts. In short, there is no overlapping in the Waterfall model.

In waterfall, development of one phase starts only when the previous phase is complete. Because of this nature, each phase of the waterfall model is quite precise well defined. Since the phases fall from a higher level to lower level, like a waterfall, It’s named as the waterfall model.

**Waterfall** methodologies result in a project schedule with 20–40% of the **time** invested for the first two phases, 30–40% of the **time** to coding, and the rest dedicated to testing and implementation. The actual project organisation needs to be highly structured.



## **When to use SDLC Waterfall Model?**

* equirements are not changing frequently
* Application is not complicated and big
* Project is short
* Requirement is clear
* Environment is stable
* Technology and tools used are not dynamic and is stable
* Resources are available and trained

**Advantages of using the Waterfall model:**

* Simple and easy to understand and use.
* For smaller projects, the waterfall model works well and yield the appropriate results.
* Since the phases are rigid and precise, one phase is done one at a time, it is easy to maintain.
* Suited for smaller projects where requirements are well defined.
* Before the next phase of development, each phase must be completed.
* They should perform quality assurance test (Verification and Validation) before completing each stage.
* Elaborate documentation is done at every phase of the software's development cycle.
* Project is completely dependent on project team with minimum client intervention.
* Any changes in software is made during the process of the development
* Results are well documented.

**Disadvantages of using Waterfall model:**

* Error can be fixed only during the phase
* It becomes very difficult to move back to the phase. For example, if the application has now moved to the testing stage and there is a change in requirement, It becomes difficult to go back and change it.
* For bigger and complex projects, this model is not good as a risk factor is higher.
* Not suitable for the projects where requirements are changed frequently.
* Delivery of the final product is late as there is no prototype which is demonstrated intermediately.
* Testing period comes quite late in the developmental process
* Documentation occupies a lot of time of developers and testers
* Clients valuable feedback cannot be included with ongoing development phase
* Small changes or errors that arise in the completed software may cause a lot of problems

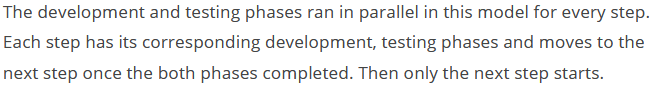
**Conclusion**

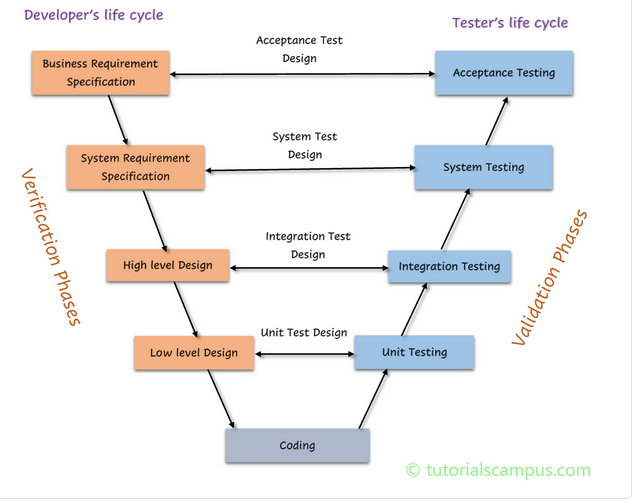
### In the waterfall model, it is very important to take the [sign off](https://www.softwaretestinghelp.com/bug-tracking-test-metrics-and-test-sign-off-free-qa-training-day-6/) of the deliverables of each phase. As of today most of the projects are moving with [Agile](https://www.softwaretestinghelp.com/category/agile-testing/) and Prototype models, Waterfall model still holds good for smaller projects. If requirements are straightforward and testable, Waterfall model will yield the best results.

### **2.What is SDLC V-Shaped Model?**

The V model is often said to be an extension of the waterfall model.

[**V- Model**](https://www.softwaretestinghelp.com/what-is-stlc-v-model/) **is also known as Verification and Validation Model. In this model Verification & Validation goes hand in hand development and testing goes parallel. V model and waterfall model are the same except that the test planning and testing start at an early stage in V-Model.**

V model is now one of the most widely used software development processes.



1. **Verification:**

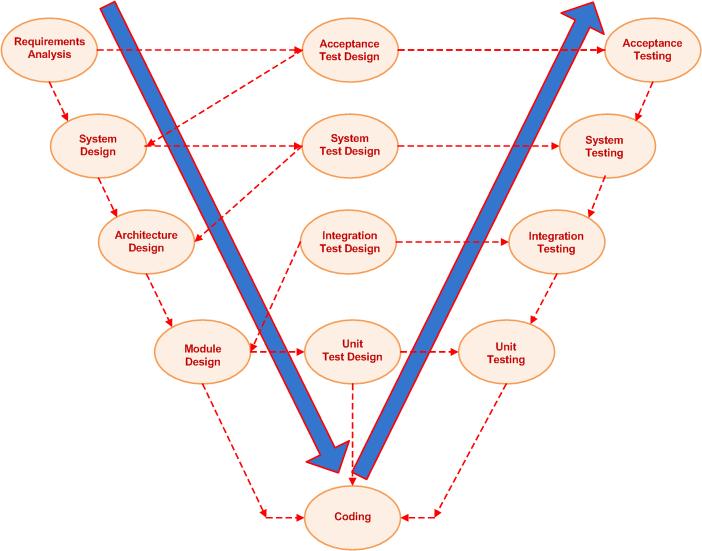
In the concept of verification in the V-Model, static analysis technique is carried out without executing the code. This evaluation procedure is carried out at the time of development to check whether specific requirements will meet or not. Examples include – Reviews, Inspection, and walkthrough.

1. **Validation:**

This concept of V-Model comprises of dynamic analysis practice (both functional as well as non-functional), and testing is done by code execution. The validation of a product is done once the development is complete for determining if the software meets up the customer hope needs. Examples include functional and non-functional testing techniques.

So both verification and validation are combined and work in parallel to make the V-Model fully functional.

In a typical development process, the left-hand side shows the development activities and the right hand side shows the testing activities. I should not be wrong if I say that in the development phase both verification and validation are performed along with the actual development activities.



**Verification Phase:**

**Requirement analysis**:

In this phase, the requirements are collected, analyzed and studied. Here how the system is implemented, is not important but, what the system is supposed to do, is important. Brain storming sessions/walkthrough, interviews are done to have the objectives clear.

* *Verification activities*: Requirements reviews.
* *Validation activities*: Creation of UAT ([User acceptance test](https://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/)) test cases
* *Artifacts produced*: Requirements understanding document, UAT test cases.

Requirement review is where detailed communication including exact functions and requirements is held, as well as acceptance test design planning.

**System requirements /** **High-level Design**: **/** **System Design:**

In this phase, the high-level design of the software is built. The team studies and investigates on how the requirements could be implemented. The technical feasibility of the requirements is also studied. The team also comes up with the modules that would be created/ dependencies, Hardware/software needs

* *Verification activities*: Design reviews
* *Validation activities*: Creation of [System test plan](https://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/) and cases, Creation of traceability metrics
* *Artifacts produced*: System test cases, Feasibility reports, System test plan, Hardware-software requirements, and modules to be created, etc.

 System design stage involves understanding and detailing out the entire hardware and communication setup for the product. System test design is best to be planned at this stage.

In this phase of SDLC, the system is designed with the entire hardware & the setup is constructed/ communication for product development.

It is also known as functional design, the aim of which is to prepare functional design of the software. In case of any functionality, that is not feasible the same is intimated to the user.

**Architectural design:**

The breakdown of system design to a more detailed version,into modules which creates different functionalities. Transferring of data and connection between internal and external modules (i.e., the outside world) is evidently identified.

In this phase, based on the high-level design**,** software architecture is created. The modules, their relationships, and dependencies, architectural diagrams, database tables, technology details are all finalized in this phase.

* *Verification activities*: Design reviews
* *Validation activities*: Integration test plan and test cases.
* *Artifacts produced*: Design documents, Integration test plan and test cases, Database table designs etc.

Architectural design stage is where high-level design is developed. Along with understanding the technical and financial feasibility of the product before it is actually developed. Integration test is best to be planned at this stage.

**Module design/Low-level Design:**

In this phase, each and every module of the software components are designed individually. Methods, classes, interfaces, data types etc are all finalized in this phase.

* *Verification activities*: Design reviews
* *Validation activities*: Creation and review of unit test cases.
* *Artifacts produced*: Unit test cases,

Module design stage focuses on designing a detailed plan for the internal modules of the system, also known as low-level design, it is important to ensure that the design is compatible with other surrounding modules in system architecture.

This particular phase breaks down the entire product development into tiny modules where each intended module is specified. So it is also termed as Low-Level Design (LLD).

**Implementation / Code**: I

In this phase, the actual coding is done.

* *Verification activities*: Code review, test cases review
* *Validation activities*: Creation of functional test cases.
* *Artifacts produced*: test cases, review checklist.

**Validation Phase:**

**Unit Testing:**

During the development of module design, unit testing is carried out. This plan is executed for eliminating bugs that are found in code at the development of your software.

[Unit testing](https://www.softwaretestinghelp.com/unit-testing/) is performed using the unit test cases that are designed and is done in the Low-level design phase. Unit testing is performed by the developer itself. It is performed on individual components which lead to early defect detection.

**Integration Testing:**

Integration testing is done to check whether there is a valid and proper communication within the internal modules of the system .This testing is done in the architecture design phase.

[Integration testing](https://www.softwaretestinghelp.com/what-is-integration-testing/) is performed using integration test cases in High-level Design phase. Integration testing is the testing that is done on integrated modules. It is performed by testers.

**System Testing:**

[System testing](https://www.softwaretestinghelp.com/system-testing/) is performed in the System Design phase. In this phase, the complete system is tested the entire system functionality is tested.

System testing is the testing of the entire system which is to ensure if the internal modules are configured to communicate effectively with the external systems.

