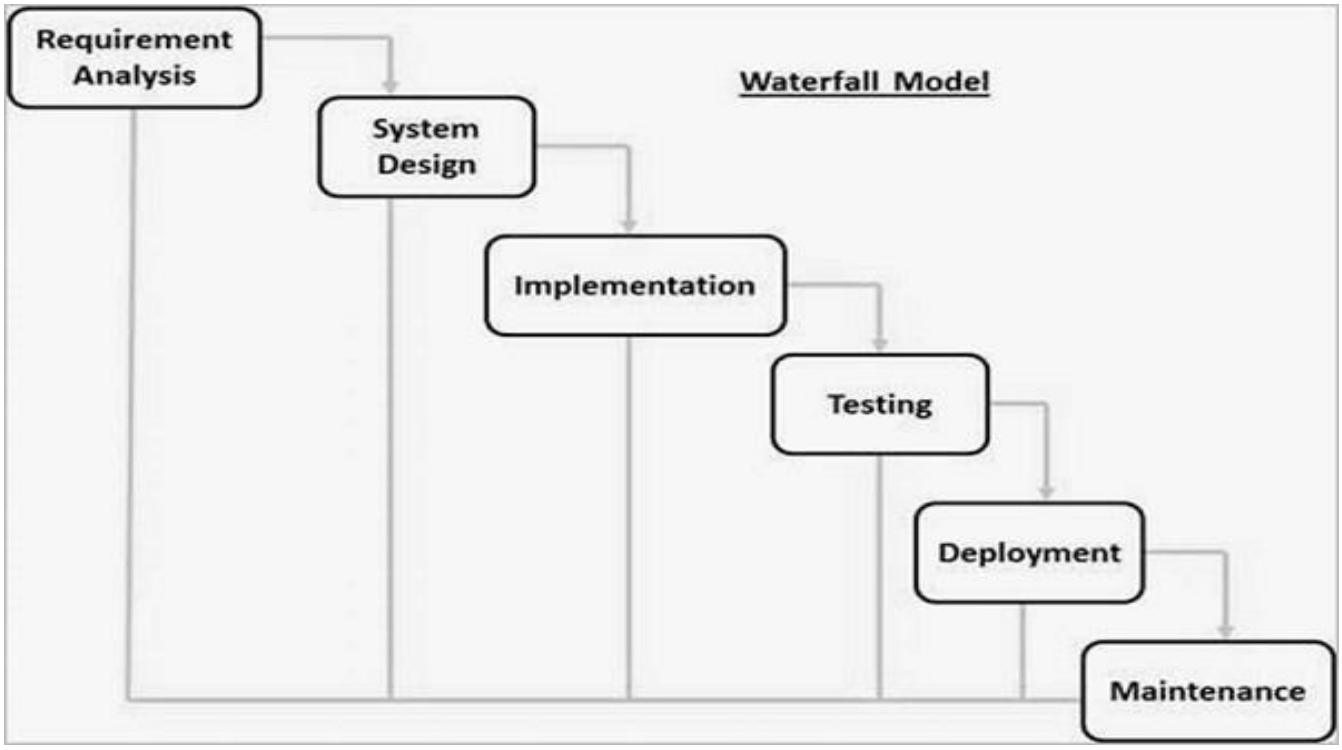
SOFTWARE DEVELOPMENT LIFE CYCLE

1.WATER FALL 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. The Waterfall model is the earliest SDLC approach that was used for software development.**
- **Any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.**
- **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.**



PHASES:

- **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.**

Application:

- **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.**

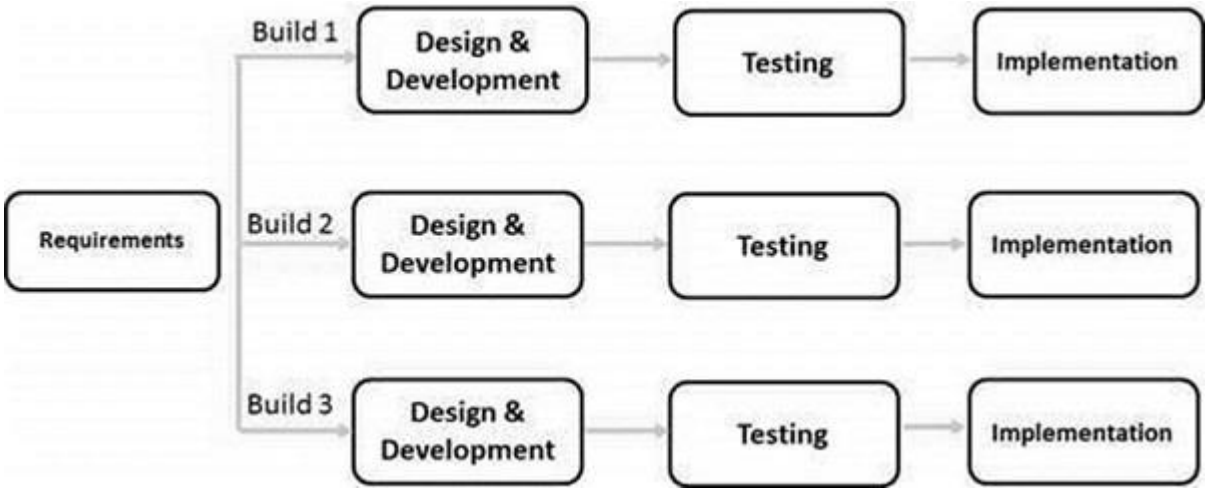
- **Process and results are well documented.**

Disadvantages:

- **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.**

2.Iterative model

- **In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements.**
- **An iterative life cycle model does not attempt to start with a full specification of requirements.**
- **Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements.**
- **This process is then repeated, producing a new version of the software at the end of each iteration of the model.**
- **At each iteration, design modifications are made and new functional capabilities are added.**
- **The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).**



Application:

- **Requirements of the complete system are clearly defined and understood.**

- **Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.**
- **There is a time to the market constraint.**
- **A new technology is being used and is being learnt by the development team while working on the project.**
- **Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.**
- **There are some high-risk features and goals which may change in the future.**

Advantages:

- **Some working functionality can be developed quickly and early in the life cycle.**
- **Results are obtained early and periodically.**
- **Parallel development can be planned.**
- **Progress can be measured.**
- **Less costly to change the scope/requirements.**
- **Testing and debugging during smaller iteration is easy.**
- **Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.**
- **Easier to manage risk - High risk part is done first.**
- **With every increment, operational product is delivered.**
- **Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.**
- **Risk analysis is better.**
- **It supports changing requirements.**
- **Initial Operating time is less.**
- **Better suited for large and mission-critical projects.**
- **During the life cycle, software is produced early which facilitates customer evaluation and feedback.**

Disadvantages:

- **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.**
- **System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.**
- **Defining increments may require definition of the complete system.**
- **Not suitable for smaller projects.**
- **Management complexity is more.**
- **End of project may not be known which is a risk.**
- **Highly skilled resources are required for risk analysis.**
- **Projects progress is highly dependent upon the risk analysis phase.**

3. Spiral model:

- Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling.
- The Spiral Model is a software development life cycle (SDLC) model that provides a systematic and iterative approach to software development.
- It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.
- The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process.
- It consists of the following phases:

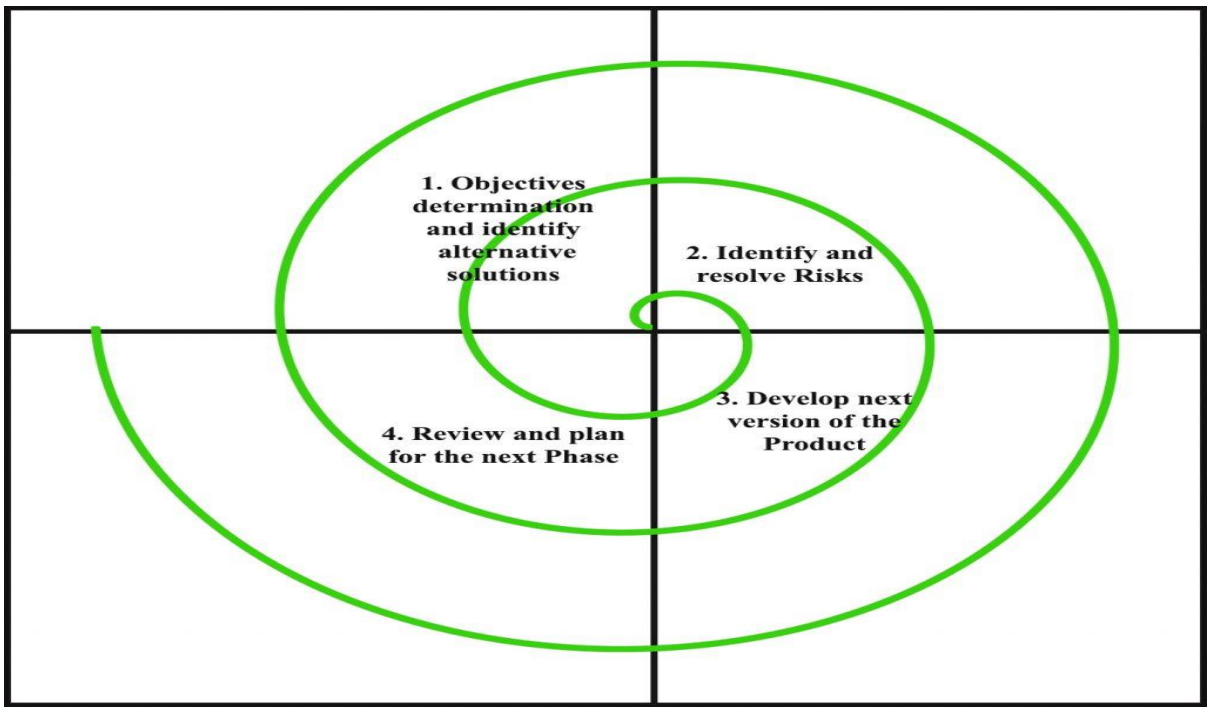
1. **Planning:** The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.

