Haroon Rashid

Reg#: 16549

Semester:6th

Sessional Assignment: INFORMATION SYSTEM & DATA PROCESSING

Submitted To: Sir **MUHAMMAD ABRAR KHAN**

Q1: Define organization; also explain the structure of an organization by giving an example of a well-known organization.

Answer:

Organization: An organization or organisation is an entity, such as a company, an institution, or an association, comprising one or more people and having a particular purpose.

Structure of organization:

Organizational Structure

What Is an Organizational Structure?

An organizational structure is a system that outlines how certain activities are directed in order to achieve the goals of an organization. These activities can include rules, roles, and responsibilities.

The organizational structure also determines how information flows between levels within the company. For example, in a centralized structure, decisions flow from the top down, while in a decentralized structure, decision-making power is distributed among various levels of the organization.

Having an organizational structure in place allows companies to remain efficient and focused.

Centralized Versus Decentralized Organizational Structures

An organizational structure is either centralized or decentralized. Traditionally, organizations have been structured with centralized leadership and a defined chain of command. The military is an organization famous for its highly centralized structure, with a long and specific hierarchy of superiors and subordinates.

There has been a rise in decentralized organizations, as is the case with many technology startups. This allows companies to remain fast, agile, and adaptable, with almost every employee receiving a high level of personal agency.

Types of Organizational Structures:

Functional Structure

Four types of common organizational structures are implemented in the real world. The first and most common is a functional structure. This is also referred to as a bureaucratic organizational structure and breaks up a company based on the specialization of its workforce. Most small-to-medium-sized businesses implement a functional structure. Dividing the firm into departments consisting of marketing, sales, and operations is the act of using a bureaucratic organizational structure.

Example of Functional Structure of well-known company:

Airtel has a functional structure which is one of the best organizational structure examples. It has directors for supply chain, marketing, human resources, technology, customer care, legal works and so on. Each of these directors controls their departments and are answerable to what their functional department is responsible for. The figure explains the functional structure of

Airtel and other companies who use functional structures. The top of the structure has the board of directors, then there usually is a CEO and then comes the functional divisions such as sales, marketing, production, finance, human resources and so on. Each of these divisions has a manager and a team of employees who achieve their set goals by working together.

Figure:

Divisional or Multidivisional Structure

The second type is common among large companies with many business units. Called the divisional or multidivisional structure, a company that uses this method structures its leadership team based on the products, projects, or subsidiaries they operate. A good example of this structure is Johnson & Johnson. With thousands of products and lines of business, the company structures itself so each business unit operates as its own company with its own president.

Example of Divisional or Multidivisional Structure of well-known company:

A multidivisional organizational structure aligns a company according to individual divisions, which are based on geographic locations, products or services. For example, a moving company might create a geographic-divisional structure, which includes the Texas and California divisions. As an alternative, a manufacturing company might implement a product-based divisional structure that includes the cable assembly and product engineering divisions. In contrast, a professional services company might organize around service lines, such as the personal and business services divisions. A small business, such as an insurance agency, with a region-based organizational structure may have multiple regions spread across one or more states and, therefore, multiple domestic markets. In turn, a country-based corporation, such as a software developer, may have offices in the United States and Pakistan, but only one domestic market.

Flatarchy Structure

Flatarchy, a newer structure, is the third type and is used among many startups. As the name alludes, it flattens the hierarchy and chain of command and gives its employees a lot of autonomy. Companies that use this type of structure have a high speed of implementation

Example of Flatarchy Structure of well-known company:

The best example of this structure within a company is if the organization has an internal incubator or innovation program. Within this system, the company can operate in an existing structure, but employees at any level are encouraged to suggest ideas and run with them, potentially creating new flat teams. Lockheed Martin, according to Forbes, was famous for its skunkworks project, which helped develop the design of a spy plane.
Google, Adobe, LinkedIn and many other companies have internal incubators where employees are encouraged to be creative and innovative in order to promote the company's overall growth.