**Acceptance Testing:**

This type of testing is carried out in front of the user or in a user environment where the product will ultimately set up. The UAT particularly test whether the product is capable enough to launch in the market or ready to work in the real world.

Acceptance testing is associated with the Requirement Analysis phase and is done in the customer’s environment

## **When to use the V Model?**

* The requirement is well defined and not ambiguous
* Acceptance criteria are well defined.
* Project is short to medium in size.
* Technology and tools used are not dynamic.
* The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

**Advantages:**

* This is a highly disciplined model and Phases are completed one at a time.
* V-Model is used for small projects where project requirements are clear.
* Simple and easy to understand and use.
* This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
* It enables project management to track progress accurately.

**Disadvantages:**

* High risk and uncertainty.
* It is not a good for complex and object-oriented projects.
* It is not suitable for projects where requirements are not clear and contains high risk of changing.
* This model does not support iteration of phases.
* It does not easily handle concurrent events.

**3.What is SDLC Prototype Model?**

**The Prototype model is one of the software development life cycle models in which a prototype is built with minimal requirements. This prototype is then tested and modified based on the feedback received from the client until a final prototype with desired functionalities gets created. This final prototype also acts as a base for the final product.**

As mentioned earlier, this model is useful when all the detailed requirements are not known to the client before starting the project. It is also useful when the product to be developed is a complex one and similar product does not exist in the market.

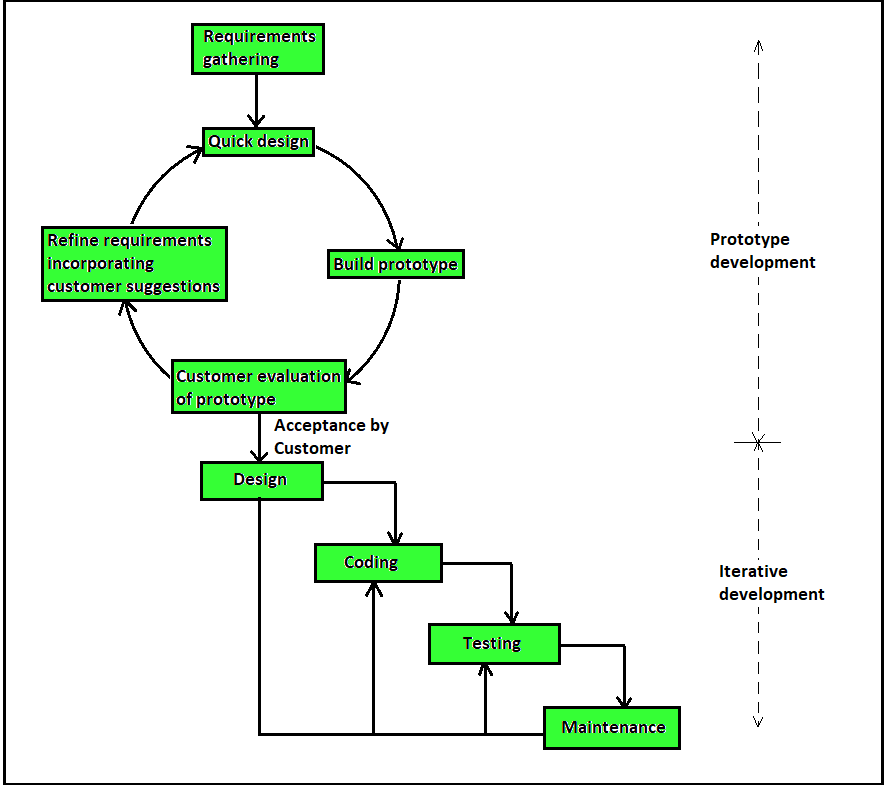
In such a scenario, the client can ask the developers to start working on the basic prototype with limited requirements. Once the basic prototype is ready, the client can see and check the prototype to decide what all changes are required.   
  
The client can also use the prototype to do market research and gather end-user or customer feedback.   
  
When the client has decided about the changes that need to be made, the client will give these requirements to the requirements gathering team. These changes then eventually reach the development team.   
  
Developers can then start working on the modifications to the basic prototype. This cycle will be repeated until the client is satisfied with the prototype which reflects the final product.

We all know prototype is a preliminary version of the complete software. It’s not compulsory that the prototype of the software constitutes all the features of the final product i.e. final software. Still, the prototype reflects the features of the customer’s interest.

Prototyping is a software development model where after each iteration the prototype is presented to the customer for evaluation. Customer evaluates the prototype and confirms whether the prototype is developed according to their demand or it needs some modification.

Developer considers the feedback given by the customer and modifies the prototype according to their demand. After modification, the prototype is demonstrated to the customer for evaluation. The customer again evaluates the prototype and provide the feedback on which developer again modifies the prototype.

These iterations continue until the customer is satisfied with the prototype. At each iteration, the prototype undergoes different phases which we will discuss in the further section. Once the customer is satisfied with the prototype developers start developing the software and then deliver it to the customer.



### **Prototyping Model has following six SDLC phases as follow:**

### **Step 1: Requirements gathering and analysis**

A prototyping model starts with requirement analysis. In this phase, the requirements of the system are defined in detail. During the process, the users of the system are interviewed to know what is their expectation from the system.

### **Step 2: Quick design**

The second phase is a preliminary design or a quick design. In this stage, a simple design of the system is created. However, it is not a complete design. It gives a brief idea of the system to the user. The quick design helps in developing the prototype.

### **Step 3: Build a Prototype**

In this phase, an actual prototype is designed based on the information gathered from quick design. It is a small working model of the required system.

### **Step 4: Initial user evaluation**

In this stage, the proposed system is presented to the client for an initial evaluation. It helps to find out the strength and weakness of the working model. Comment and suggestion are collected from the customer and provided to the developer.

### **Step 5: Refining prototype**

If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback and suggestions.

This phase will not over until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed based on the approved final prototype.

### **Step 6: Implement Product and Maintain**

Once the final system is developed based on the final prototype, it is thoroughly tested and deployed to production. The system undergoes routine maintenance for minimizing downtime and prevent large-scale failures.

Types of Prototyping Models

Four types of Prototyping models are:

1. Rapid Throwaway prototypes
2. Evolutionary prototype
3. Incremental prototype
4. Extreme prototype

### **Rapid Throwaway Prototype**

Rapid throwaway is based on the preliminary requirement. It is quickly developed to show how the requirement will look visually. The customer's feedback helps drives changes to the requirement, and the prototype is again created until the requirement is baselined.

In this method, a developed prototype will be discarded and will not be a part of the ultimately accepted prototype. This technique is useful for exploring ideas and getting instant feedback for customer requirements. (Hand+Pen+Pencil)

### **Evolutionary Prototyping**

Here, the prototype developed is incrementally refined based on customer's feedback until it is finally accepted. It helps you to save time as well as effort. That's because developing a prototype from scratch for every interaction of the process can sometimes be very frustrating.

This model is helpful for a project which uses a new technology that is not well understood. It is also used for a complex project where every functionality must be checked once. It is helpful when the requirement is not stable or not understood clearly at the initial stage.

### **Incremental Prototyping**

In incremental Prototyping, the final product is decimated into different small prototypes and developed individually. Eventually, the different prototypes are merged into a single product. This method is helpful to reduce the feedback time between the user and the application development team.

### **Extreme Prototyping:**

Extreme prototyping method is mostly used for web development. It is consists of three sequential phases.

1. Basic prototype with all the existing page is present in the HTML format.
2. You can simulate data process using a prototype services layer.
3. The services are implemented and integrated into the final prototype.

This method is mainly used for web development. It is consists of three sequential independent phases:

**D.1)** In this phase a basic prototype with all the existing static pages are presented in the HTML format.

**D.2)**  In the 2nd phase, Functional screens are made with a simulate data process using a prototype services layer.

**D.3)** This is the final step where all the services are implemented and associated with the final prototype.

his Extreme Prototyping method makes the project cycling and delivery robust and fast, and keeps the entire developer team focus centralized on products deliveries rather than discovering all possible needs and specifications and adding unnecessitated features.

**When to use Prototype model:**

* The Prototyping Model should be used when the requirements of the product are not clearly understood or are unstable.
* It can also be used if requirements are changing quickly.
* This model can be successfully used for developing user interfaces, high technology software-intensive systems, and systems with complex algorithms and interfaces. It is also a very good choice to demonstrate the technical feasibility of the product.
* Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
* Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
* Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.
* Whenever the customer not clears about the requirement in this situation we generally go for prototype model.
* If it is complex project then prototype model makes clear understand the requirement.
* Prototyping make sure that the customer constantly work with the system and provide a feedback about the system.

## **Advantages of the Prototyping Model**

* Users are actively involved in development. Therefore, errors can be detected in the initial stage of the software development process.
* Missing functionality can be identified, which helps to reduce the risk of failure as Prototyping is also considered as a risk reduction activity.
* Helps team member to communicate effectively
* Customer satisfaction exists because the customer can feel the product at a very early stage.
* There will be hardly/less any chance of software rejection.
* Quicker user feedback helps you to achieve better software development solutions.
* Allows the client to compare if the software code matches the software specification.
* It helps you to find out the missing functionality in the system.
* It also identifies the complex or difficult functions.
* Encourages innovation and flexible designing.
* It is a straightforward model, so it is easy to understand.
* No need for specialized experts to build the model
* The prototype serves as a basis for deriving a system specification.
* The prototype helps to gain a better understanding of the customer's needs.
* Prototypes can be changed and even discarded.
* A prototype also serves as the basis for operational specifications.
* Prototypes may offer early training for future users of the software system.