2. **Risk Analysis:** In the risk analysis phase, the risks associated with the project are identified and evaluated.

3. **Engineering:** In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

4. **Evaluation:** In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.

5. **Planning:** The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.



Functions of the four quadrants:

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

Advantage:

1. **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.

2. **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.

3. **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.

4. **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

5. **Iterative and Incremental Approach:** The Spiral Model provides an iterative and incremental approach to software development, allowing for flexibility and adaptability in response to changing requirements or unexpected events.

6. **Emphasis on Risk Management:** The Spiral Model places a strong emphasis on risk management, which helps to minimize the impact of uncertainty and risk on the software development process.

7. **Improved Communication:** The Spiral Model provides for regular evaluations and reviews, which can improve communication between the customer and the development team.

8. **Improved Quality:** The Spiral Model allows for multiple iterations of the software development process, which can result in improved software quality and reliability.

Disadvantage:

1. **Complex:** The Spiral Model is much more complex than other SDLC models.

2. **Expensive:** Spiral Model is not suitable for small projects as it is expensive.

3. **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.

4. **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.

5. **Complexity:** The Spiral Model can be complex, as it involves multiple iterations of the software development process.

6. **Time-Consuming:** The Spiral Model can be time-consuming, as it requires multiple evaluations and reviews.

7. **Resource Intensive:** The Spiral Model can be resource-intensive, as it requires a significant investment in planning, risk analysis, and evaluations.

4.V-model

- The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model.
- It is based on the association of a testing phase for each corresponding development stage.
- Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.
- The V-Model is a software development life cycle (SDLC) model that provides a systematic and visual representation of the software development process.
- It is based on the idea of a “V” shape, with the two legs of the “V” representing the progression of the software development process from requirements gathering and analysis to design, implementation, testing, and maintenance.

Phases:

1. **Requirements Gathering and Analysis:** The first phase of the V-Model is the requirements gathering and analysis phase, where the customer’s requirements for the software are gathered and analyzed to determine the scope of the project.

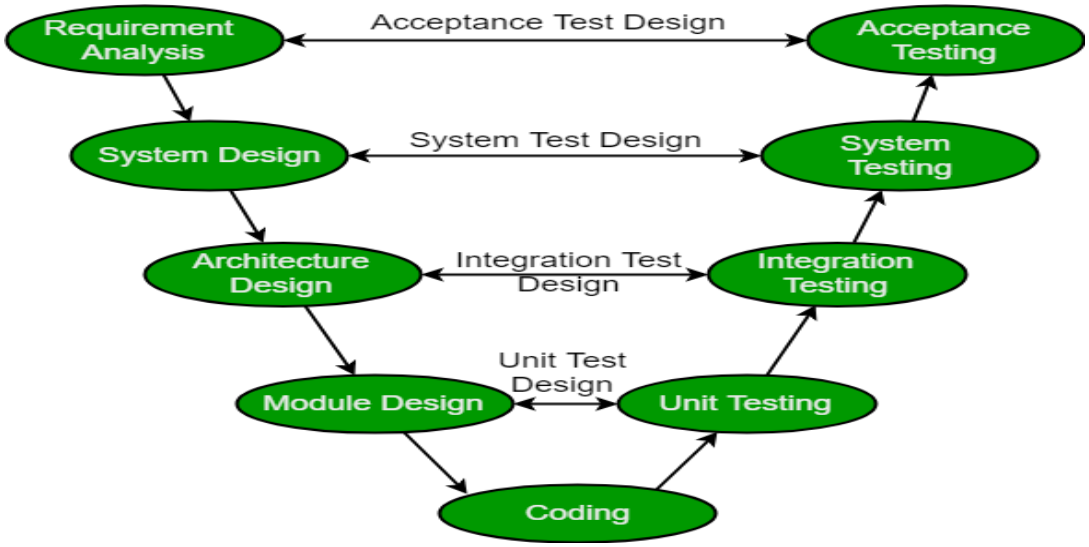
2. **Design:** In the design phase, the software architecture and design are developed, including the high-level design and detailed design.