Matrix Structure

The fourth and final organizational structure is a matrix structure. It is also the most confusing and the least used. This structure matrixes employees across different superiors, divisions, or departments. An employee working for a matrixed company, for example, may have duties in both sales and customer service.

Example of Matrix Structure of well-known company:

The matrix structure followed by Starbucks coffee is one of the best organizational structure examples. The primary reason for the firm's success is its structure. Different organizational structures that combine to form Starbuck's matrix structure are divisional structure, functional structure, and team-based structure.

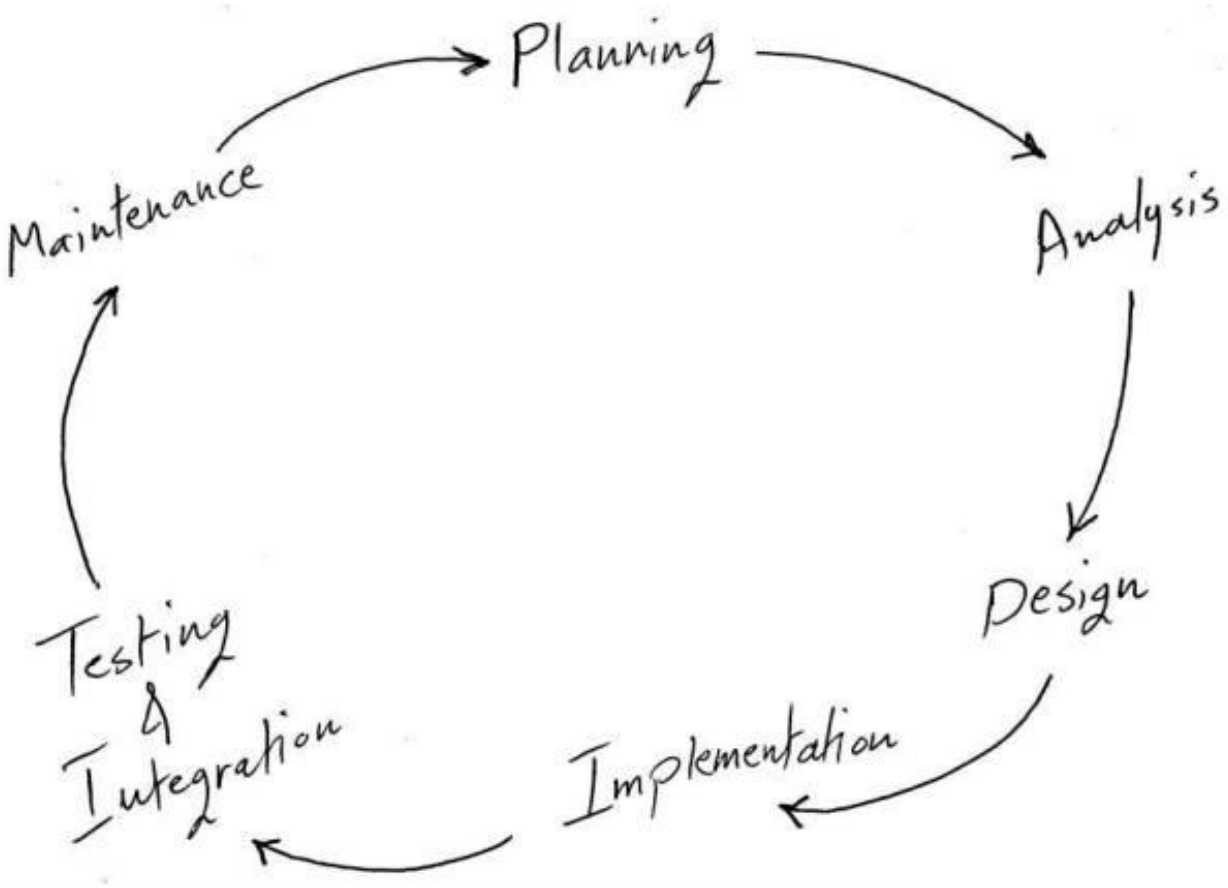
Question2:

Explain System Development Life Cycle; also explain different types system development life cycle.