## **Disadvantages of the Prototyping Model**

* Costly w.r.t time as well as money
* Prototyping is a slow and time taking process.
* The cost of developing a prototype is a total waste as the prototype is ultimately thrown away.
* Prototyping may encourage excessive change requests.
* Some times customers may not be willing to participate in the iteration cycle for the longer time duration.
* There may be far too many variations in software requirements when each time the prototype is evaluated by the customer.
* Poor documentation because the requirements of the customers are changing.
* It is very difficult for software developers to accommodate all the changes demanded by the clients.
* After seeing an early prototype model, the customers may think that the actual product will be delivered to him soon.
* The client may lose interest in the final product when he or she is not happy with the initial prototype.
* Developers who want to build prototypes quickly may end up building sub-standard development solutions.

## **Conclusion**

The prototype model is a trial and error method which has its advantages and disadvantages. It is particularly useful when the client does not have clarity on what all features, they need in the product.

**4.What is SDLC Incremental**  **Model?**

**In the Incremental model, the processes involved in software development are divided into multiple individual modules in the Software development life cycle (SDLC). This model can be otherwise called a Successive version model.**

Once the modules are divided, the process of incremental development will begin in stages. These stages will cover all the analysis, implementation, designing, testing/verification, and even maintenance. The incremental model can be related to the Waterfall model in the sense that it combines the elements of the waterfall model but iteratively. The waterfall model is applied repeatedly in each increment. The incremental model also applies linear sequences in a required pattern.

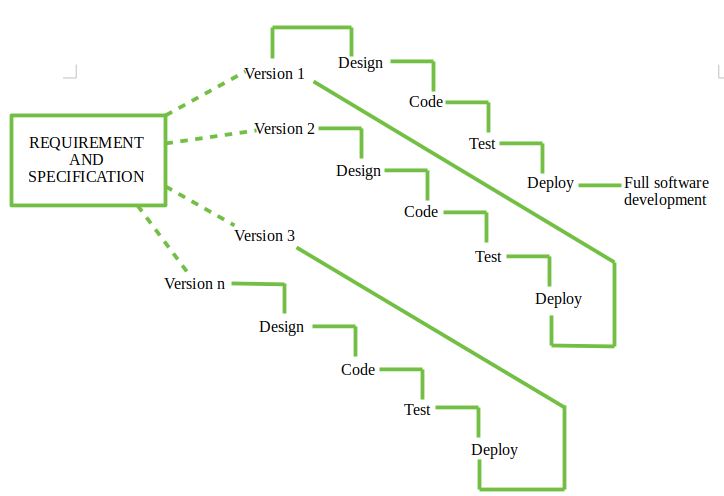
Each of the iterations in the Incremental model passes through four different phases. They are the requirements phase, design phase, coding phase, and testing phase. When a new system is released, it improves the function of the previous release. This is done until all the design functionality has been actualized.

## **Life Cycle Activities Involved in Incremental Model**

In the incremental model, the software requirements are first broken down into different modules that can be constructed incrementally and delivered. There are no long-term plans at any time during this process as the plan is just for the next increment. This makes it easier to modify the version depending on the customer’s needs.

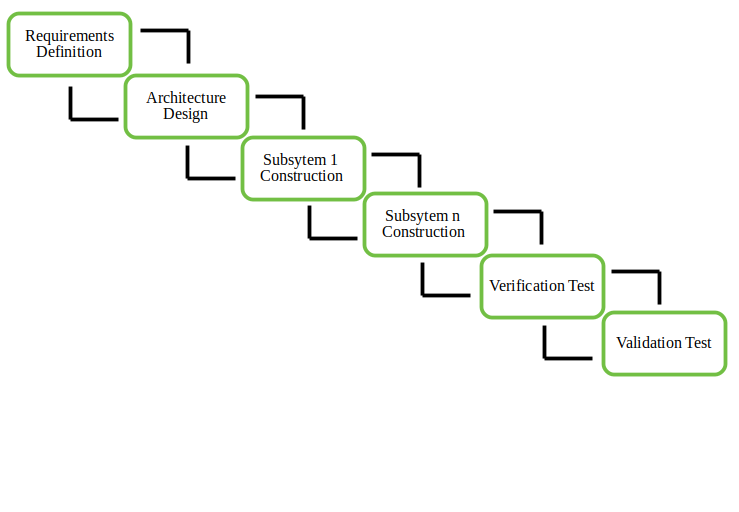
Firstly, the development team develops the core features of the system. After successfully developing the core features, they are then refined by adding new functionalities in successive versions which will increase the level of capabilities. The incremental versions are developed with an [iterative waterfall model](https://en.wikipedia.org/wiki/Waterfall_model). As the process continues and each successive version is constructed and delivered, the feedback of the customer is then taken which is added in the next version. This means that each version of the software possesses more features than the previous ones.

After the requirements have been gathered and specified, they are then divided into different versions beginning with version-1. After each successful increment, the next version is constructed and then at the customer site. When the process ends, the last version (version n) is now deployed at the customer client.

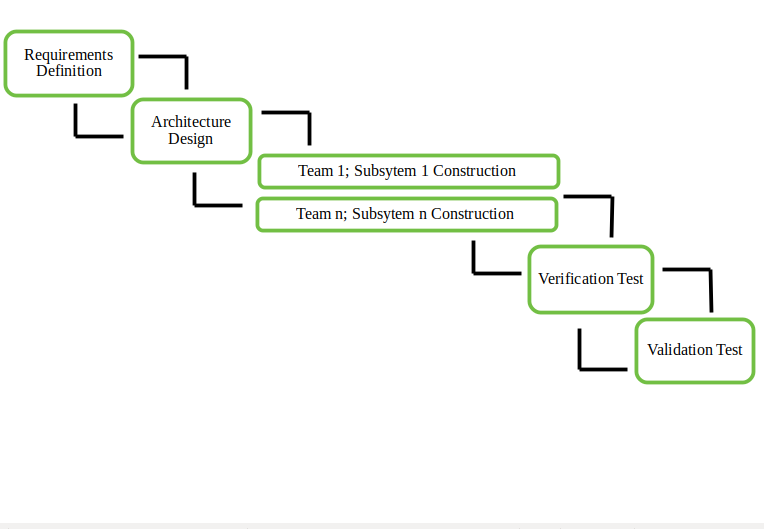


**Types of Incremental model –**

1. **Staged Delivery Model –** Construction of only one part of the project at a time.



1. **Parallel Development Model –** Different subsystems are developed at the same time or simultaneously. It can decrease the calendar time needed for the development, i.e. TTM (Time to Market), if enough Resources are available.



## **Characteristics of Incremental Model**

* At any point in time, there are no long-term plans involved. This means that the plan is done for only the existing increment and complete focus is on the requirement worked on at the moment.
* Priority is given to each of the requirements/stages and the requirement of the highest priority is first tackled.
* The software that is to be developed will be divided into different stages. This leads to many mini sub-projects present in the software.
* The phases used in developing each increments version follow the analysis, design, code, and test phase. The iterative waterfall model is also frequently used to develop the incremental version, even though other models can be used as well.
* The partial systems that are to be developed are combined to produce the complete objective of the software.
* After handling a requirement of the increment, it is then frozen so that focus can shift to the next requirement or increment.

**When to use this –**

* Funding Schedule, Risk, Program Complexity, or need for early realization of benefits.
* When Requirements are known up-front.// Requirements of the system are clearly understood
* When Projects having lengthy developments schedules.
* When a project possesses a new technology.
* When demand for an early release of a product arises
* When a project has a lengthy development schedule.
* When software engineering team are not very well skilled or trained
* When high-risk features and goals are involved
* Such methodology is more in use for web application and product based companies
* When there is a need to have the basic functionality delivered as quickly as possible.
* When the domain of the project is new to the team.
* When the requirements needed for the project are prioritized.
* When most of the requirements needed are known before the start of the project, but they are expected to change as the project progresses.

**Advantages –**

* Error Reduction (core modules are used by the customer from the beginning of the phase and then these are tested thoroughly)
* Uses divide and conquer for breakdown of tasks.
* Lowers initial delivery cost.// The cost of initial delivery is reduced.
* Incremental Resource Deployment.
* Since the project is divided into several incremental stages, 100% requirements will be achieved as well as 100% objectives of the software met.
* Feedback can be provided by the customer at each stage of the model. This will prevent sudden changes in the requirement and also value work efforts.
* Since each incremental phase is tested, there will be numerous testing for the software which will, in turn, lead to fewer defects and better results.
* The incremental model tends to be cheaper on the side of the consumer when compared to other software development models.
* It is easy to identify errors when using this model.
* The model is flexible. The cost is low even when there is a change in the scope of the requirement.
* The software is quickly generated during its life cycle.
* Risk management is easy because the risky parts are identified and properly handled during iteration.
* A new feature is added to the product after a new release.

**Disadvantages –**

* Requires good planning and design.
* Total cost is not lower.
* Well defined module interfaces are required.
* Rectifying a problem in one unit requires correction in all the units and consumes a lot of time
* The total cost of the complete system is high.
* The model requires an efficient design to ensure the inclusion of the required functionality as well as providing for changes later in the project.
* The interfaces used for software development must be well-defined, as some of the interfaces are developed long before others begin development.
* The model requires proper planning of iterations.
* The model also requires the early definition of a fully functional and complete system to allow the definition of increments.

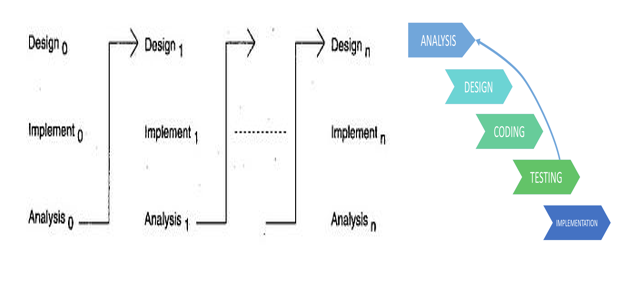
## **Conclusion**

Many models can develop software and meet the required objective, but we have seen that incremental models meet the requirements with 100% objectives of the software expected.

