3-1) SDLC METHDOLOGIES:

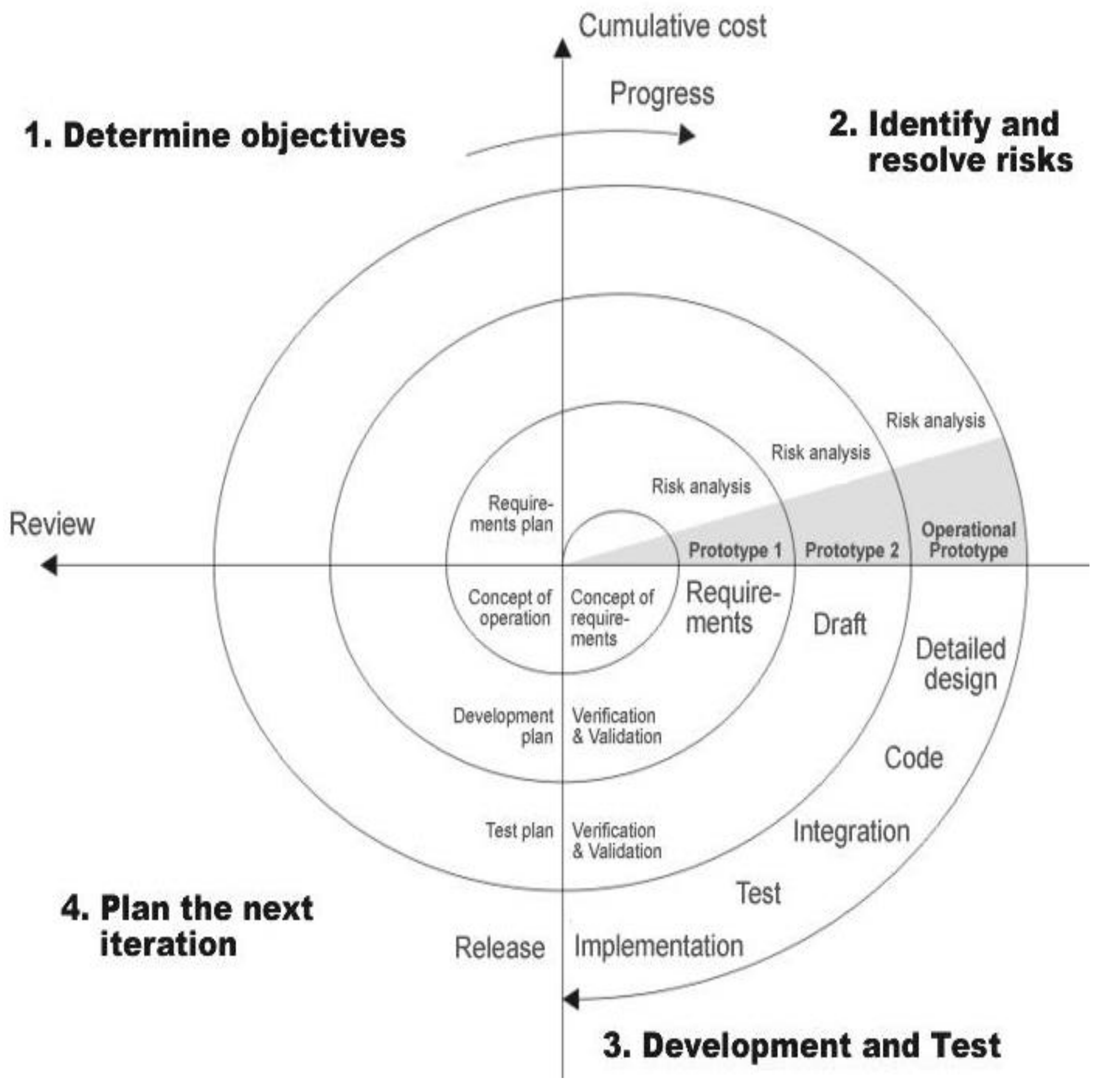
This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

Spiral Model is a combination of a waterfall model and iterative model. Each phase in spiral model begins with a design goal and ends with the client reviewing the progress. The spiral model was first mentioned by Barry Boehm in his 1986 paper.

The development team in Spiral-SDLC model starts with a small set of requirement and goes through each development phase for those set of requirements. The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase.

3-2) Spiral Model Phases:

Planning	• It includes estimating the cost, schedule and resources for the iteration. It also involves understanding the system requirements for continuous communication between the system analyst and the customer
Risk Analysis	• Identification of potential risk is done while risk mitigation strategy is planned and finalized
Engineering	• It includes testing, coding and deploying software at the customer site
Evaluation	• Evaluation of software by the customer. Also, includes identifying and monitoring risks such as schedule slippage and cost overrun



3-3) Reasons to Use 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.
- Project estimates in terms of schedule, cost etc become more and more realistic as the project moves forward and loops in spiral get completed.
- It is suitable for high risk projects, where business needs may be unstable.
  A highly customized product can be developed using this.

3-4) Software Requirements:

• Operating System	:	Windows
• Graphical User Interface	:	Java Swing, AWT.
• Application Logic	:	Java SE 11.0.2.
• Database	:	Oracle 10c.
• IDE/Workbench	:	My Eclipse 2017.

3-5) Hardware Requirements:

• Processor	:	Pentium III – 900 MHz
• Hard Disk	:	20 GB
• RAM	:	128 MB

3-6) Type of Design:

Object–Oriented Design (OOD) involves implementation of the conceptual model produced during object-oriented analysis. In OOD, concepts in the analysis model, which are technology–independent, are mapped onto implementing classes, constraints are identified and interfaces are designed, resulting in a model for the solution domain, a detailed description of how the system is to be built on concrete technologies.

The implementation details generally include:

- Restructuring the class data.
- Implementation of methods, internal data structures and algorithms.
- Implementation of control.
- Implementation of associations.

Grady Booch has defined object-oriented design as “a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design”.

A key goal of the object-oriented approach is to decrease the "semantic gap" between the system and the real world, and to have the system be constructed using terminology that is almost the same as the stakeholders use in everyday business. Object-oriented modeling is an essential tool to facilitate this.

Object: An object is a real-world element in an object–oriented environment that may have a physical or a conceptual existence. Each object has:

- Identity that distinguishes it from other objects in the system.
- State that determines the characteristic properties of an object as well as the values of the properties that the object holds.
- Behavior that represents externally visible activities performed by an object in terms of changes in its state.

Objects can be modeled according to the needs of the application. An object may have a physical existence, like a customer, a car, etc.; or an intangible conceptual existence, like a project, a process, etc.

Class: A class represents a collection of objects having same characteristic properties that exhibit common behavior. It gives the blueprint or description of the objects that can be created from it. Creation of an object as a member of a class is called instantiation. Thus, object is an instance of a class. The constituents of a class are:

- A set of attributes for the objects that are to be instantiated from the class.
- Generally, different objects of a class have some difference in the values of the attributes. Attributes are often referred as class data.
- A set of operations that portray the behavior of the objects of the class. Operations are also referred as functions or methods.

There are two Properties of Object orientation:

1) **Abstraction:** Abstraction allows us to represent complex real world in simplest manner. It is the process of identifying the relevant qualities and behaviors an object should possess; in other words, to represent the necessary feature without representing the background details.
2) **Encapsulation:** Encapsulation is the process of binding both attributes and methods together within a class. Through encapsulation, the internal details of a class can be hidden from outside. It permits the elements of the class to be accessed from outside only through the interface provided by the class.