Software Development Methodologies:

Assignment 1

Aim:

Software Development Life Cycle (SDLC) Models

Submitted by:

Team 12 KH – CICADA 3301

Team Members:

220940320115	Shubham Rajendrakumar Koli
220940320116	Shubham Singh
220940320117	Siddhesh Ramesh Ahire
220940320118	Sourabh Chhattani
220940320119	Souvick Saha
220940320120	Srijeet Kumar Srivastava
220940320121	Subrata Dey
220940320122	Suraj Sarode
220940320123	Tanveer Mahemud Sayyad
220940320124	Timy A Sam
220940320125	Tushar Pandey

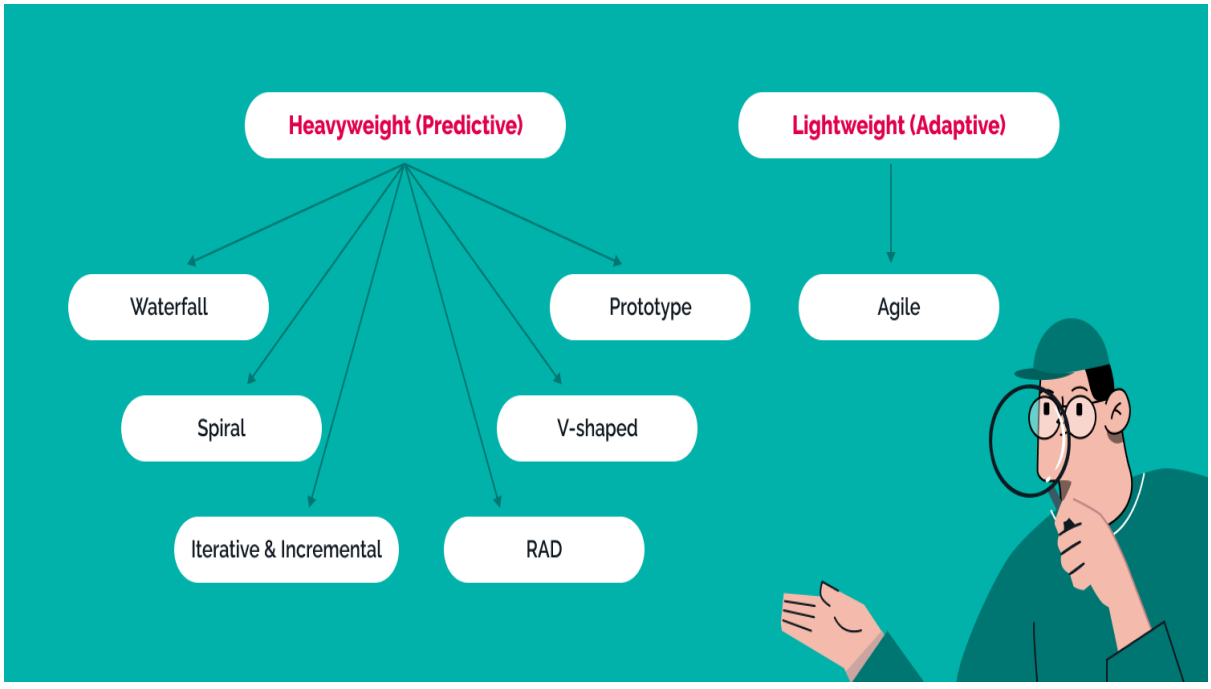
Software development life cycle

Software development life cycle or SDLC is a process used by the industry to **design, develop and test high-quality software.** SDLC aims to produce high-quality software that **meets or exceeds customer expectations and requirements, closes in the shortest possible time, and helps you estimate costs** as you know the pipelines and stages of software development.

SDLC provides **a well-structured sequence of stages** that enables an organization to swiftly deliver top-notch software that has been thoroughly tested and is suitable for production usage.

Types of SDLC models

The two primary software development life cycle models are **heavyweight** and **lightweight.**



Heavyweight or predictive processes indicate that the **scope of work is known ahead of time.**

Among the predictive (heavyweight) models are the following:

- Waterfall
- Spiral
- V-shaped
- Iterative
- RAD
- Prototype

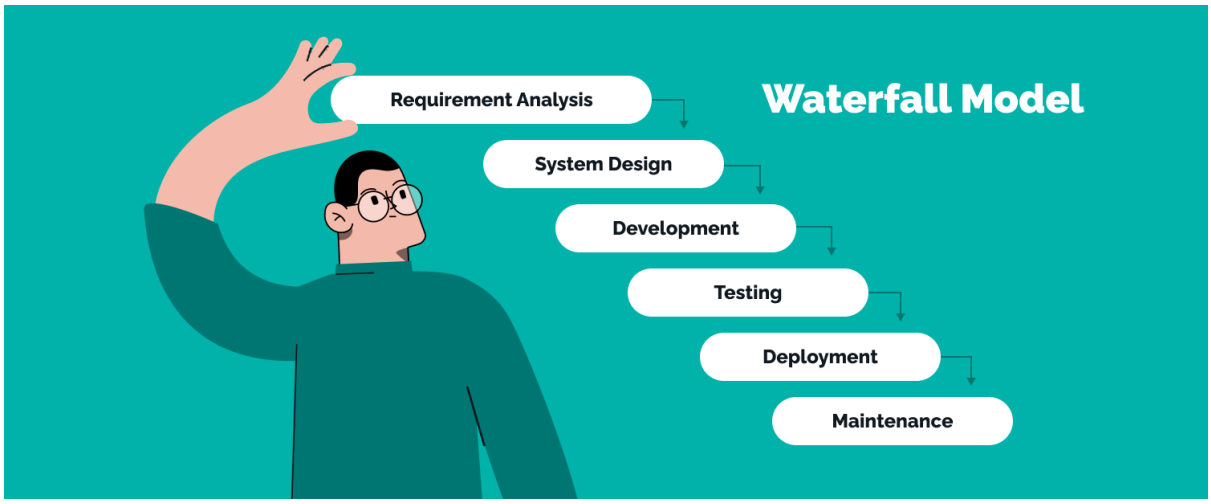
Lightweight or agile procedures are often **known as adaptable**. In terms of a software life cycle, agile approaches need far less documentation.