# 5.What is SDLC Iterative model

**An iterative** [**life cycle model**](http://tryqa.com/what-are-the-software-development-models/) **does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.**

The Iterative SDLC model does not need the full list of requirements before the project starts. The development process may start with the requirements to the functional part, which can be expanded later. The process is repetitive, allowing to make new versions of the product for every cycle. Every iteration (which last from two to six weeks) includes the development of a separate component of the system, and after that, this component is added to the functional developed earlier. Speaking with math terminology, the iterative model is a realization of the sequential approximation method; that means a gradual closeness to the planned final product shape.



**When to use iterative model:**

* The requirements to the final product are strictly predefined or Primary necessities of the system can be defined; at the same time, some system's working can be improved with the development process.
* Applied to the large-scale projects
* Major requirements must be defined; however, some details can evolve with time.
* If a new technology needs prior understanding, this model can be helpful to know the latest technology and increment or update the model accordingly.
* This model is also useful when there are high risks in the system characteristic and goals.
* Situations where resources with required skill sets are not accessible, and the system needs to be developed on a contract basis, choosing this model is a suitable decision.

## **Advantages of Iterative Model:**

* Some working functionality can be developed and early in the software development life cycle (SDLC).
* It is easily adaptable to the ever changing needs of the project as well as the client.
* It is best suited for agile organisations.
* It is more cost effective to change the scope or requirements in Iterative model.
* Parallel development can be planned.
* Testing and debugging during smaller iteration is easy.
* Risks are identified and resolved during iteration; and each iteration is an easily managed.
* In iterative model less time is spent on documenting and more time is given for designing.
* One can get reliable user feedback, when presenting sketches and blueprints of the product to users for their feedback.

## **Disadvantages of Iterative Model:**

* More resources may be required.
* Although cost of change is lesser, but it is not very suitable for changing requirements.
* More management attention is required.
* It is not suitable for smaller projects.
* Highly skilled resources are required for skill analysis.
* Project progress is highly dependent upon the risk analysis phase.
* Defining increments may require definition of the complete system.

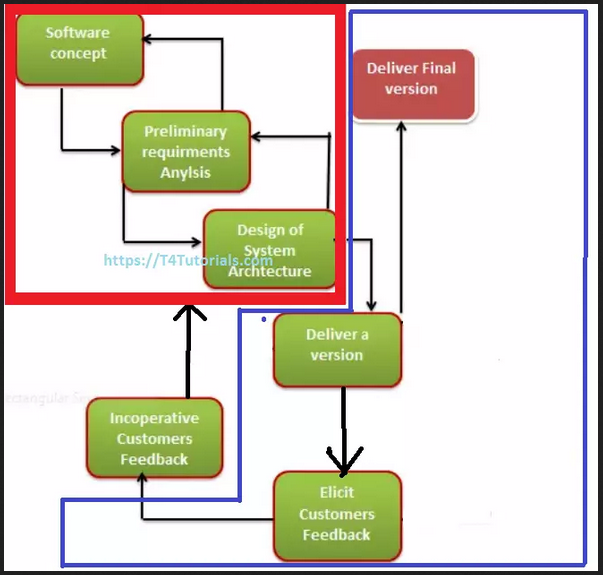
**Conclusion:**

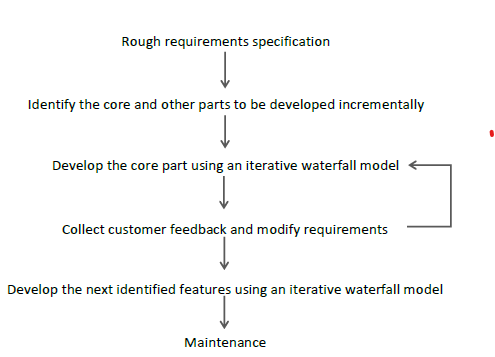
## Iterative model iterates Requirements, Design, Build and test phases again and again for each requirement and builds up a system iteratively till it is completely built. Moreover, iterative model can accommodate changes in requirements, which are very common in most of the projects. It also provides an opportunity to identify and build any major requirement or design flaws throughout the process because of its iterative nature.

## **6.What is SDLC Evolutionary Model?**

**Evolutionary model is a combination of** [**Iterative**](https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/) **and** [**Incremental model**](https://www.geeksforgeeks.org/software-engineering-incremental-process-model/) **of software development life cycle. Evolutionary development is based on the idea of developing an initial implementation, exposing this to user comment and refining it through many versions until an adequate system has been developed Specification, development and validation activities are interleaved with rapid feedback across activities**.

The evolution model divides the development cycle into smaller, "Incremental Waterfall Model" in which users are able to get access to the product at the end of each cycle.





**There are two fundamental types of evolutionary development:**

1. **Exploratory development.** The objective of the process is to work with the customer in order to explore their requirements and deliver a final system. The development starts with the parts of the system that are well understood. The system evolves by adding new features proposed by the customer.
2. **Throwaway prototyping.** In this case the objective of the evolutionary development process is to understand the customer’s unclear requirements, namely to validate and derive the requirements definition for the system. The prototype concentrates on experimenting with the customer requirements that are poorly understood.

**When Use Evolutionary Model:**

* It is used in large projects where you can easily find modules for incremental implementation. Evolutionary model is commonly used when the customer wants to start using the core features instead of waiting for the full software.
* Evolutionary model is also used in object oriented software development because the system can be easily portioned into units in terms of objects.
* They are iterative in Process.
* They enable the software developer to develop increasingly more Complex versions of the software.
* Like all Complex systems, software involve over period of the time and hence evolutionary models are more suited to software development.
* Requirements gets changed while the software is under development.

## **Advantages of Evolutionary Model**

* Error reduction: As the version is tested with customer which reduces the error throughlly.
* User satisfaction: User gets satisfied and he gets the full chance of experimenting partially developed system.
* Business benefit: Successful use of this model can benefit not only business result but marketing and the internal operations as well.
* High quality: As you should get satisfied with every version, it produces the high quality product.
* Low risk: There is significant reduction of risk as a versions is implemented. **This risk may be associated with**   
  - mising schedule deadline  
  - wrong feature sets   
  - poor quality
* Reduction Cost: Some design issues are cheaper to resolve through experimentation than through analysis. It reduces cost by providing structured and disciplined avenue for experimentation.

## **Disadvantages of Evolutionary Model**

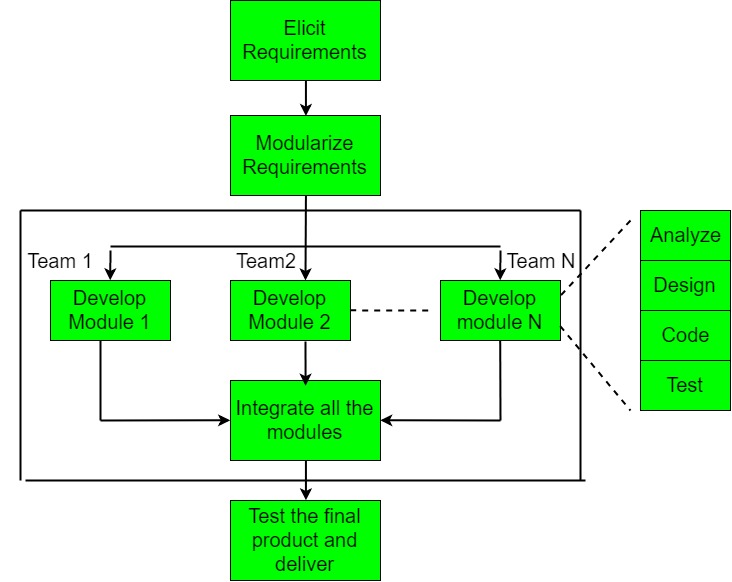
## Several version release: Developer has to make table version which increases their Efforts.

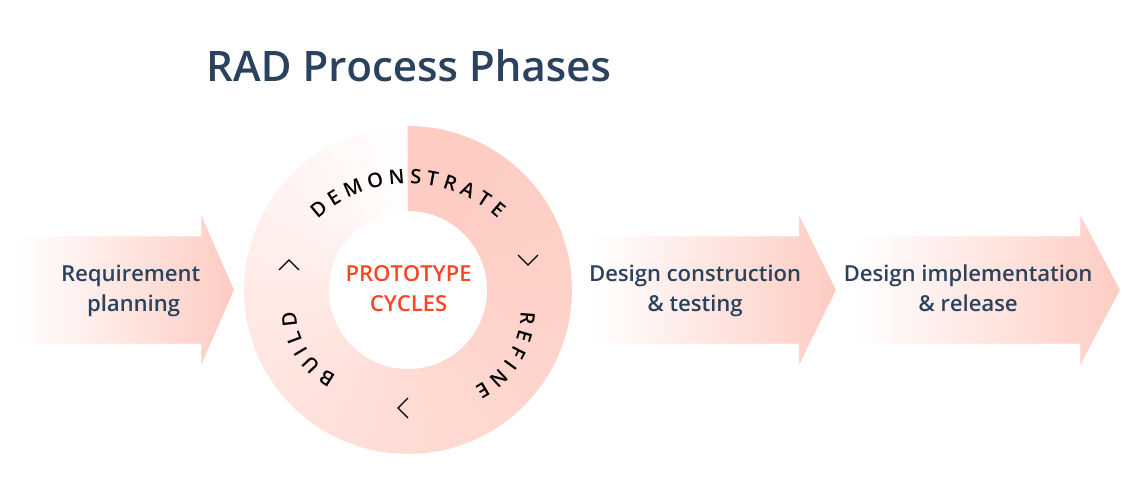
* Dividing software: It is difficult to "divide the software and the problems in several versions that would be acceptable to the customer which can be implemented and delivered incrementally.
* Uncertain nature of customer needs: A confused user has uncertainity over his requirements, so giving him several version may change his requirement Rapidly.
* Time And Cost:As this model reduces "Time And Cost" but requirement is not gathered correctly. It will subsequently time, cost and efforts.