Answer:

System Development Life Cycle: System Development Life Cycle (SDLC) is a series of six main phases to create a hardware system only, a software system only or a combination of both to meet or exceed customer's expectations.

System is a broad and a general term, and as per to Wikipedia; "A **system** is a set of interacting or interdependent components forming an integrated whole" it's a term that can be used in different industries, therefore Software Development Life Cycle is a limited term that explains the phases of creating a software component that integrates with other software components to create the whole system.
Below we'll take a general look on System Development Life Cycle phases, bearing in mind that each system is different from the other in terms of complexity, required components and expected solutions and functionalities:



Phases of SDLC:

1- System Planning

The Planning phase is the most crucial step in creating a successful system, during this phase you decide exactly what you want to do and the problems you're trying to solve, by:

- Defining the problems, the objectives and the resources such as personnel and costs.
- Studying the ability of proposing alternative solutions after meeting with clients, suppliers, consultants and employees.
- Studying how to make your product better than your competitors'.

After analyzing this data, you will have three choices: develop a new system, improve the current system or leave the system as it is.

2- System Analysis

The end-user's requirements should be determined and documented, what their expectations are for the system, and how it will perform. A feasibility study will be made for the project as well, involving determining whether it's organizationally, economically, socially, technologically feasible. it's very important to maintain strong communication level with the clients to make sure you have a clear vision of the finished product and its function.

3- System Design

The design phase comes after a good understanding of customer's requirements, this phase defines the elements of a system, the components, the security level, modules, architecture and the different interfaces and type of data that goes through the system.

A general system design can be done with a pen and a piece of paper to determine how the system will look like and how it will function, and then a detailed and expanded system design is produced, and it will meet all functional and technical requirements, logically and physically.

4- Implementation and Deployment

This phase comes after a complete understanding of system requirements and specifications, it's the actual construction process after having a complete and illustrated design for the requested system.

In the Software Development Life Cycle, the actual code is written here, and if the system contains hardware, then the implementation phase will contain configuration and fine-tuning for the hardware to meet certain requirements and functions.

In this phase, the system is ready to be deployed and installed in customer's premises, ready to become running, live and productive, training may be required for end users to make sure they know how to use the system and to get familiar with it, the implementation phase may take a long time and that depends on the complexity of the system and the solution it presents.

5- System Testing and Integration

Bringing different components and subsystems together to create the whole integrated system, and then Introducing the system to different inputs to obtain and analyze its outputs and behavior and the way it functions. Testing is becoming more and more important to ensure customer's satisfaction, and it requires no knowledge in coding, hardware configuration or design.

Testing can be performed by real users, or by a team of specialized personnel, it can also be systematic and automated to ensure that the actual outcomes are compared and equal to the predicted and desired outcomes

6- System Maintenance

In this phase, periodic maintenance for the system will be carried out to make sure that the system won't become obsolete, this will include replacing the old hardware and continuously evaluating system's performance, it also includes providing latest updates for certain components to make sure it meets the right standards and the latest technologies to face current security threats.

These are the main six phases of the System Development Life Cycle, and it's an iterative process for each project. It's important to mention that excellent communication level should be maintained with the customer, and Prototypes are very important and helpful when it comes to meeting the requirements. By building the system in short iterations; we can guarantee meeting the customer's requirements before we build the whole system.