Adaptable (adaptive) SDLC model is:

I. Agile

Software development Life Cycle models

I. Waterfall model



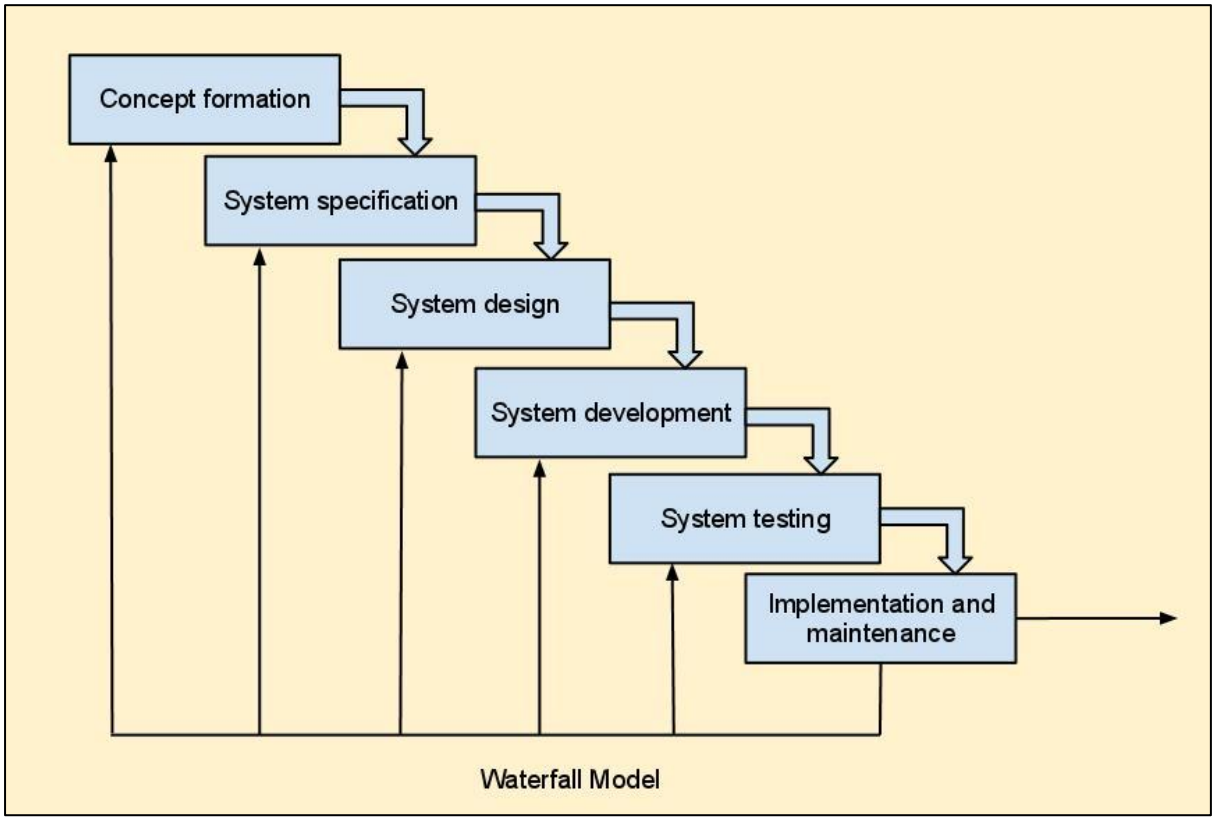
Based on Statista report, **26% of companies** use the Waterfall method in 2022.

The first SDLC methodology used for software development was the waterfall model.

A **linear sequential life cycle model** is another name for it. There are no matches between the stages in the waterfall model, and each step must be finished before the subsequent phase can start. Here, every step of the development process may start only after the one before it has been finished.

- **Requirements Analysis**: All possible requirements for the system under development are fixed.
- **System Design:** This system design helps define hardware and system requirements and helps define the overall system architecture.
- **Development:** Based on the initial design data, engineers develop the system as small programs called modules integrated into the next phase.
- **Testing:** Each module created during the implementation phase is tested before being incorporated into the overall system. Following integration, the entire system is examined for flaws and defects.

- **Deployment**: After functional and non-functional testing is finished, the product is deployed in the client environment or made available for purchase.
- **Maintenance**: Patches are provided to address these issues and enhance the product.



When to use the Waterfall model?

- If requirements are well documented**, clear, and fixed**, usually for small teams
- If technology is **clear and not dynamic**, for example for healthcare/military projects
- If the **project is short** such as a landing page

Waterfall model benefits:

- Basic, **manageable and easy-to-understand** and use model

- All SDLC life cycle phases are **completed step by step** and clearly defined
- Deliverables of each phase are precisely defined
- Each stage has tangible results and review process
- Phases are processed and **completed one at a time**
- **Simple to organize tasks**
- The process and results are documented

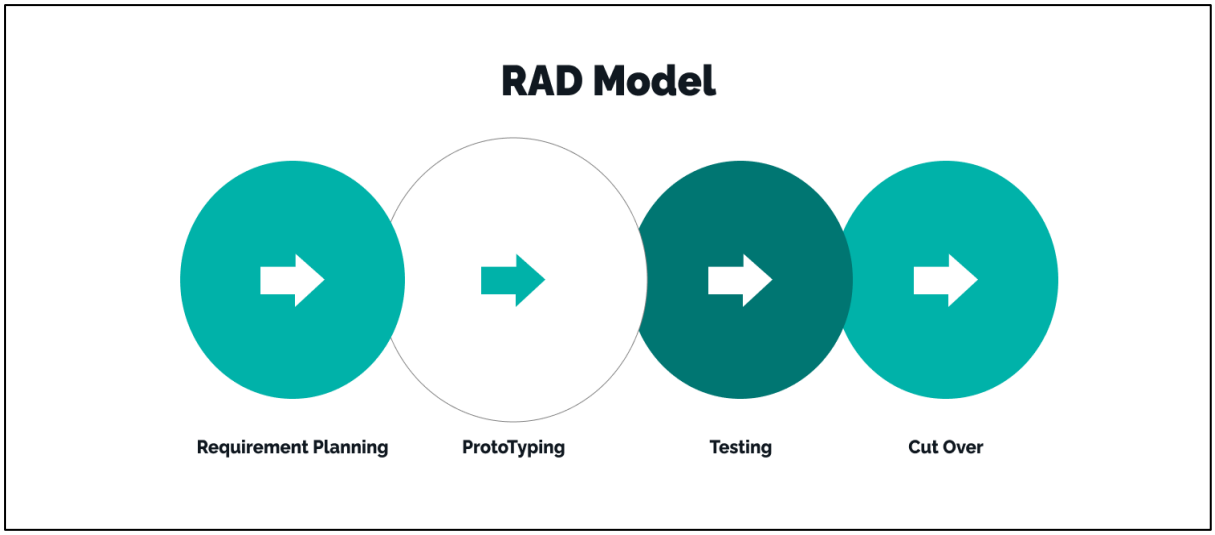
Waterfall model drawbacks:

- Lots of **risk and uncertainty**
- It **is not easy to accommodate changes** in the waterfall model
- If you have a complex, lengthier, or a bigger project, this model is not preferred
- Integration is carried out as a «big bang» at the very end
- Time-consuming and cannot be employed in short-term projects (exception landing page)
- Any change in the later SDLC life cycle stages would result in higher costs

Real Time Example:

- Use to develop enterprise applications like Customer Relationship Management (CRM) systems
- Human Resource Management Systems (HRMS)
- Supply Chain Management Systems
- Inventory Management Systems
- Point of Sales (POS) systems for Retail chains
- Development of Department Of Defence (DOD)
- military and aircraft programs followed Waterfall model in many organizations

II. RAD model



The Rapid Application Development methodology **is built on iterative development and prototyping with little preparation.** The process of creating software entails product development planning. **Functional modules are built in parallel** as prototypes for speedier product delivery and then integrated to form a full product.

The RAD breaks down the steps of analysis, design, assembly, and testing into short iterative development cycles.

There are different phases involved in the RAD model:

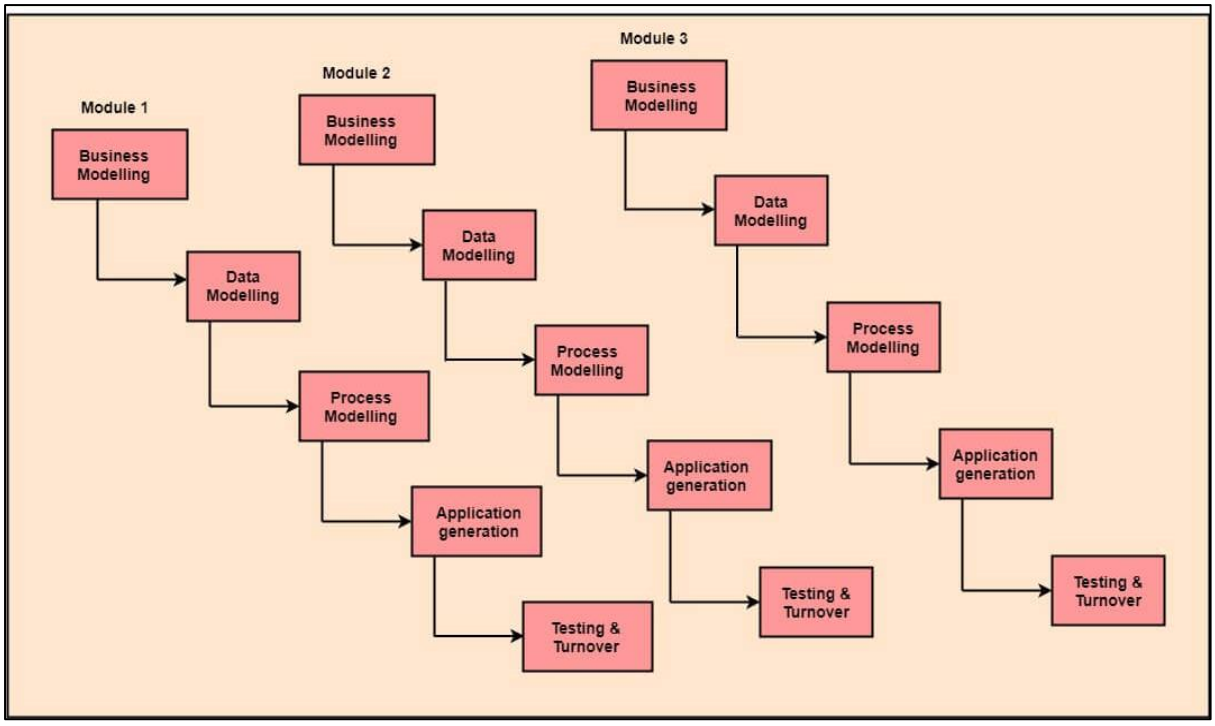
1. **Business modeling**: The information flow is identified between various business functions.

2. **Data modeling**: Information gathered from business modeling is used to define data objects that are needed for the business.

3. **Process modeling**: Data objects defined in data modeling are converted to achieve the business information 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.



When can the RAD model be used?

- If the **system is modular to deliver it gradually**
- If **designers are highly available** for modelling
- If the budget allows **automatic code-generation tools**

RAD model benefits are:

- Progress can be measured
- **Fast delivery** as it reduces the overall development time due to the ability to reuse components and parallel development
- Shorten iteration time
- Reducing development time
- Increasing the reusability of components