3. **Implementation:** In the implementation phase, the software is actually built based on the design.

4. **Testing:** In the testing phase, the software is tested to ensure that it meets the customer's requirements and is of high quality.

5. **Deployment:** In the deployment phase, the software is deployed and put into use.

6. **Maintenance:** In the maintenance phase, the software is maintained to ensure that it continues to meet the customer's needs and expectations.



Verification: It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

Validation: It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.
So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.

Design Phase:

- **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.

- **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.

- **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.

- **Module Design:** In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

Testing Phases:

- **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.

- **Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.

- **System Testing:** System testing test the complete application with its functionality, inter dependency, and communication.It tests the functional and non-functional requirements of the developed application.

- **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real world.

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.
- Clear and Structured Process: The V-Model provides a clear and structured process for software development, making it easier to understand and follow.
- Emphasis on Testing: The V-Model places a strong emphasis on testing, which helps to ensure the quality and reliability of the software.

- **Improved Traceability: The V-Model provides a clear link between the requirements and the final product, making it easier to trace and manage changes to the software.**
- **Better Communication: The clear structure of the V-Model helps to improve communication between the customer and the development team.**

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.**
- **Inflexibility: The V-Model is a linear and sequential model, which can make it difficult to adapt to changing requirements or unexpected events.**
- **Time-Consuming: The V-Model can be time-consuming, as it requires a lot of documentation and testing.**
- **Overreliance on Documentation: The V-Model places a strong emphasis on documentation, which can lead to an overreliance on documentation at the expense of actual development work.**

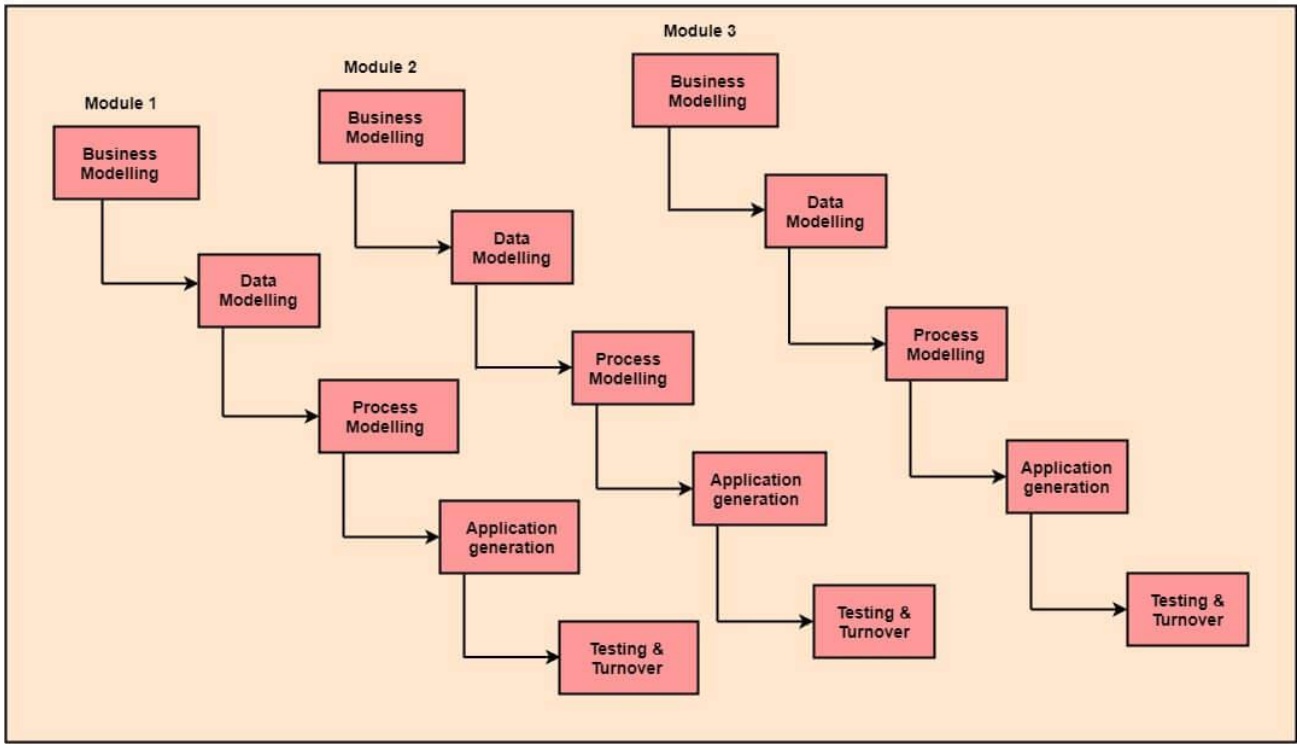
5.RAD (Rapid Application Development)