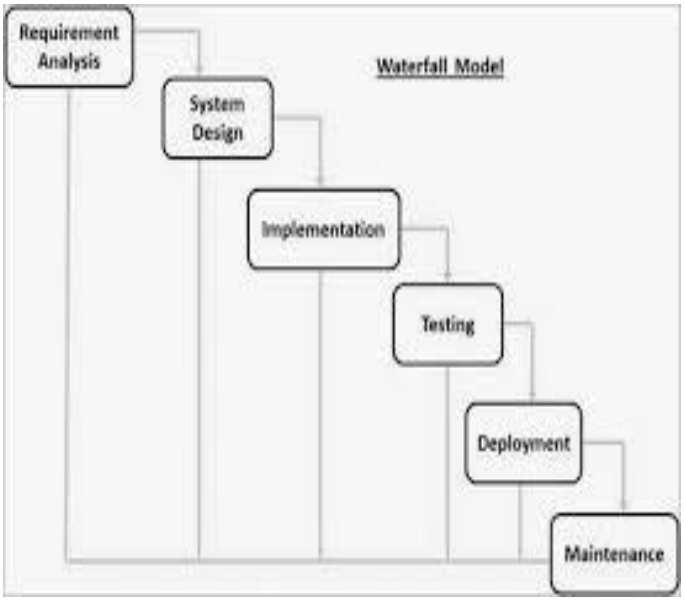
Many models of system development life cycle came up from the idea of saving effort, money and time, in addition to minimizing the risk of not meeting the customer's requirement at the end of project, some of these models are SDLC Iterative Model, and SDLC Agile Model.

Different types system development life cycle.

- **Waterfall model**
- **Iterative model**
- **Spiral model**
- **V-shaped model**
- **Agile model**

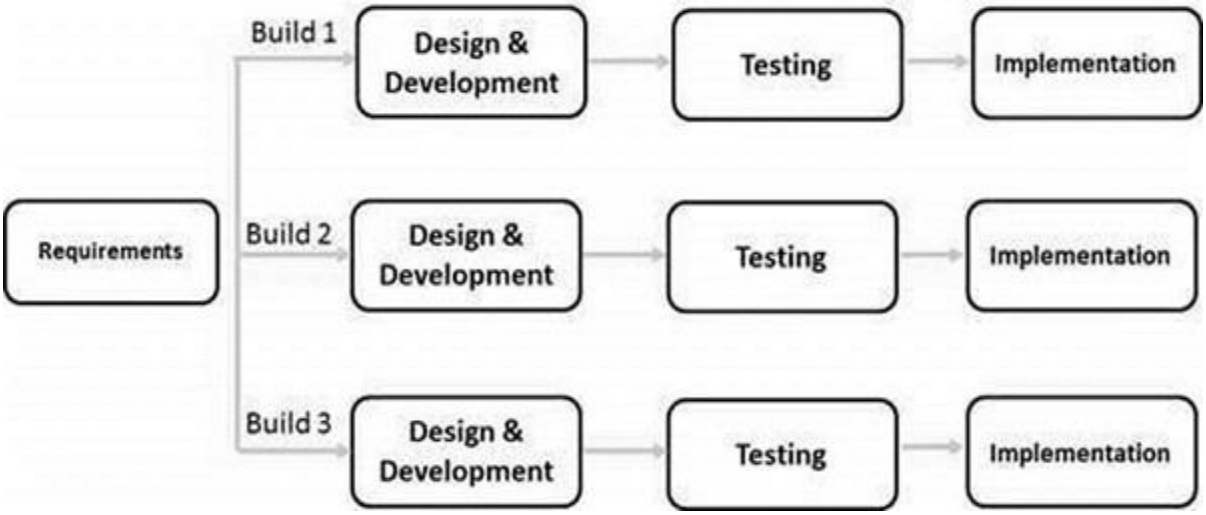
Waterfall Model:

The **Waterfall model** is the earliest SDLC approach that was used for software development.
The **waterfall Model** illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete.

