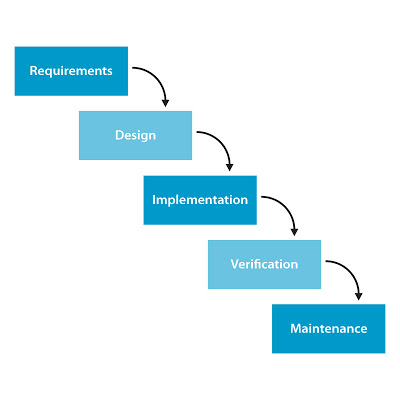
**waterfall model:**

The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back.

**When to use the waterfall model:**

* Requirements are very well known, clear and fixed.
* Product definition is stable.
* Technology is understood.
* There are no ambiguous requirements
* Ample resources with required expertise are available freely
* The project is short.

**Stages of Waterfall model:**

[](http://1.bp.blogspot.com/-gxj9EEmaKCc/UXfs6d3D2ZI/AAAAAAAAAzI/dRepueNwWK0/s1600/waterfall-model-sdlc1.jpg)

* Project Planning
* Requirements definition
* Design
* Development
* Integration and Testing
* Installation/Acceptance
* Maintenance

Advantage

* It allows for departmentalization and managerial control.
* 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.
* A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a car-wash, and theoretically, be delivered on time.

Disadvantage

* It does not allow for much reflection or revision.
* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* 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.

**V process model**

V- model means Verification and Validation model. Just like the [waterfall model](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. **V-Model** is one of the [many software development models](http://tryqa.com/what-are-the-software-development-models/).

Testing of the product is planned in parallel with a corresponding phase of development in **V-model**.

**Diagram of V-model:**



The various phases of the V-model are as follows:

**Requirements** like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a [system test](http://tryqa.com/what-is-system-testing/) plan is created.  The [test plan](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/) focuses on meeting the functionality specified in the requirements gathering.

**The high-level design (HLD)** phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An [integration test](http://tryqa.com/what-is-integration-testing/) plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

**The low-level design** **(LLD)** phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. [Component tests](http://tryqa.com/what-is-component-testing/) are created in this phase as well.

**The implementation** phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

**Coding:** This is at the bottom of the V-Shape model. Module design is converted into code by developers. [Unit Testing](http://tryqa.com/what-is-unit-testing/) is performed by the developers on the code written by them.

**Advantages of V-model:**

* Simple and easy to use.
* Testing activities like planning, [test designing](http://tryqa.com/what-is-test-design-or-how-to-specify-test-cases/) happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**Disadvantages of V-model:**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents has to be updated.

**When to use the V-model:**

* The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
* The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

**Prototype models**

The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a [software development model](http://tryqa.com/what-are-the-software-development-models/). By using this prototype, the client can get an “actual feel” of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.  Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.

The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

**Diagram of Prototype model:**



**Advantages of Prototype model:**

* Users are actively involved in the development
* Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
* Errors can be detected much earlier.
* Quicker user feedback is available leading to better solutions.
* Missing functionality can be identified easily
* Confusing or difficult functions can be identified  
  Requirements validation, Quick implementation of, incomplete, but  
  functional, application.

**Disadvantages of Prototype model:**

* Leads to implementing and then repairing way of building systems.
* Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
* Incomplete application may cause application not to be used as the  
  full system was designed  
  Incomplete or inadequate problem analysis.

**When to use Prototype model:**

* 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.

**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. Incremental model is a type of software development model like [V-model](http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/), [Agile model](http://tryqa.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/) etc.

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.

For example:

In the diagram above when we work **incrementally** we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it’s complete. As in the image above a person has thought of the application.

Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers.

Likewise in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

**Diagram of Incremental model:**



**Advantages of Incremental model:**

* 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.

**Disadvantages of Incremental model:**

* 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/).

**When to use the 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
* There are some high risk features and goals.

**Spiral model**

The spiral model is similar to the [incremental model](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/), with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral. Its one of the [software development models](http://tryqa.com/what-are-the-software-development-models/) like [Waterfall](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), [Agile](http://tryqa.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/), [V-Model](http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/).

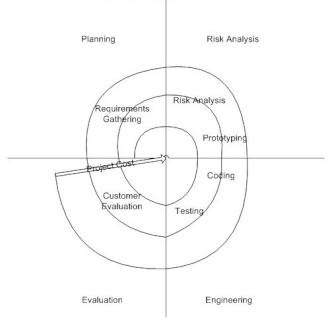
**Planning Phase:** Requirements are gathered during the planning phase. Requirements like ‘BRS’ that is ‘Bussiness Requirement Specifications’ and ‘SRS’ that is ‘System Requirement specifications’.

**Risk Analysis:** In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions.  A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

**Engineering Phase:** In this phase software is **developed**, along with [testing](http://tryqa.com/what-is-a-software-testing/) at the end of the phase. Hence in this phase the development and testing is done.

E**valuation phase:** This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

**Diagram of Spiral model:**

[](http://tryqa.com/wp-content/uploads/2012/01/Spiral-model.jpg)

**Advantages of Spiral model:**

* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission-critical projects.
* Strong approval and documentation control.
* Additional Functionality can be added at a later date.
* Software is produced early in the [software life cycle](http://tryqa.com/what-are-the-software-development-life-cycle-phases/).

**Disadvantages of Spiral model:**

* Can be a costly model to use.
* Risk analysis requires highly specific expertise.
* Project’s success is highly dependent on the risk analysis phase.
* Doesn’t work well for smaller projects.

**When to use Spiral model:**

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