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

RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

- **Gathering requirements using workshops or focus groups**
- **Prototyping and early, reiterative user testing of designs**
- **The re-use of software components**
- **A rigidly paced schedule that refers design improvements to the next product version**
- **Less formality in reviews and other team communication**

Fig: RAD Model



PHASES:

1.Business Modelling: The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.

2. Data Modelling: The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

3. Process Modelling: The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

4. Application Generation: Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

5. Testing & Turnover: Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

When to use RAD Model?

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

Advantage:

- **This model is flexible for change.**
- **In this model, changes are adoptable.**
- **Each phase in RAD brings highest priority functionality to the customer.**
- **It reduced development time.**
- **It increases the reusability of features.**

Disadvantage:

- **It required highly skilled designers.**
- **All application is not compatible with RAD.**
- **For smaller projects, we cannot use the RAD model.**
- **On the high technical risk, it's not suitable.**
- **Required user involvement.**

6.Big Bang Model

- **The Big Bang Model comprises of focusing all the possible resources in the software development and coding, with very little or no planning.**
- **The requirements are understood and implemented as they come.**
- **Any changes required may or may not need to revamp the complete software.**
- **This model is ideal for small projects with one or two developers working together and is also useful for academic or practice projects.**
- **It is an ideal model for the product where requirements are not well understood and the final release date is not given.**

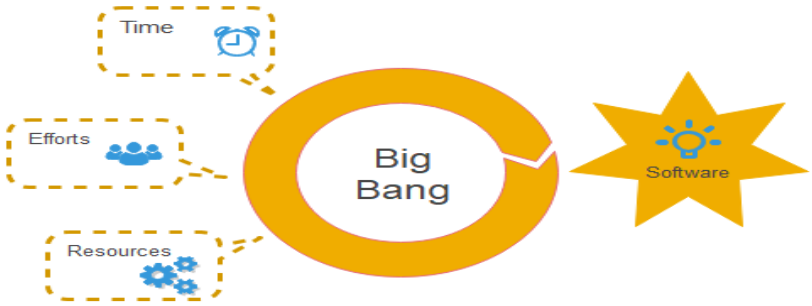


Fig. Big Bang Model

When to use it:

This SDLC model is suitable for small projects when few people are working on the project, the customer's demands are not exact and keep changing, or if it is a dummy/side-project. As there is no proper planning in this model it is considered the worst SDLC model and is highly unsuitable for large projects.

It is recommended to go for the Big Bang model only due to the following cases i.e.

1. **Developing a project for learning purposes or experiment purposes.**
2. **No clarity on the requirements from the user side.**
3. **When newer requirements need to be implemented immediately.**
4. **Changing requirements based on the current developing product outcome.**
5. **No strict guideline on product release or delivery date.**

Advantages:

- **This is a very simple model**
- **Little or no planning required**
- **Easy to manage**
- **Very few resources required**
- **Gives flexibility to developers**
- **It is a good learning aid for new comers or students.**

Disadvantages:

- **Very High risk and uncertainty.**
- **Not a good model for complex and object-oriented projects.**
- **Poor model for long and ongoing projects.**
- **Can turn out to be very expensive if requirements are misunderstood.**