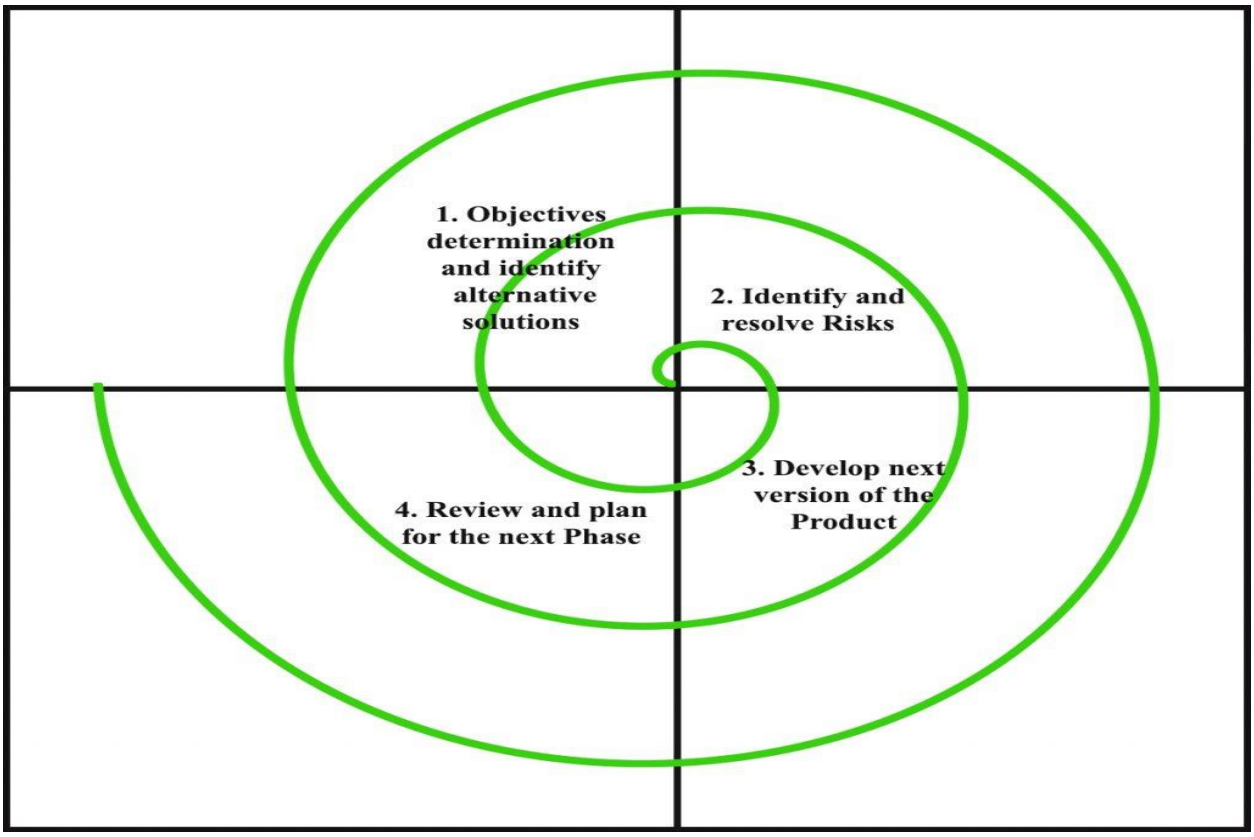


Iterative model

The **iterative model** is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete.