# 7.What is Rapid Application Development Model (RAD)?

RAD model stands for Rapid Application Development model is a parallel development of functions and subsequent integration, where each component or function is developed in parallel as if they were mini-projects.

RAD Model’s working process is same as Incremental Model. In this model,





## **Rapid Application Development Process Phases**

For decades, project management teams stuck with the traditional software development process that includes strict planning, process, and documentation.

This way, RAD allows project managers and stakeholders to arrange all project changes in real-time and implement them immediately. This allows greater efficiency, faster release and time-to-market, and effective communication.

## **Phase One: Requirement Planning**

## According to the mentioned above study around 34% of projects, schedules aren’t baselined. Baselining, in other words, approval of project planning documentation should be of the planning stage, which allows tracking the software development progress against plans.

## In the rapid app development planning stage is like a project scoping meeting that results in creating a scoping document in the result. During this stage, developers, stakeholders, and team members communicate to define the goals for the project. They also review current and potential issues to be addressed during the build.

**Tip:** This stage includes:

* Researching the current problem
* Defining project requirements and specifications
* Approval of those requirements by stakeholders
* Everyone can evaluate the goals and expectations for the project and weigh in

## **Phase Two: User Design and Prototypes**

After this first phase, it’s time to jump right into design development. This involves [building the common stages of the design process](https://www.softermii.com/blog/ux-design-process-common-steps-practices-to-follow). But for the RAD model, the main goals on this stage is to develop the customer journey and wireframes applying the [main UX deliverables](https://www.softermii.com/blog/20-ux-deliverables-methods-you-should-include-in-your-design-project) and finalize the design version through various prototype iterations.

During this phase, clients work closely with developers. In such a way, they can discuss the project requirements that are met at every step of the design process. It’s similar to customizable software development where the users test each prototype of the product at each stage to see if it meets their expectations.

All the kinks and bugs are worked out through iterative means. The development team designs a prototype, and the client and users test it. Then, all the issues are discovered and comments are considered in the next design version. This method lets developers tweak the model as they go until they devise a satisfactory design.

**Tip**: At the end of this stage you have the final tested design version that you implement into software.

## **Phase Three: Rapid Construction**

This phase's main task is to put the design prototypes into beta systems and then into a working model.

As the majority of the problems are discovered and changes are made through the [iterative design phase](https://www.enginess.io/insights/what-is-iterative-design), now. Developers can now construct the final working model more quickly than they could by following a traditional project management approach.

**Tip:** This phase includes:

* Preparation for rapid construction
* Program and application development
* Coding
* Unit, integration, and system testing

This third phase is vital because the client still has the opportunity to give input. They can suggest changes or new ideas that will solve problems as they arise.

## **Phase Four: Cutover**

This is the implementation phase when the finished product is launched.

**Tip:** At this stage, you need to care about:

* Data conversion
* Testing
* Getting client feedback
* Finalizing your system

## **When can You Use Rapid Application Development?**

* This model should be used for a system with known requirements and requiring short development time.
* It is also suitable for projects where requirements can be modularized and reusable components are also available for development.
* The model can also be used when already existing system components can be used in developing a new system with minimum changes.
* This model can only be used if the teams consist of domain experts. This is because relevant knowledge and ability to use powerful techniques is a necessity.
* The model should be chosen when the budget permits the use of automated tools and techniques required.
* 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

**Advantages –**

* Use of reusable components helps to reduce the cycle time of the project.
* Feedback from the customer is available at initial stages.
* Reduced costs as fewer developers are required.
* Use of powerful development tools results in better quality products in comparatively shorter time spans.
* The progress and development of the project can be measured through the various stages.
* It is easier to accommodate changing requirements due to the short iteration time spans.

**Disadvantages**

* The use of powerful and efficient tools requires highly skilled professionals.
* The absence of reusable components can lead to failure of the project.
* The team leader must work closely with the developers and customers to close the project in time.
* The systems which cannot be modularized suitably cannot use this model.
* Required professional developers and designers
* Customer involvement is required throughout the life cycle.
* It is not meant for small scale projects as for such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.
* Modeling skills have lots of dependencies.
* Not suitable for a low-budget project.

# 8.What is Component Based Development Model ?

**Component Based Software Engineering (CBSE)** is a process that focuses on the design and development of computer-based systems with the use of reusable software components.

The component-based assembly model uses object-oriented technologies. In object-oriented technologies, the emphasis is on the creation of classes. Classes are the entities that encapsulate data and algorithms.

In component-based architecture, classes (i.e., components required to build application) can be uses as reusable components. This model uses various characteristics of spiral model. This model is evolutionary by nature. Hence, software development can be done using iterative approach. In CBD model, multiple classes can be used. These classes are basically the prepackaged components.

**The model works in following manner:**

* **Step-1:**  
  First identify all the required candidate components, i.e., classes with the help of application data and algorithms.
* **Step-2:**  
  If these candidate components are used in previous software projects then they must be present in the library.
* **Step-3:**  
  Such preexisting components can be excited from the library and used for further development.
* **Step-4:**  
  But if the required component is not present in the library then build or create the component as per requirement.
* **Step-5:**  
  Place this newly created component in the library. This makes one iteration of the system.
* **Step-6:**  
  Repeat steps 1 to 5 for creating n iterations, where n denotes the number of iterations required to develop the complete application.

Object-oriented modeling results in a plethora of fine-grained classes, objects and relationships. It is very hard to discover reusable parts among these smaller units. The idea behind CBD is to integrate the related parts and reuse them collectively. These integrated parts are known as components.  
  
Component-based development techniques consist of non-conventional development routines, including component evaluation, component retrieval, etc. It is important that the CBD is carried out within a middleware infrastructure that supports the process, for example, Enterprise Java Beans.  
  
The key goals of CBD are as follows:

* Save time and money when building large and complex systems: Developing complex software systems with the help of off-the-shelf components helps reduce software development time substantially. Function points or similar techniques can be used to verify the affordability of the existing method.
* Enhance the software quality: The component quality is the key factor behind the enhancement of software quality.
* Detect defects within the systems: The CBD strategy supports fault detection by testing the components; however, finding the source of defects is challenging in CBD.

Some advantages and disadvantage of CBD include:

* Minimized delivery:
  + Search in component catalogs
  + Recycling of pre-fabricated components
* Improved efficiency:
  + Developers concentrate on application development
* Improved quality:
  + Component developers can permit additional time to ensure quality
* Minimized expenditures

The specific routines of CBD are:

* Component development
* Component publishing
* Component lookup as well as retrieval
* Component analysis
* Component assembly

**CBSE Framework Activities**  
Framework activities of Component Based Software Engineering are as follows:-

1. **Component Qualification:**  
   This activity ensures that the system architecture define the requirements of the components for becoming a reusable component.Reusable components are generally identified through the traits in their interfaces. It means “the services that are given, and the means by which customers or consumers access these services ” are defined as a part of the component interface.
2. **Component Adaptation:**  
   This activity ensures that the architecture defines the design conditions for all component and identifying their modes of connection. In some of the cases, existing reusable components may not be allowed to get used due to the architecture’s design rules and conditions. These components should adapt and meet the requirements of the architecture or refused and replaced by other, more suitable components.
3. **Component Composition:**  
   This activity ensures that the Architectural style of the system integrates the software components and form a working system. By identifying connection and coordination mechanisms of the system, the architecture describes the composition of the end product.
4. **Component Update:**  
   This activity ensures the updation of reusable components. Sometimes, updates are complicated due to inclusion of third party (the organization that developed the reusable component may be outside the immediate control of the software engineering organization accessing the component currently.).

9.What is Rational Unified Process (RUP)

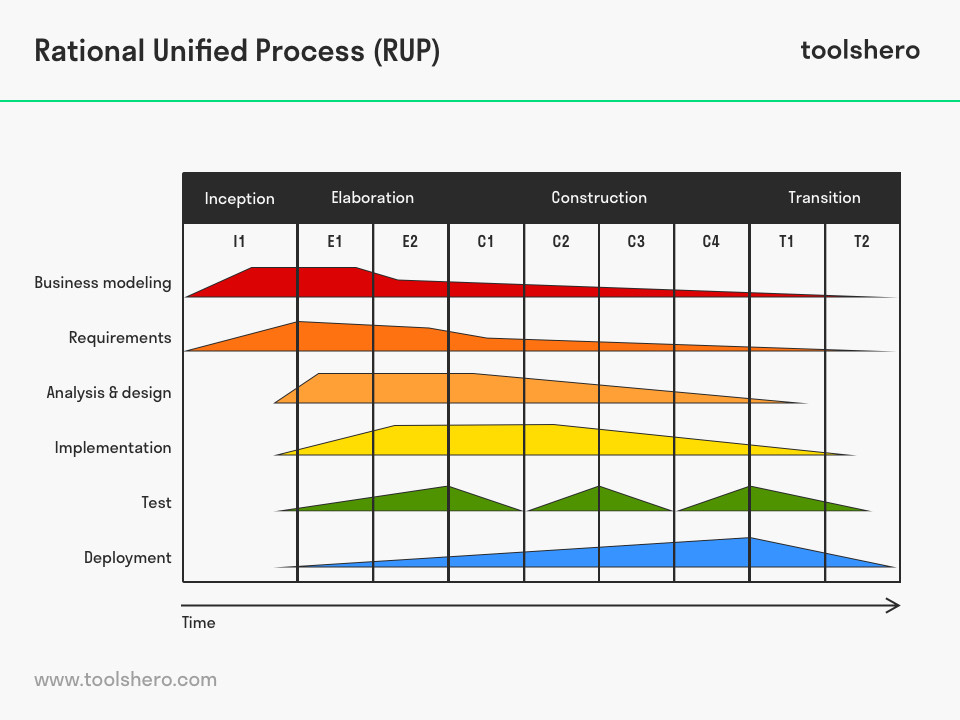
# Model ?