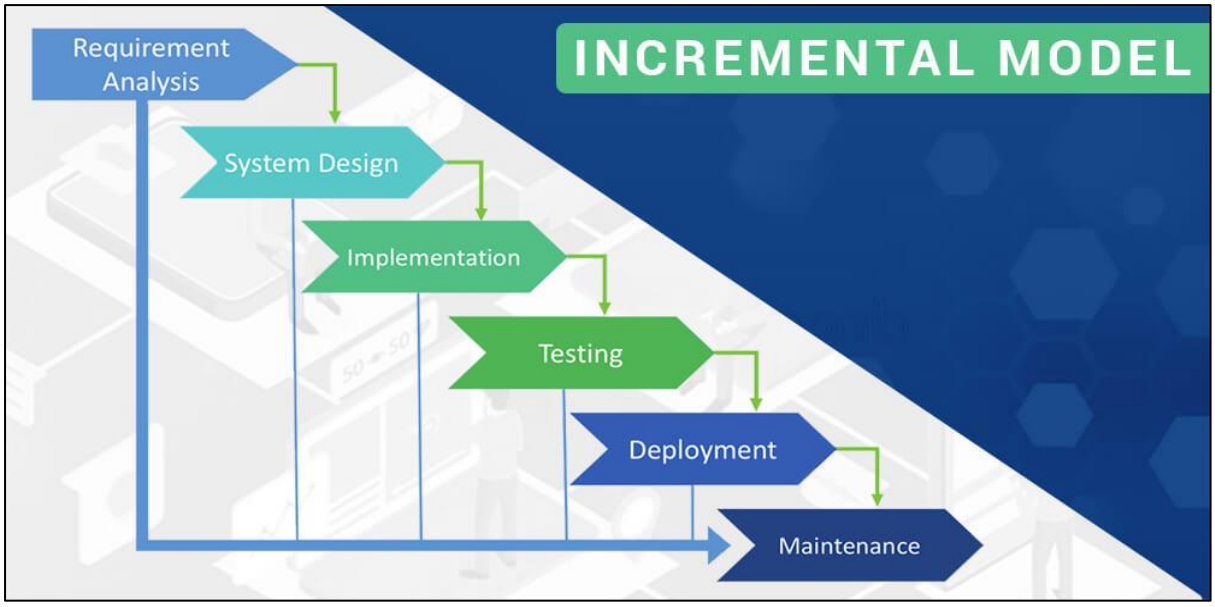
RAD model drawbacks are:

- **Dependence on technically strong team members** to determine requirements
- Modular system can be built using RAD
- **High dependence on modelling skills**
- Suitable for component-based and scalable systems, for short development projects
- Need **user involvement throughout the life cycle**

Real Time Example:

- General Goals - Bonus Subsystem
- Current System - Vehicle Subsystem
- Proposed System - VIP Subsystem
    - User Interface and Human Factors - Maintenance Subsystem
    - Documentation - Travel Subsystem
    - Hardware Consideration - Logbook Subsystem
    - Performance Characteristics - Bonus Subsystem
    - Screen Mock up - Maintenance Subsystem
    - Navigational Path for the Web Application - Bonus Subsystem
    - Dynamic Models - Logbook Subsystem

III. Incremental model

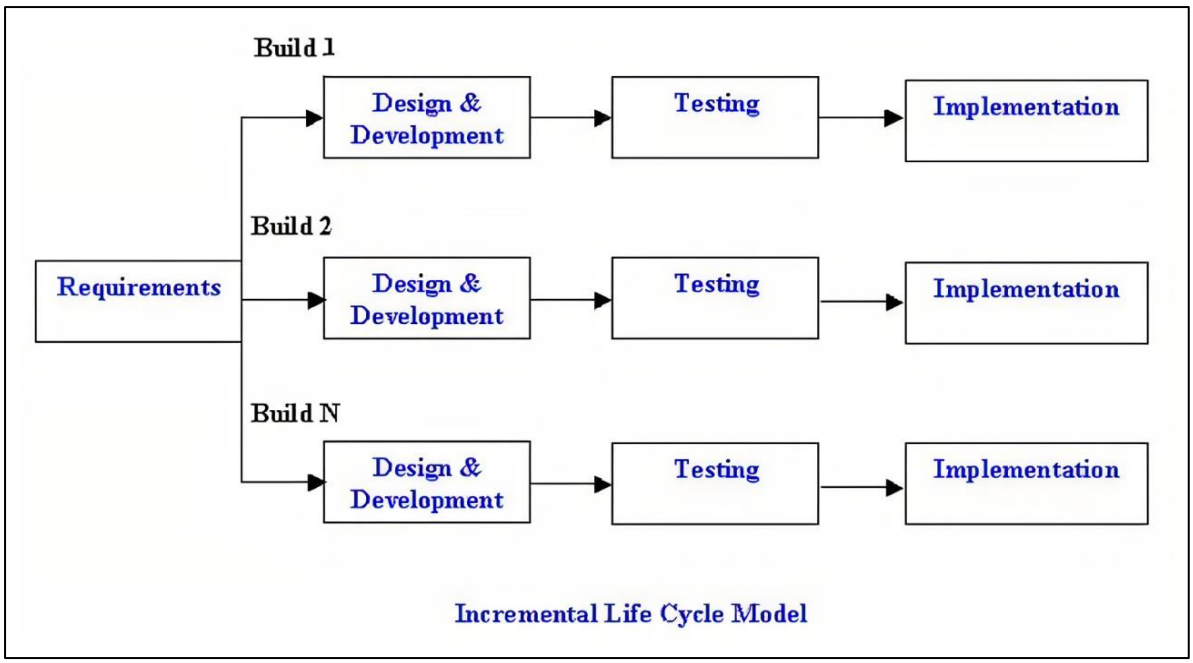


As the name suggests, the entire **system's functionality is divided into small phases** in the incremental model. After the division, all those **small phases are delivered one after the other quickly.** Thus, when new modules are added into the iterations, little to no changes are preferred in the modules earlier added.

The **development process in the incremental software development model can be either parallel or sequential.** However, with many repeated cycles, the sequential method can become costly and lengthy. While on the other hand, **the parallel method helps in adding to the speed of the delivery.**

When to use it

- The incremental model is most effective when most of the **project requirements are defined** while other details are likely to evolve with time.
- In addition, if you want the product to use new technology and **get it early in the market**, the incremental model can prove beneficial.
- Also, you should go for the model if there are some **high-risk features and goals involved** in the project.