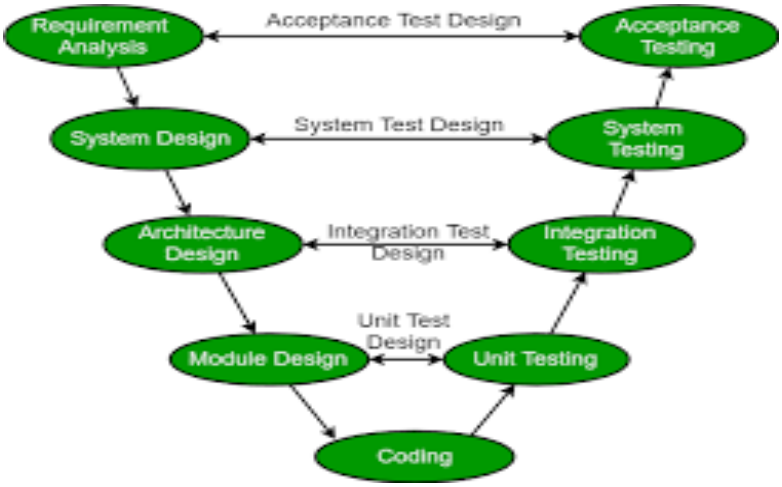


Spiral model: The **spiral model** is a systems development lifecycle (SDLC) method used for risk management that combines the iterative development process **model** with elements of the waterfall **model**. The **spiral model** is used by software engineers and is favored for large, expensive and complicated projects.



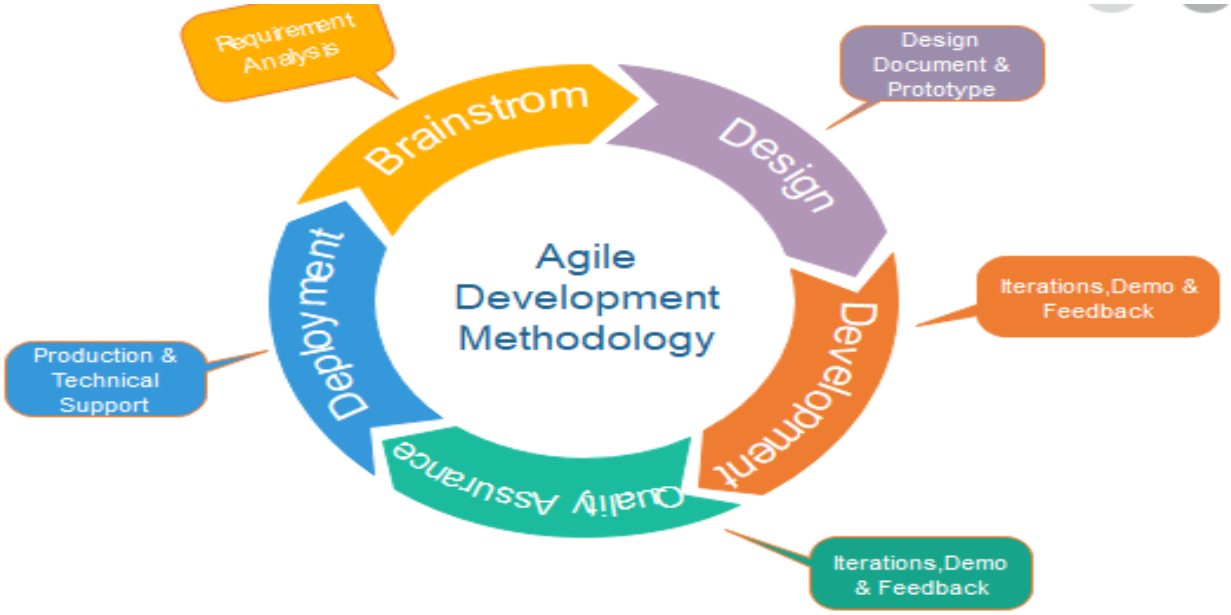
V-Model

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.**



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.

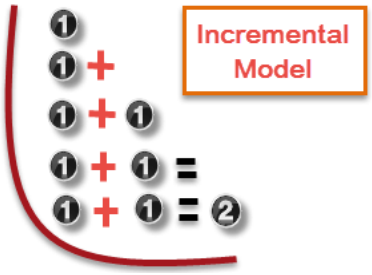


Question3:

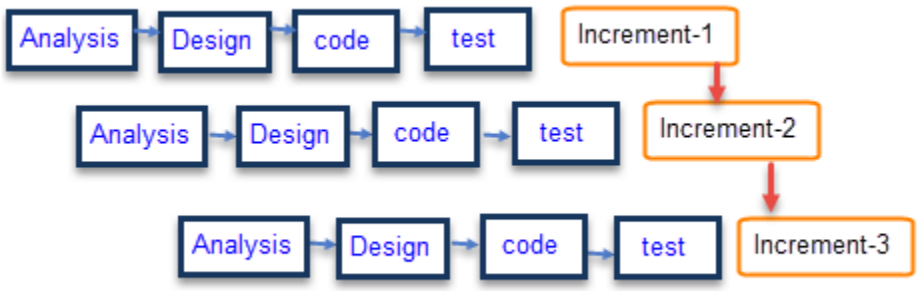
Explain Incremental model and Spiral; also explain main deference between spiral and incremental model.

Answer:

Incremental model: Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.



Each iteration passes through the **requirements, design, coding and testing phases**. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.



The system is put into production when the first increment is delivered. The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments. Once the core product is analyzed by the client, there is plan development for the next increment.

Characteristics of an Incremental module includes

- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first
- Once the requirement is developed, requirement for that increment are frozen

When to use Incremental models?

- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- 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.

Advantages:

- The software will be generated quickly during the software life cycle
- It is flexible and less expensive to change requirements and scope
- Throughout the development stages changes can be done.

Disadvantages:

- It requires a good planning designing.
- Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle.

- Each iteration phase is rigid and does not overlap each other.

Spiral Model: The spiral model is a systems development lifecycle (SDLC) method used for risk management that combines the iterative development process model with elements of the waterfall model. The spiral model is used by software engineers and is favored for large, expensive and complicated projects. When viewed as a diagram, the spiral model looks like a coil with many loops. The number of loops varies based on each project and is often designated by the project manager. Each loop of the spiral is a phase in the software development process. The spiral model enables gradual releases and refinement of a product through each phase of the spiral as well as the ability to build prototypes at each phase. The most important feature of the model is its ability to manage unknown risks after the project has commenced; creating a prototype makes this feasible.

Uses of the spiral model

As mentioned before, the spiral model is best used in large, expensive and complicated projects. Other uses include:

- projects in which frequent releases are necessary;
- projects in which changes may be required at any time;
- long term projects that are not feasible due to altered economic priorities;
- medium to high risk projects;
- projects in which cost and risk analysis is important;
- projects that would benefit from the creation of a prototype; and
- projects with unclear or complex requirements.

Spiral model phases

When looking at a diagram of a spiral model, the radius of the spiral represents the cost of the project and the angular degree represents the progress made in the current phase. Each phase begins with a goal for the design and ends when the developer or client reviews the progress. Every phase can be broken into four quadrants: identifying and understanding requirements, performing risk analysis, building the prototype and evaluation of the software's performance.

Phases begin in the quadrant dedicated to the identification and understanding of requirements. The overall goal of the phase should be determined and all objectives should be elaborated and analyzed. It is important to also identify alternative solutions in case the attempted version fails to perform.

Next, risk analysis should be performed on all possible solutions in order to find any faults or vulnerabilities -- such as running over the budget or areas within the software that could be open to cyber-attacks. Each risk should then be resolved using the most efficient strategy.

In the next quadrant, the prototype is built and tested. This step includes: architectural design, design of modules, physical product design and the final design. It takes the proposal that has been created in the first two quadrants and turns it into software that can be utilized.
Finally, in the fourth quadrant, the test results of the newest version are evaluated. This analysis allows programmers to stop and understand what worked and didn't work before progressing with a new build. At the end of this quadrant, planning for the next phase begins and the cycle repeats. At the end of the whole spiral, the software is finally deployed in its respective market.

Steps of the spiral model

While the phases are broken down into quadrants, each quadrant can be further broken down into the steps that occur within each one. The steps in the spiral model can be generalized as follows:

The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.

A preliminary design is created for the new system.