**Rational Unified Process (RUP)** is a software development process for object-oriented models. It is also known as the Unified Process Model. It is created by Rational Software corporation and is designed and documented using UML (Unified Modeling Language). This process is included in IBM Rational Method Composer (RMC) product. IBM (International Business Machine Corporation) allows us to customize, design, and personalize the unified process.

**A processed product** - the development team for RUP is working closely with customers, partners, groups organizations to ensure that the process is constantly updated

It uses commercially proven techniques and practices.

It is a process considered heavy and preferably applicable to large development teams and large projects, but the fact that it is extensively customizable allows it to be adapted to projects of any scale.



**Phases of RUP :**

There are total five phases of life cycle of RUP:

**1.Inception || Requirment**

* Communication and planning are main.
* Identifies Scope of the project using use-case model allowing managers to estimate costs and time required.
* Customers requirements are identified and then it becomes easy to make a plan of the project.
* Project plan, Project goal, risks, use-case model First use case (20% completed), Project description, are made.
* Project is checked against the milestone criteria and if it couldn’t pass these criteria then project can be either cancelled or redesigned.

**2.Elaboration || Analysis and Degine**

* Planning and modeling are main.
* Detailed evaluation, development plan is carried out and diminish the risks.
* Revise or redefine use-case model .. Use case (80% completed), business case, risks.
* Again, checked against milestone criteria and if it couldn’t pass these criteria then again project can be cancelled or redesigned.
* Executable architecture baseline

**3. Construction || Codding and start testing**

* Project is developed and completed.
* System or source code is created and then testing is done.
* Coding takes place.

**4.Transition || Deployment**

* Final project is released to public.
* Transit the project from development into production.
* Update project documentation.
* Beta testing is conducted.
* Defects are removed from project based on feedback from public.

**5.Production**

* Final phase of the model.
* Project is maintained and updated accordingly.

**When is rational unified process used?**

* Develop software iteratively
* Manage requirements.
* Use component-based architectures.
* Visually model software.
* Continuously verify software quality.
* Control changes to software.

## **Advantages of RUP Software Development**

* This is a complete methodology in itself with an emphasis on accurate documentation
* It is proactively able to resolve the project risks associated with the client's evolving requirements requiring careful [change request management](http://www.my-project-management-expert.com/change-request-management.html)
* Less time is required for integration as the process of integration goes on throughout the [software development life cycle](http://www.my-project-management-expert.com/software-development-life-cycle-model.html).
* The development time required is less due to reuse of components.
* There is online training and tutorial available for this process.

## **Disadvantages of RUP Software Development**

* The team members need to be expert in their field to develop a software under this methodology.
* The development process is too complex and disorganized.
* On cutting edge projects which utilise new technology, the reuse of components will not be possible. Hence the time saving one could have made will be impossible to fulfill.
* Integration throughout the [process of software development](http://www.my-project-management-expert.com/process-of-software-development.html), in theory sounds a good thing. But on particularly big projects with multiple development streams it will only add to the confusion and cause more issues during the stages of testing

Integrating Risk Management in SDLC

Software development life cycle (SDLC) is a conceptual model for defining the tasks performed at each step of software development process. Though there are various models for SDLC, but in general SDLC comprises of following steps-

* Preliminary Analysis
* System analysis and Requirement definition
* System Design
* Development
* Integration and System Testing
* Installation, Operation and Acceptance Testing
* Maintenance
* Disposal

**1.Preliminary analysis:**

* The organization’s objective
* Nature and scope of problem under study
* Propose alternative solutions and proposals after having the deep understanding of problem and what competitors are doing
* Describe costs and benefits.

**Risk Factors–**

* Establish a process and responsibilities for risk management
* Document Initial known risks
* Project Manager should prioritize the risks

**2.System analysis and requirement definition:**  
This step is very important for a clear understanding of customer expectation and requirement. Thus it is very important to conduct this phase with utmost care and given due time as any possible error will cause the failure of entire process. Following are the series of steps that are conducted during this stage.

* Identify assets that need to be protected and assigning their criticality in terms of confidentiality, integrity and availability
* Identify threats and resulting risk to those assets
* Determine existing security controls to reduce that risks

In short we can divide this phase into five sub phases: Feasibility study, requirement elicitation, requirement analysis, requirement validation, requirement documentations.

* **Feasibility Study**

**Risk Factors -**

* + Project manager often make a mistake in estimating cost, time, resources and scope of the project. Unrealistic budget, time, inadequate resources and unclear scope often leads to project failure.
  + Unrealistic Budget
  + Unrealistic Schedule
  + Insufficient resources
  + Unclear project scope
* **Requirement Elicitation**.

**Risk Factors -**

* + Incomplete requirements
  + Inaccurate requirements
  + Unclear requirements
  + Ignoring non functional requirements
  + Conflicting user requirements
  + Gold plating
  + Unclear description of real operating environment
* **Requirement Analysis Activity**

**Risk Factors**

* Non verifiable requirements
* Infeasible requirement
* Inconsistent requirement
* Non traceable requirement
* Unrealistic requirement
* **Requirement Validation Activity**

This involves validating the requirements that are gathered and analyzed till now to check whether they actually defines what user want from the system.

**Risk Factors**

* Misunderstood domain specific terminology
* Using natural language to express requirements
* **Requirement Documentation Activity**

This steps involves creating a Requirement Document(RD) by writing down all the agreed upon requirements using formal language. RD serves as a means of communication between different stakeholders.

**Risk Factors**

* Inconsistent requirements data and RD
* Non modifiable RD

**3.System Design**  
This is the second phase of SDLC wherein system architecture must be established and all requirements that are documented needs to be addressed.  
In this phase the system (operations and features) is described in detail using screen layouts, pseudocode, business rules, process diagrams etc.  
**Support from Risk Management Activities**

* Accurate Classification of assets criticality
* Planned controls accurately identified

**It involves a series of seven activities:**

* Examine requirement Document
* Risk Factors 🡪RD is not clear for developers
* Choosing the architectural design method activity
* Risk Factors 🡪 Improper Architectural Design method
* Choosing the programming language activity
* Risk Factors 🡪 Improper choice of programming language
* Constructing physical model activity 🡪Risk Factors :
  + Complex system
  + Complicated design
  + Large size components
  + Unavailability of expertise for reusability
  + Less reusable components
* Verifying design activity -Verifying design means to ensure that the design is the correct solution for the system under construction and it meets all user requirements.

**Risk Factors**

* + Difficulties in verifying design to requirements
  + Many feasible solutions
  + Incorrect design
* Specifying design activity 🡪This activity involves following main tasks:
* Identify the components and defines data flow between them
* For each identified component state its function, data input, data output and resource utilization.

**Risk Factors**

* + Difficulty in allocating functions to components
  + Extensive specification
  + Omitting data processing functions
* Documenting design activity – In this phase design document(DD) is prepared. This will help to control and coordinate the project during implementation and other phases.  
  **Risk Factors**
  + Incomplete DD(Design document)
  + Inconsistent DD
  + Unclear DD
  + Large DD

**4. Development**  
This stage involves the actual coding of the software as per the agreed upon requirements between developer and client .This phase involves two steps: Coding and Unit Testing.

* Coding Activity  
  **Risk Factors**
  + Unclear design document
  + Lack of independent working environment
  + Wrong user interface and user functions developed
  + Programming language incompatible with architectural design
  + Repetitive code
  + Modules developed by different programmers
* **Unit Testing Activity –** Each module is tested individually to check whether it meets the specified requirements or not and perform the functions it is intended to do. **Risk Factors** 
  + Lack of fully automated testing tools.
  + Code not understandable by reviewers.
  + Coding drivers and stubs.
  + Poor documentation of test cases.
  + Testing team not experienced.
  + Poor regression testing.

**5.Integration and System Testing:**  
In this phase, first all modules are independently checked for errors, bugs. Then they are related with their dependents and dependency is checked for errors and finally all modules are integrated into one complete software and checked as a whole for bugs.

This phase includes three activities: Integration Activity, Integration Testing Activity, System Testing Activity.

We will be discussing these activities in a bit detail along with risk factors in each activity.

* Integration Activity – In this phase individual units are combined into one working system.  
  **Risk Factors**
* Difficulty in combining components.
* Integrate wrong versions of components.
* Omissions.
* Integration Testing Activity – After integrating the components next step is to test whether the components interface correctly and to evaluate their integration. This process is known as integration testing.  
  **Risk Factors** 
  + Bugs during integration.
  + Data loss through interface.
  + Desired functionality not achieved.
  + Difficulty in locating and repairing errors.
* System Testing Activity – In this step integrated system is tested to ensure that it meets all the system requirements gathered from the users.  
  **Risk Factors** 
  + Unqualified testing team.
  + Limited testing resources.
  + Not possible to test in real environment.
  + Testing cannot cope up with requirements change.
  + System being tested is not testable enough.

**6. Installation, Operation and Acceptance Testing:**  
This is the last and longest phase in SDLC. In this system is delivered, installed, deployed and tested for user acceptance.

This phase involves three activities: Installation, Operation, Acceptance Testing.

* Installation Activity – The software system is delivered and installed at the customer site.  
  **Risk Factors** 
  + Problems in installation.
  + Change in environment.
* Operation Activity--Here end users are given training on how to use software system and its services.  
  **Risk Factors**
* New requirements emerge.
* Difficulty in using system.
* Acceptance Testing Activity – Delivered system is put into acceptance testing to check whether it meets all user requirements or not.  
  **Risk Factors** 
  + User resistance to change.
  + Too many software.
  + Insufficient data handling.
  + Missing requirements.

**7. Maintenance:**

In this stage, the system is assessed to ensure it does not become obsolete. This phase also involves continuous evaluation of the system in terms of performance and changes are made time to time to initial software to make it up-to date.

