1. **Waterfall Model:**

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.



Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The sequential phases in Waterfall model are −

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

**Applications use Waterfall Model:**

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Ample resources with required expertise are available to support the product.
* The project is short.

**Advantages:**

* Simple and easy to understand and use
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.
* Process and results are well documented.

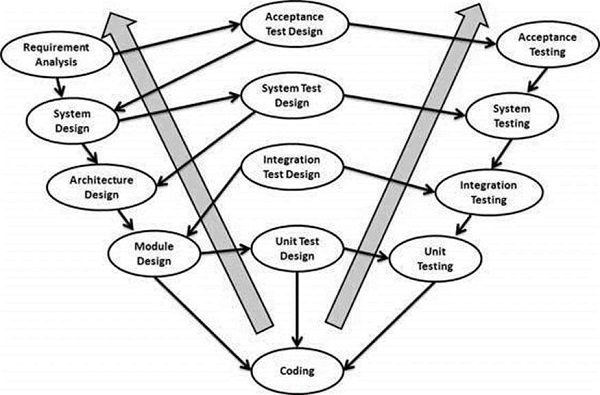
**Dis-Advantages:**

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

1. **V - Model:**

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**.

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



## Verification Phases

There are several Verification phases in the V-Model, each of these are explained in detail below.

### **1. Business Requirement Analysis**

This is the first phase in the development cycle where the product requirements are understood from the customer’s perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement.

### **2. System Design**

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development.

### **3. Architectural Design**

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as **High Level Design (HLD)**.

### **4. Module Design**

In this phase, the detailed internal design for all the system modules is specified, referred to as **Low Level Design (LLD)**. It is important that the design is compatible with the other modules in the system architecture and the other external systems.

## Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.

The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

## Validation Phases

The different Validation Phases in a V-Model are explained in detail below.

### **1. Unit Testing**

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

### **2. Integration Testing**

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

### **3. System Testing**

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

### **4. Acceptance Testing**

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

**Applications use V-Model:**

Requirements have to be very clear before the project starts, because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain.

The following pointers are some of the most suitable scenarios to use the V-Model application.

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

**Advantages:**

The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today’s dynamic world, it becomes very expensive to make the change.

The advantages of the V-Model method are as follows −

* This is a highly-disciplined model and Phases are completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Simple and easy to understand and use.
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

**Dis-Advantages:**

* High risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.
* Once an application is in the testing stage, it is difficult to go back and change a functionality.

1. **Incremental Model:**

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a [“multi-waterfall” cycle](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/).  Cycles are divided up into smaller, more easily managed modules.

In this model, each module passes through the requirements, design, implementation and [testing](http://tryqa.com/what-is-a-software-testing/) phases. A working version of software is produced during the first module, so you have working software early on during the [software life cycle](http://tryqa.com/what-are-the-software-development-life-cycle-phases/). Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.





**When to use Incremental Model:**

* This model can be used when the requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some details can evolve with time.
* There is a need to get a product to the market early.
* A new technology is being used
* Resources with needed skill set are not available

**Advantages:**

* Generates working software quickly and early during the software life cycle.
* This model is more flexible – less costly to change scope and requirements.
* It is easier to test and debug during a smaller iteration.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

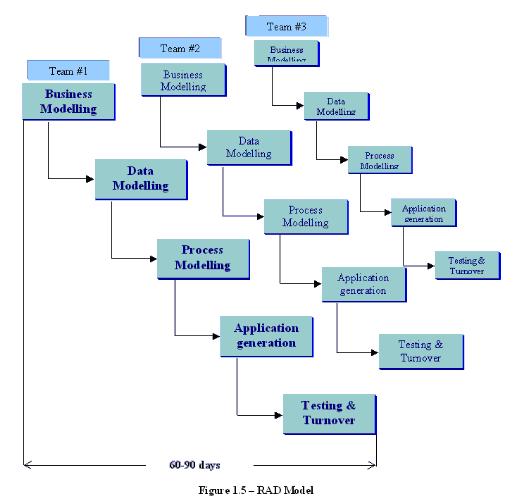
**Dis-Advantages:**

* Needs good planning and design.
* Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
* Total cost is higher than [waterfall](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/).