A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

A second prototype is evolved by a fourfold procedure: (1) evaluating the first prototype in terms of its strengths, weaknesses, and risks; (2) defining the requirements of the second prototype; (3) planning and designing the second prototype; (4) constructing and testing the second prototype.

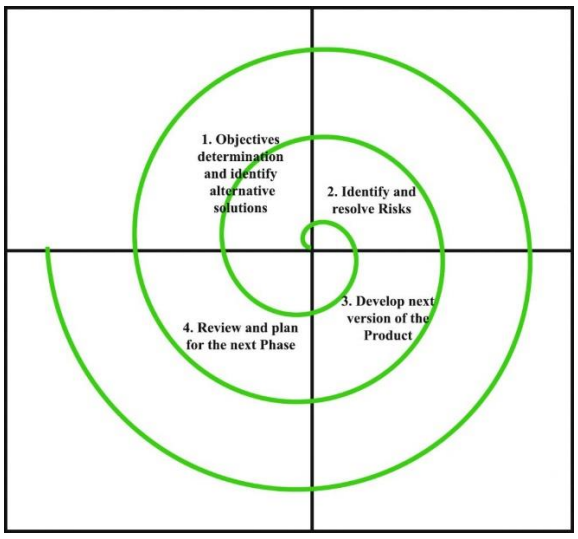
The entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation and other factors that could result in a less-than-satisfactory final product.

The existing prototype is evaluated in the same manner as was the previous prototype, and, if necessary, another prototype is developed from it according to the fourfold procedure outlined above.

The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.

The final system is constructed, based on the refined prototype.

The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.



Benefits of the spiral model

As mentioned before, the spiral model is a great option for large, complex projects. The progressive nature of the model allows developers to break a big project into smaller pieces and tackle one feature at a time, ensuring nothing is missed. Furthermore, since the prototype building is done progressively, the cost estimation of the whole project can sometimes be easier.
Other benefits of the spiral model include:
Flexibility - Changes made to the requirements after development has started can be easily adopted and incorporated.

Risk handling - The spiral model involves risk analysis and handling in every phase, improving security and the chances of avoiding attacks and breakages. The iterative development process also facilitates risk management.

Customer satisfaction - The spiral model facilitates customer feedback. If the software is being designed for a customer, then the customer will be able to see and evaluate their product in every phase. This allows them to voice dissatisfactions or make changes before the product is fully built, saving the development team time and money.

Limitations of the spiral model

Limitations of the spiral model include:

High cost - The spiral model is expensive and, therefore, is not suitable for small projects.

Dependence on risk analysis - Since successful completion of the project depends on effective risk handling, then it is necessary for involved personnel to have expertise in risk assessment.

Complexity - The spiral model is more complex than other SDLC options. For it to operate efficiently, protocols must be followed closely. Furthermore, there is increased documentation since the model involves intermediate phases.

Hard to manage time - Going into the project, the number of required phases is often unknown, making time management almost impossible. Therefore, there is always a risk for falling behind schedule or going over budget.

Main deference between spiral and incremental model:

A iterative model is a way to describe a SDLC as a sequence of consecutive steps.

A spiral model is a way to implement a iterative model, where each iteration follows a waterfall-like model. With each iteration, the product is updated, more features are added etc.

Properties of Model	Iterative/Incremental Model	Spiral Model
1. Planning in early stage	Yes	Yes
2. Returning to an earlier phase	Yes	Yes
3. Handle Large Project	Not Appropriate	Appropriate
4. Detailed Documentation	Not much	Yes
5. Cost	Low	Expensive
6. Requirement Specifications	Beginning	Beginning
7. User Involvement	Intermediate	High
8. Risk Involvement	Low	Medium-High
9. Testing	After every iteration	At the end of the engineering phase
10. Overlapping Phases	Yes(Parallel development exists)	No
11. Objective	Rapid development	High Assurance
12. Team size	Moderate size tea,	Large team