Errors, faults discovered during acceptance testing are fixed in this phase. This step involves making improvements to the system, fixing errors, enhancing services and upgrading software.

**Risk Factors –**

* Budget overrun: Finding errors and fixing them involves repeating few steps in SDLC again. Thus exceeding the budget.
* Problems in upgrading: Constraints from end user or not so flexible architecture of the system forces it to be not easily maintainable.

**8. Disposal:**  
In this phase, plans are developed for discarding system information, hardware and software to make transition to a new system. The purpose is to prevent any possibility of unauthorized disclosure of sensitive data due to improper disposal of information. All of this should be done in accordance with the organization’s security requirements.

**Risk Factors**

* Lack of knowledge for proper disposal.
* Lack of proper procedures.

**10.Agile Software Development Model in SDLC**

### **What is Agile?**

### **Agile is a software development approach where a self-sufficient and cross-functional team works on making continuous deliveries through iterations and evolves throughout the process by gathering feedback from the end users.**

### Agile is not a set of rules. Agile is not a set of guidelines. Agile is not even a methodology. Rather, Agile is a set of principles that encourage flexibility, adaptability, communication and working software over plans and processes.

Agile is an alternative to the transitional waterfall development model. There can be changes in the requirements. Agile supports these changing requirements. Therefore, it is adaptable and flexible. It divides the product into small incremental builds. In this methodology, a set of requirements are taken. Then the analysis, designing, coding, and testing is done for that section. Then, there is a meeting with the customer to get feedback. If it is successful, the team can move on to another set of requirements. And, this process repeats until developing the entire software. Each iteration takes one to three weeks. These time periods are called time boxes or sprints. **Agile** is **suitable** or may be possible: Small to medium-sized **software** developments.

**What is Agile Software Development?**

**The Agile model was primarily designed to help a project to adapt to change requests quickly. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task agility is required. Agility is achieved by fitting the process to the project, removing activities that may not be essential for a specific project. Also, anything that is wastage of time and effort is avoided.**

**Agile development model is also a type of** [**Incremental model**](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/)**.**

In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed. The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only. At a time one iteration is planned, developed and deployed to the customers. Long-term plans are not made.

Agile model is the combination of iterative and incremental process models.

* **Agile is suitable or may be possible: Small to medium-sized software developments.**

**The Agile development process generally looks something like this:**

* Establish a few initial requirements
* Design
* Develop
* Test
* Deploy
* Evaluate the resulting micro outcome (ie a product feature)
* Collect feedback on what’s been presented thus far
* Establish new requirements for next sprint based on feedback & repeat the micro outcome cycles until you achieve the final desired product

Agile software development allows the team to work together more efficiently and effectively in developing complex projects. It consists of practices that exercise iterative and incremental techniques which are easily adopted and display great results.

# What is Agile Methodology?

[**Agile methodology**](https://en.wikipedia.org/wiki/Agile_software_development) **is a type of project management process, mainly used for software development, where demands and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers.**

The Agile methodology is a collection of principles that value adaptability and flexibility. Agile  aims to provide better responsiveness to changing business needs and therefore focuses on enabling teams to deliver in workable increments.

Agile Methodology is a people-focused, results-focused approach to software development that respects our rapidly changing world. It’s centered around adaptive planning, self-organization, and short delivery times. It’s flexible, fast, and aims for continuous improvements in quality, using tools like Scrum and eXtreme Programming.

**There are several agile methodologies in practice across the world. We are going to learn more in detail about four of the most popular ones.**

* Extreme Programming (XP)
* Scrum
* Dynamic Systems Development Method (DSDM)
* Feature-Driven Development (FDD)
* Crystal Methods
* Lean Development (LD)
* Adaptive Software Development (ASD)

## **What is the Agile Manifesto?**

**The** [**Agile Manifesto**](https://blog.zenkit.com/uncovering-the-agile-manifesto-af9435cc4f00) **is a declaration of the values and principles expressed in Agile methodology. Made up of four foundational values and 12 key principles, it aims to help uncover better ways of developing software by providing a clear and measurable structure that promotes iterative development,** [**team collaboration**](https://blog.zenkit.com/10-must-haves-for-successful-team-collaboration-d8735963943e)**, and change recognition.**

The values and principles of the ‘[Manifesto for Agile Software Development](http://agilemanifesto.org/)’ are:

### **Values:** The agile software development emphasizes on four core values.

Here, value is defined as the financial benefit that an organization receives for expenditures. When measured, value can encompass an entire organization, or be constrained, such as to a single division or product line. Regardless, it must encompass those areas affected by the expenditure.

* 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

### **Principles:**

* 1. Customer satisfaction through early and continuous software delivery
  2. Accommodate changing requirements throughout the development process
  3. Frequent delivery of working software
  4. Collaboration between the business stakeholders and developers throughout the project
  5. Support, trust, and motivate the people involved
  6. Enable face-to-face interactions
  7. Working software is the primary measure of progress
  8. Agile processes to support a consistent development pace
  9. Attention to technical detail and design enhances agility
  10. Simplicity
  11. Self-organizing teams encourage great architectures, requirements, and designs
  12. Regular reflections on how to become more effective

Human factors

* Competence
* Common focus
* Collaboration
* Decision-making ability
* Fuzzy-problem solving ability
* Mutual trust and respect
* Self-organization

**What is involved in Timeboxing?**

Depending on the particular work, a timebox can be of a few weeks, days, or even hours. It entails getting started on the tasks in an organized manner without wasting any time. Here are some of the steps to implement timeboxing.

1. Decide on the task that must be completed
2. Establish a time limit for the tasks
3. Focus on finishing the task within the specified time limit
4. Stop working on the task when the time period is complete
5. Review the work, and establish what has been achieved

**The Benefits of Timeboxing**

**Following fixed timeboxes helps improves the focus, which in turn results in increased productivity.** It avoids the prospect of postponing the product demo until all the work is completed. In addition, metrics can be used as an input in review meetings for further improvements.

**Timeboxing help project team members to avoid wasting time on superfluous aspects of the project. It helps them keep laser focused on the main deliverables.** This results in achieving the assigned task in the minimum timeframe that subsequently results in reduced costs and increased profitability for the company.

**Lastly, timeboxing help in prioritizing important tasks. The team will complete the most important tasks first. If the team knows that there is a fixed time period, it will avoid wasting time on unnecessary work**. This will result in completing more important tasks that will make a major positive impact on the project outcome.

**The importance of timeboxing in effective**[**task management**](https://www.copperproject.com/)**cannot be denied.** However, you must use a good online tasks management software to implement the technique. Using the right task management tool will matter as it will determine how favorable the impact on the end project goal is.

**Timebox & scope**

**Time boxes for iterations are usually one to four weeks, with most teams opting for a two-week iteration length. Any fixed time iteration is going to be somewhat artificial. Software development tasks don’t all lend themselves to neat little two weeks sprints. So then, why do we time box?**

**We time box in order to get good at the scope boxing. Time boxing means to work on a task for a specific length of time. Scope boxing means to work on a small task until it’s complete. But people coming from the Waterfall world often don’t have the skills to break large tasks down into smaller tasks, so time boxing is usually easier to do than scope boxing.**

By making a time box efficiently small, it means that we can only work on the core parts of a feature. This often leads to keeping development more focused around building a single thing at a time. Building this skill helps us get good at managing scope and building the smallest increment we can.

Once we get good at breaking tasks down into smaller components, we don’t necessarily need to use time boxes anymore and we can move on to more of a pull system like Kanban rather than working in fixed iterations.

This very much depends on the nature of the work we’re doing. Maintenance tasks and bug fixing might lend itself more toward a scope box approach, whereas new development might be more appropriate for a time box approach.

In the end, time boxing is artificial but it can be a valuable discipline. Like the beat of a drum in a song helps keep all the musicians in sync, time boxing helps keep everyone on the team working at the same cadence.

Ultimately, the most mature teams that I see tend to move toward scope boxing rather than time boxing.

**What is an assumption agile?**

An **assumption** in requirements is an assertion on which a given requirement or set of requirements depends.

Examples include: **Assumption**: The test database will be available on 9/1/2019. **Assumption**: The SME will join the team as scheduled on 6/1/2019.

Here is our list of 12 common assumptions agile methodologies are making about your company:

1. **Teams stay together over time**

**2. People are specializing generalists there is less of a need to assign them to multiple teams to keep the busy.**

**3. People are engaged and motivated**

**4. You’ve got the top 10%**

**5. Teams deliver products**

**6. Projects come to teams**

**7. Team are loosely coupled**

**8. Teams have minimal external dependencies**

**9. Fully engaged customers**

**10. Established architecture**

**11. Minimal process governance**

**12. Team Members are co-located**

https://dzone.com/articles/12-key-agile-assumptions

### **When to use Agile model:**

* When new changes need to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
* To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
* Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users’ needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
* Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

### Agile Advantages

* The methodology is focused on the client process. It makes sure that the client is continuously involved in every stage. Customers have an opportunity to see the work being delivered and to make decisions and changes throughout development.
* Customers have a b sense of ownership by working extensively and directly with the project team throughout the project.
* Development is often more user-focused. This is the result of more and frequent direction from the customer.
* It guarantees that the quality of the development is clearly maintained.
* The [Agile offshore development](https://mobilunity.com/blog/how-agile-offshore-development-works/) process is completely based on incremental progress.
* Teams are extremely and self-organized and motivated. This provides better results for development projects.
* The clients know exactly what is complete and what is not that decreases risks in the development process.
* The deliverables are flexible. Stakeholders can set deliverables by order of importance.
* It offers higher quality and user-friendly products. Clients can provide feedback after each sprint, so the products developed using this method often end up being very user-friendly.