1. **RAD Model:**

RAD model is Rapid Application Development model. It is a type of [incremental model](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/). In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype.

This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.



**1. Business modelling:**

### The information flow is identified between various business functions. **2. Data modelling:**

### Information gathered from business modelling is used to define data objects that are needed for the business. **3. Process modelling:**

### Data objects defined in data modelling are converted to achieve the businessinformation flow to achieve some specific business objective. Description are identified and created for CRUD of data objects. **4. Application generation:**

### Automated tools are used to convert process models into code and the actual system. **5. Testing and turnover:**

### Test new components and all the interfaces.

**Advantages:**

* Reduced development time.
* Increases reusability of components
* Quick initial reviews occur
* Encourages customer feedback
* Integration from very beginning solves a lot of [integration issues](http://tryqa.com/what-is-system-integration-testing/).

**Disadvantages:**

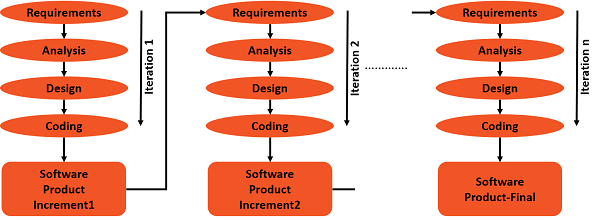
* Depends on strong team and individual performances for identifying business requirements.
* Only system that can be modularized can be built using RAD
* Requires highly skilled developers/designers.
* High dependency on modelling skills
* Inapplicable to cheaper projects as cost of modelling and automated code generation is very high.

**When to use RAD model:**

* RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
* It should be used if there’s high availability of designers for modelling and the budget is high enough to afford their cost along with the cost of automated code generating tools.

1. **Iterative Model:**

In an Iterative Incremental model, initially, a partial implementation of a total system is constructed so that it will be in a deliverable state. Increased functionality is added. Defects, if any, from the prior delivery are fixed and the working product is delivered. The process is repeated until the entire product development is completed. The repetitions of these processes are called iterations. At the end of every iteration, a product increment is delivered.



**Advantages:**

* You can develop prioritized requirements first.
* Initial product delivery is faster.
* Customers gets important functionality early.
* Lowers initial delivery cost.
* Each release is a product increment, so that the customer will have a working product at hand all the time.
* Customer can provide feedback to each product increment, thus avoiding surprises at the end of development.
* Requirements changes can be easily accommodated.

**Disadvantages:**

* Requires effective planning of iterations.
* Requires efficient design to ensure inclusion of the required functionality and provision for changes later.
* Requires early definition of a complete and fully functional system to allow the definition of increments.
* Well-defined module interfaces are required, as some are developed long before others are developed.
* Total cost of the complete system is not lower.

**When to use Iterative model:**

* Most of the requirements are known up-front but are expected to evolve over time.
* The requirements are prioritized.
* There is a need to get the basic functionality delivered fast.
* A project has lengthy development schedules.
* A project has new technology.
* The domain is new to the team.

1. **Agile Model:**

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

* Planning
* Requirements Analysis
* Design
* Coding
* Unit Testing and
* Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

**What is Agile?**

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.



The most popular Agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as **Agile Methodologies**, after the Agile Manifesto was published in 2001.

Following are the Agile Manifesto principles −

* **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
* **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
* **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
* **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

**Advantages:**

* Is a very realistic approach to software development.
* Promotes teamwork and cross training.
* Functionality can be developed rapidly and demonstrated.
* Resource requirements are minimum.
* Suitable for fixed or changing requirements
* Delivers early partial working solutions.
* Good model for environments that change steadily.
* Minimal rules, documentation easily employed.
* Enables concurrent development and delivery within an overall planned context.
* Little or no planning required.
* Easy to manage.
* Gives flexibility to developers.

**Disadvantages:**

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is a very high individual dependency, since there is minimum documentation generated.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.