Incremental Life Cycle Model

Pros of the model

- The software process model is **flexible and less expensive to change** scope and requirements
- The model **lowers the initial delivery cost**, and it is easier to test
- The developers find it **easier to debug and manage risks**
- In this model, the **customer can respond to each phas**e or milestone

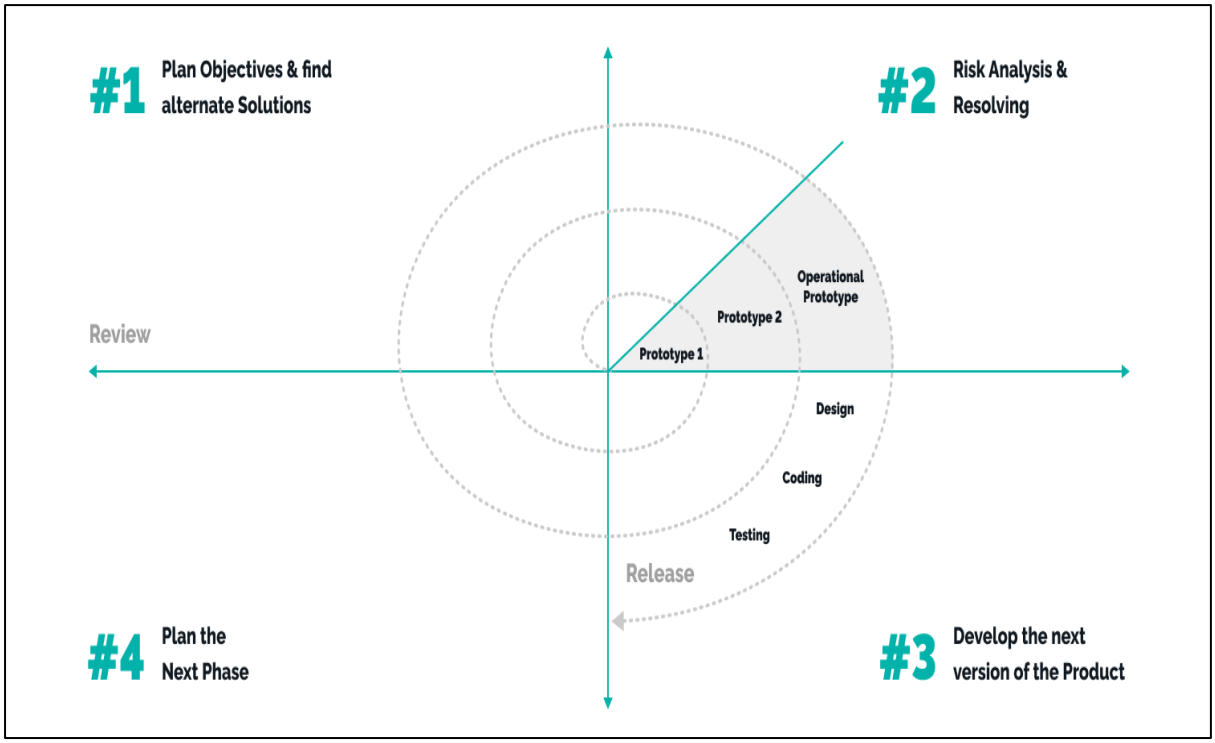
Cons of the model

- A clear and complete definition of the entire system is required before breaking it down and built incrementally
- Costly as compared to the waterfall model
- Good design and planning are required

Real Time Example:

- Microsoft uses Incremental models to bring update to their products.
- Android apps are developed using the Incremental model as the user requirements changes day by day so that they can update the existing software.

IV. Spiral model



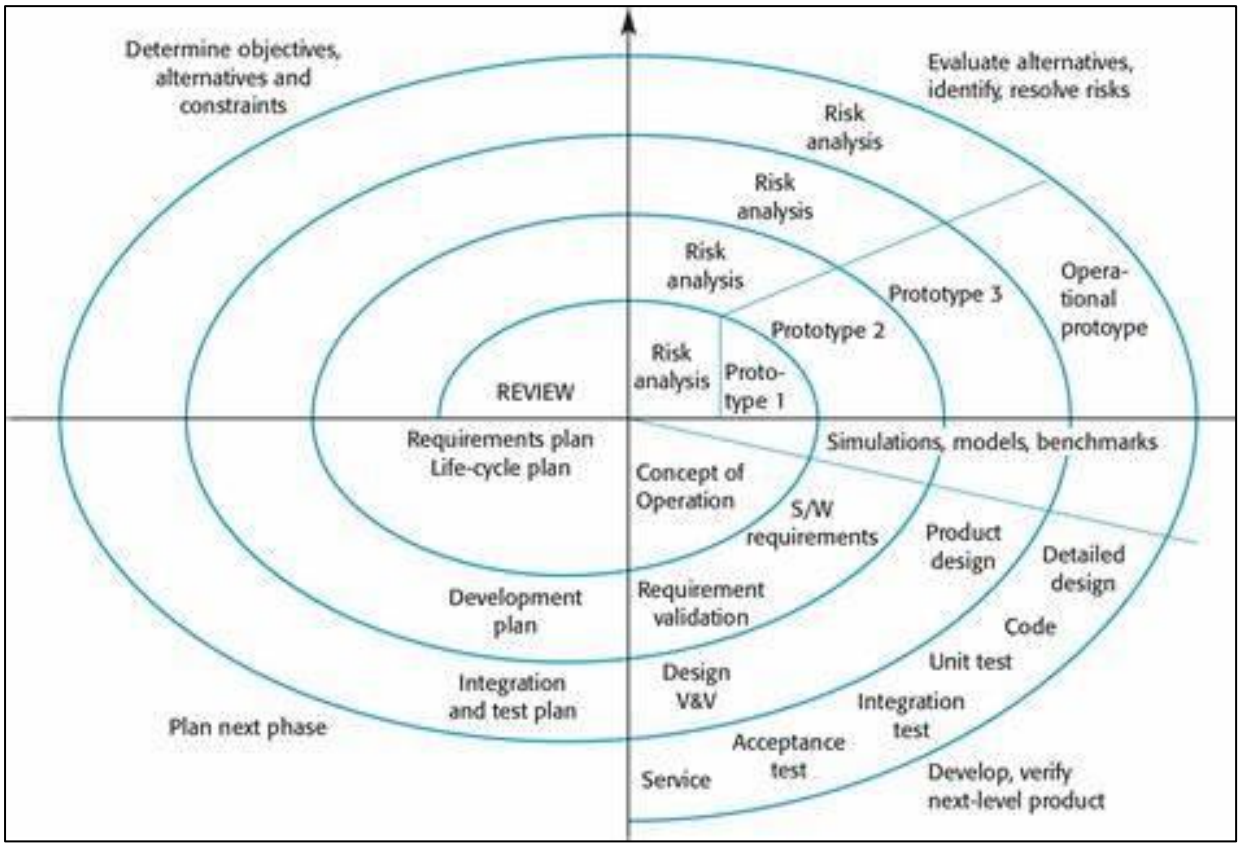
Spiral model, is a waterfall methodology with a **strong emphasis on risk analysis**, which **combines an iterative development model and a sequential linear development model.**