### **Agile Disadvantages**

* It is not useful for small development projects.
* It provides intense commitment. Unlike the traditional approach, Agile development only works well when the entire development team is committed to the project for the duration.
* The projects can easily go off track if project managers are not sure what outcome they want.
* The meetings require the presence of an expert to take important decisions.
* Communication issues. The method requires a high level of collaboration, so development projects using this methodology will also require a high level of communication.
* The cost of implementing the methodology is a little more compared to other methodologies.
* The high degree of customer involvement may present problems for some clients who may not have the time or interest in this type of participation.
* The close working relationships in the iterative approach require working in the same physical space, which is not always possible.

Agile Developments also fails at times due to Unrealistic Expectations – Agile actually is and what it can help you achieve. Agile is commonly believed to be a set a practices, processes and tools, when in fact, Agile is really more of a mind-set and culture.

## Difference Between Agile and Iterative

### **Definition**

Agile is a model of software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers.

Iterative is a model that starts with a simple implementation of a small set of software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

* **Continuous Improvement** :

In agile, the next sprint depends on the previous sprint, whereas in iterative, the next iteration depends on the previous iteration

* **Customer Involvement** Importantly, in agile, there is a meeting with the customer at the end of each sprint while in iterative, there is a meeting with the customer at the end of each iteration.
* **Plan review**

Another difference between agile and iterative is that in agile, the team can review during mid-sprint planning while I terative relies on baseline iteration plan.

### **Effort estimation**

Furthermore, in agile, the scrum master facilitates, and the team does the estimation, whereas in iterative, the project manager provides the estimation for each iteration.

### **Ownership**

In agile, the entire team is responsible for completing the tasks, but in iteration, the project manages is responsible for completing each iteration.

### **Participants of the testing**

Also, in agile, anyone in the team can identify, prepare and execute test cases while in iterative, the testers identify, prepare and execute test cases.

* **Testing**

In agile, test case preparation and execution starts after analyzing, designing and coding, whereas in iterative, test case preparation and execution starts after analyzing, designing and coding.

* **Delivery**

Moreover, in agile, demonstration and delivering working software at the end of every sprint while in iterative, delivering working software at the end of each iteration.

## Difference between Agile and Waterfall Model:

|  |  |
| --- | --- |
| **Agile** | **Waterfall** |
| It separates the project development lifecycle into sprints. | Software development process is divided into distinct phases. |
| It follows an incremental approach | Waterfall methodology is a sequential design process. |
| Agile methodology is known for its flexibility. | Waterfall is a structured software development methodology  so most times it can be quite rigid. |
| Agile can be considered as a collection of many different projects. | Software development will be completed as one single project. |
| Agile is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed. | There is no scope of changing the requirements once  the project development starts. |
| Agile methodology, follow an iterative development approach because of this planning, development, prototyping and other software development phases may appear more than once. | All the project development phases like designing, development, testing, etc.  are completed once in the Waterfall model. |
| Test plan is reviewed after each sprint | The test plan is rarely discussed during the test phase. |
| Agile development is a process in which the requirements are expected to change and evolve. | The method is ideal for projects which have definite requirements  and changes not at all expected. |
| In Agile methodology, testing is performed concurrently with software development. | In this methodology, the "Testing" phase comes after the "Build" phase |
| Agile introduces a product mindset where the software product satisfies needs of its end customers and changes itself as per the customer's demands. | This model shows a project mindset and places its focus completely on accomplishing the project. |
| Agile methdology works exceptionally well with Time & Materials or non-fixed funding. It may increase stress in fixed-price scenarios. | Reduces risk in the firm fixed price contracts by getting risk agreement at  the beginning of the process. |
| Prefers small but dedicated teams with a high degree of coordination and synchronization. | Team coordination/synchronization is very limited. |
| Products owner with team prepares requirements just about every day during a project. | Business analysis prepares requirements before the beginning of  the project. |
| Test team can take part in the requirements change without problems. | It is difficult for the test to initiate any change in requirements. |
| Description of project details can be altered anytime during the SDLC process. | Detail description needs to implement waterfall software  development approach. |
| The Agile Team members are interchangeable, as a result, they work faster. There is also no need for project managers because the projects are managed by the entire team | In the waterfall method, the process is always straightforward so,  project manager plays an essential role during every stage of SDLC. |

Most agile development methods break product development work into small increments that minimize the amount of up-front planning and design. Iterations, or sprints, are short time frames ([timeboxes](https://en.wikipedia.org/wiki/Timeboxing" \o "Timeboxing)) that typically last from one to four weeks.

* **Agile is suitable or may be possible: Small to medium-sized software developments.**

### **When to Use the Waterfall Methodology**

### The Waterfall methodology prevails when the project is constrained by cost and/or time, and the requirements and scope are well understood. In these cases, the Waterfall methodology provides a set of processes that are built on the principle of approval of the previous phase.

### The bottom line is that the Waterfall methodology does a better job at providing a well-defined feature set within a constrained budget or timeline.

### **When to Use an Agile Methodology**

### Agile wins the day when the product team is unsure at the onset what needs to be built or they wish to discover what should be built based on adjustments they make along the way. Agile will produce more features in a shorter period of time and also gives the team more flexibility throughout the process so that they can take advantage of opportunities as the project unfolds.

## What are the Similarities Between Waterfall Model and V Model?

* Both Waterfall Model and V Model are software process models.
* Both Waterfall model and V models are not suitable for large and complex projects.

## What is the Difference Between Waterfall Model and V Model?

|  |  |
| --- | --- |
| Waterfall Model vs V Model | |
| The waterfall model is a relatively linear sequential design approach to develop software projects. | The V model is a model in which the execution of the phases happens in a sequential manner in a v shape. |
| **Methodology** | |
| The waterfall model is a continuous process. | The V model is a simultaneous process. |
| **Total Defects** | |
| In waterfall model, the total defects in the developed software are higher. | In v model, the total defects in the developed software are lower. |
| **Defect Identification** | |
| In waterfall model, the defects are identified in the testing phase. | In v model, the defects are identified from the initial phase. |

|  |  |  |
| --- | --- | --- |
| Definition | V-Model is the development model in which the entire model is divided into various sub development phase where corresponding testing phase for each development phase is practices. In other words we can say that for every stage in the development cycle, there is an associated testing phase and corresponding testing phase of the development phase is planned in parallel. | On other hand Waterfall model there is first development of the application and after which the different testing of application take place. In other words we can say that in WaterFall the complete process is divided into several phases among which one phase should be completed in order to reach the next phase and testing is almost at end phase of the development. |
| 2 | Type/Nature | As mentioned above that in V-Model the execution of the phases i.e., development and testing happens in a sequential manner so type of V-Model is Sequential/Parallel in nature. | On other hand WaterFall Model is a relatively linear sequential design approach as each phase should be completed in order to reach the next phase. So type of this model is Continuous in nature. |
| 3 | Testing and Validation | In V-Model each development phase get tested at its own level and hence no pending testing occurs in this model also if any validation requires to be implemented then it could be implemented at that phase. | On other hand in case of WaterFall Model the testing occurs after development is completed and thus if any missing validation is identified to be implemented then first that phase of development needs to be recognized and then that validation get implemented. |
| 4 | Cost and Complexity | As sequential phases need to be functional in case of V-Model hence the cost is higher as compared to that of WaterFall Model also the complexity is more than WaterFall. | On other hand in WaterFall Model due to linear development only one phase of development is operational and hence cost and complexity is low as compared to that of V-Model. |
| 5 | Defects | In V-Model the probability of total number of defects in the development of application is low as testing is done in parallel to the development. | On other hand in WaterFall Model the probability of total number of defects in the development of application is high as testing is done post development. |

## Problem with the Waterfall Model

As you may observe, that **testing in the model starts only after implementation is done.**

But if you are working in the large project, where the systems are complex, it's easy to miss out the key details in the requirements phase itself. In such cases, an entirely wrong product will be delivered to the client and you might have to start afresh with the project OR if you manage to note the requirements correctly but make serious mistakes in design and architecture of your software you will have to redesign the entire software to correct the error.

Assessments of thousands of projects have shown that **defects introduced during requirements & design make up close to half of the total number of defects.**

Also, the **costs of fixing a defect increase across the development lifecycle. The earlier in life cycle a defect is detected, the cheaper it is to fix it.** As they say, "A stitch in time saves nine."

## Solution: The V Model

To address this concern, **the V model of testing** was developed where **for every phase, in the Development life cycle there is a corresponding Testing phase**

### Waterfall Model Vs. V Model

The main difference between waterfall model and V model is that in waterfall model, the testing activities are carried out after the development activities are over. On the other hand in V model, testing activities start with the first stage itself. In other words, waterfall model is a continuous process, while the V model is a simultaneous process.

|  |  |
| --- | --- |
| **Waterfall model** | **V-model** |
| The cost of Waterfall model is low. | V-model is expensive. |
| Simplicity of Waterfall model is simple. | Simplicity of V-model is Intermediate. |
| Flexibility of Waterfall model is Rigid. | Flexibility of V-model is Little flexible. |
| Waterfall model is a sequential execution process. | It is also a sequential execution process. |
| Waterfall model’s steps move in a linear way. | V-model’s steps don’t move in linear way. | Re-usability of Waterfall model is Limited. | V-model can be Re-use for some extent. |
| User involvement in Waterfall model is only in beginning. | User involvement in V-model is also only in beginning. |  |  |
| In Waterfall model testing activities start after the development activities are over. | In V-model testing activities start with the first stage. |  |  |
| Guarantee of success through Waterfall model is low. | Guarantee of success through V-model is high. |  |  |
| Waterfall model is a continuous process. | V-model is a simultaneous process. |  |  |
| Software made using Waterfall model, the number of defects are less in comparison of software made using V-model. | Software made using V-model, the number of defects are greater in comparison of software made using Waterfall model. |  |  |
| Requirement specification in Waterfall model is necessary in beginning. | Requirement specification in V-model is also necessary in beginning. |  |  |
| Waterfall model is less used now-a-days in software engineering. | V-model is widely used in software engineering. |  |  |