Fourth-generation programming languages, rapid application development prototyping, and risk analysis are added to the waterfall technique. While this method greatly reduces risk, it may not fit the budget and is used differently depending on the application.

Spiral model has the following stages:

- **Determine objectives:** This phase begins with the collection of business requirements in the basic spiral.
- **Identify risks:** It includes the identification, assessment, and monitoring of technical feasibility and management risks, such as slippage and cost overruns.
- **Plan the next iteration:** Here the development and testing of the indicated characteristics take place. The new software version is ready at the conclusion of the third quarter.

- **Development and test:** In the baseline, when the product is only thought through and the design is developed, this phase is developed by POC (Proof of Concept) to obtain customer feedback.



When to use the spiral model?

- If **risk assessment is important and budgets are tight**
- If long-term **commitment to the project change over time**
- If the **client is unsure of his requirements** or they are complex and unclear

Spiral SDLC model benefits are:

- Changes in requirements can be accommodated
- Allows wide use of prototypes
- Users are involved early in the development of the system

- Requirements can be fixed more precisely
- Development can be divided into smaller parts

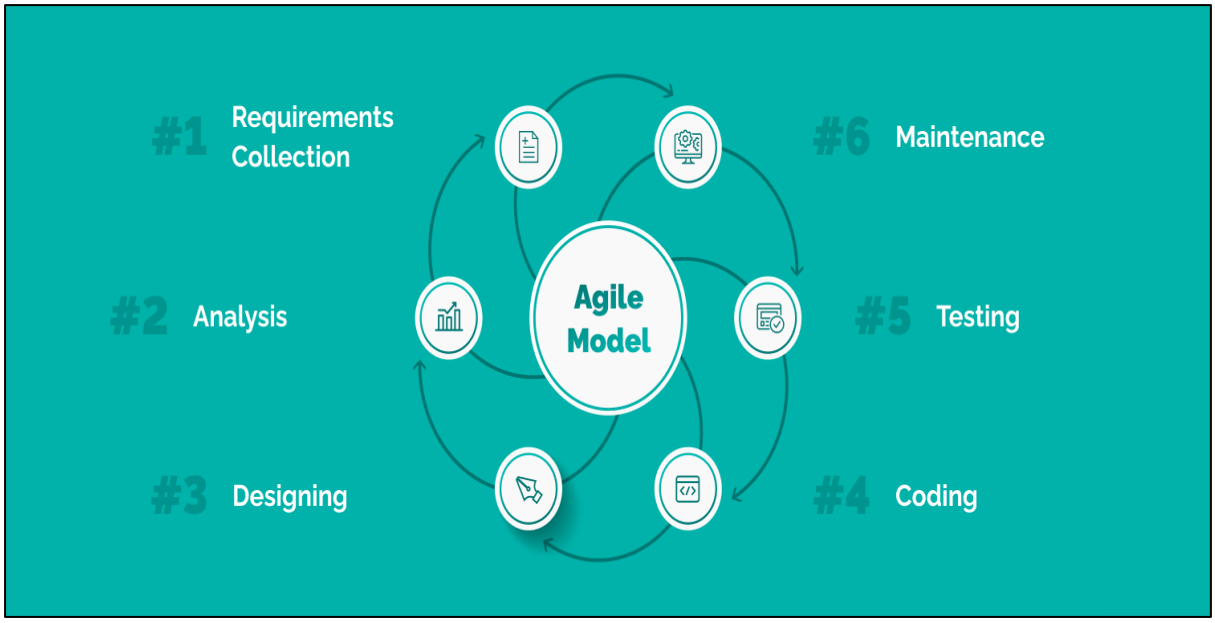
Spiral SDLC model drawbacks are:

- Strict controls are required to complete such products
- **Risk of running the spiral in an indefinite cycle**
- The end of the project may be unknown early
- Not suitable for small or low-risk projects and **can be expensive** for small projects
- The **complex process of management**
- **Lots of documentation**

Real Time Example:

- Working on the missiles or satellites is the real time example of a spiral model.
- The spiral model uses the approach of Prototyping Model by building a prototype at the start of each phase as a risk handling technique.
- Gantt chart software – GanttPRO a tool for simple task handling.
- Evolution of Microsoft Windows operating system.

v. Agile model

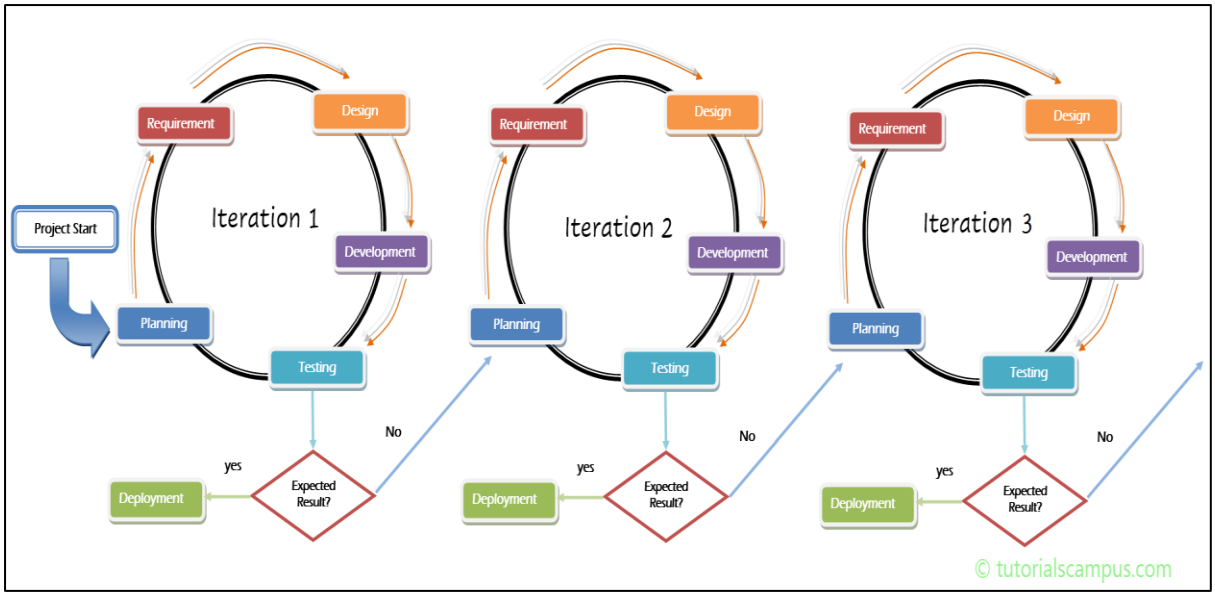


37% of worldwide companies use the Agile method in 2022. A functioning software **product is quickly delivered** as part of the Agile SDLC, emphasizing process flexibility and customer satisfaction. **Tasks are broken down into time limits** to offer specific features for release. According to the Agile software development life cycle model, each project should be managed uniquely and current methodologies should be tailored to the project's needs.

Iterations of these assemblies are supplied. Typically, **an iteration lasts one to three weeks**. Cross-functional instructions that operate concurrently across many domains are included in each iteration.

Agile standard SDLC stages are:

- Scheduling
- Requirement's analysis
- Design
- Development
- Unit and acceptance testing



Agile Model benefits are:

- Teamwork and cross-training
- Resource requirements are minimal
- **Suitable for fixed or changing** requirements
- Minimum rules, documentation, easy to use
- Provides development and delivery in the overall planned context
- Little or no planning is required
- Simple to operate

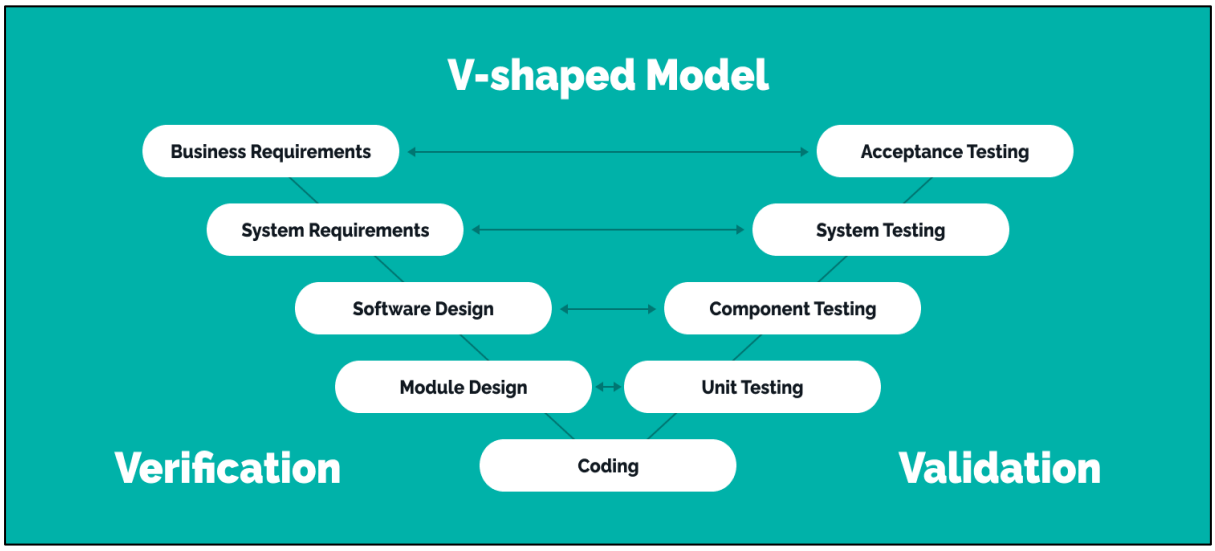
Agile Model drawbacks are:

- Not suitable for handling complex dependencies
- More risk of stability, repairability and extensibility
- **Strict delivery management to meet deadlines**
- Onboarding new team members can be difficult due to a lack of documentation

Real Time Example:

- **Restaurant orders:**
  - Preparation of some of the food before opening the shop (sprint planning)
  - continuous delivery of orders (adhoc stories)
  - number of successful orders (velocity)
- **Cricket:**
  - over (sprint length)
  - team (scrum team self-sufficient)
  - Run rate (velocity)
  - Captain/ coach (scrum master)

VI. V-shaped model



Another name for the V-Model is the **Verification and Validation Model**. This technique assumes that **development and testing take place simultaneously** since verification and validation go hand in hand.

Before the project begins, the criteria must be extremely clear because return and modification are frequently expensive. It is a well-defined model, and each step only starts after the one before it is finished.

The V-model has several steps in design and testing phase:

- **Business requirements:** The validation project planning is done, as business requirements can be used as input data for acceptance testing.
- **System Requirements:** The system design, as well as the **whole hardware and communication infrastructure** for building a product, are all incorporated.
- **Software Design:** During software system development life cycle, testing plan is developed based on system design. Doing this earlier leaves more time for the actual test to be completed later.
- **Module Design:** The **system fragments into little modules** during this stage. Specification of the modules' intricate design, commonly known as Low-Level Design (LLD).

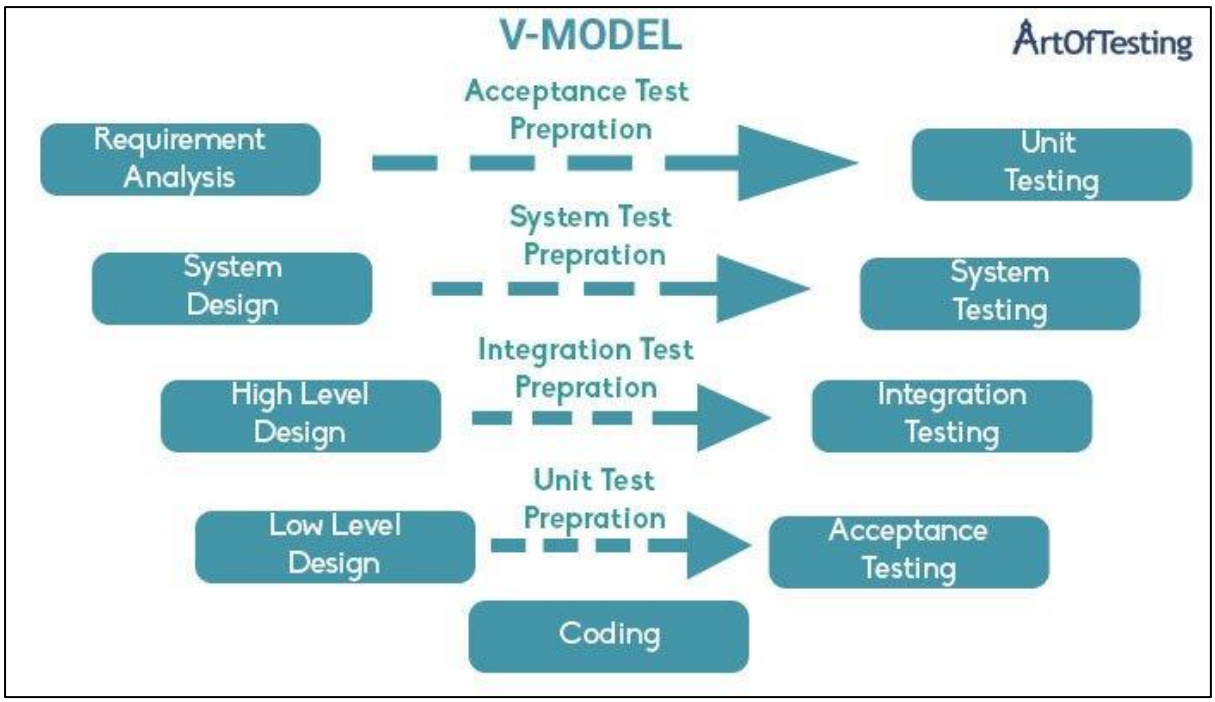
V-shaped model testing phase consists of:

- **Acceptance testing:** In a user environment that mirrors the production environment and how the system has been delivered, it satisfies user requirements, and it is prepared for deployment.
- **System testing:** It evaluates the functioning, interdependence, and communication of the entire application and both the application's functional and non-functional needs.
- **Component testing:** The system is tested as the components are integrated during integration testing. In the architecture design phase, integration testing is done. This test confirms that modules can communicate with one another.
- **Unit testing:** These unit test plans are conducted to get rid of defects at the level of the code or unit.

And links these two phases **Coding**. The actual coding of system modules developed during the design phase is considered during the coding phase. The most appropriate programming language choice is based on system and architectural requirements.

When to use a V-shaped model?

- If requirements are **clearly defined, and documented**
- If the **technology is not dynamic**



V-shaped model benefits:

- Easy to understand and apply
- Works well for small projects where requirements are very well understood.
- Each **stage has concrete results and review process**

V-shaped model drawbacks:

- **High risk and uncertainty**
- Not flexible to change
- Not a good model for complex and object-oriented, long and ongoing projects

Real Time Example:

- IT projects by federal agencies
- public-sector software projects
- In electronic and mechanical system in research and science
- software for agencies and ministries

Comparison Between the Models

Tips to choose the right SDLC model

When selecting the SDLC system development life cycle, there is a danger of becoming overwhelmed by the different possibilities. We create a table to organize all of the questions and techniques. They label the boxes with "+" or "-." Then selecting the best alternative becomes quite simple.

	Waterfall	V-Shaped	Iterative & Incremental	Agile	Spiral	Prototype
High Complexity	+/-	+/-	+	-	+	+
Extensive Documentation	+	+/-	+	-	-	+/-
Unclear Requirements	-	-	+/-	+	+	+/-
Tight Timeframes	-	-	+/-	+	+	+/-
Restricted Budget	-	-	-	+	-	-

Properties of Model	Water-Fall Model	Incremental Model	Spiral Model	RAD Model	Agile Model	V shaped Model
Planning in early stage	Yes	Yes	Yes	No	No	Yes
Returning to an earlier phase	No	Yes	Yes	Yes	Yes	No
Handle Large-Project	Not Appropriate	Not Appropriate	Appropriate	Not Appropriate	Yes	Not appropriate
Detailed Documentation	Necessary	Yes, but not much	Yes	Limited	Yes	Yes
Cost	Low	Low	Expensive	Low	Low	Expensive
Requirement Specifications	Beginning	Beginning	Beginning	Time boxed release	Time boxed release	Beginning

Flexibility to change	Difficult	Easy	Easy	Easy	Easy	Difficult
User Involvement	Only at beginning	Intermediate	High	Only at the beginning	High	Only at the beginning
Maintenance	Least	Promotes Maintainability	Typical	Easily Maintained	Easily Maintained	Easily maintained
Duration	Long	Very long	Long	Short	Depends on project	Long
Risk Involvement	High	Low	Medium to high risk	Low	Low	Moderate to high
Framework Type	Linear	Linear + Iterative	Linear + Iterative	Linear	Iterative and incremental	Sequential
Testing	After completion of coding phase	After every iteration	At the end of the engineering phase	After completion of coding	After every iteration	After every iteration
Overlapping Phases	No	Yes (As parallel development is there)	No	Yes	No	No
Re-usability	Least possible	To some extent	To some extent	Yes	Yes	No
Time-Frame	Very Long	Long	Long	Short	Long	Ideal time
Working software availability	At the end of the life-cycle	At the end of every iteration	At the end of every iteration	At the end of the life cycle	At the end of every iteration	At the end of every iteration
Team size	Large Team	Not Large Team	Large Team	Small Team	Large team	Large team