INTRODUCTION TO SOFTWARE ENGINEERING AND SOFTWARE LIFECYCLE MODELS

Prof. Sudip Misra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur
http://cse.iitkgp.ac.in/~smisra/

What is Software?

- **Software** is more than just a program code.
- A **program** is an executable code, which serves some computational purpose.
- **Software** is considered to be collection of executable programming code, associated libraries and documentations.

What is Engineering?

- **Engineering** is all about developing products using well-defined, scientific principles and methods.

What is Software Engineering?

- **IEEE defines software engineering as:**

The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.

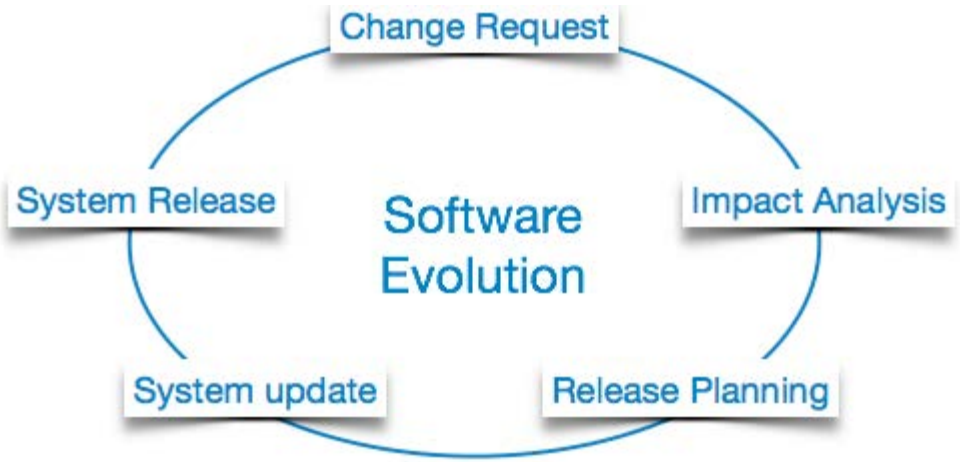
WHY STUDY SOFTWARE ENGINEERING?

To acquire the following skills :

- The skill to participate in development of large software products.
- To learn how to effectively handle complexity in a software development problem.

Software Evolution

- The process of developing a software product using software engineering principles and methods is referred to as **software evolution.**



Need of Software Engineering

- Large software
- Scalability
- Cost
- Dynamic Nature
- Quality Management

Characteristics of good software

- Operational
- Transitional
- Maintenance

Characteristics : Operational

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security

Characteristics : Transitional

- Portability
- Interoperability
- Reusability
- Adaptability

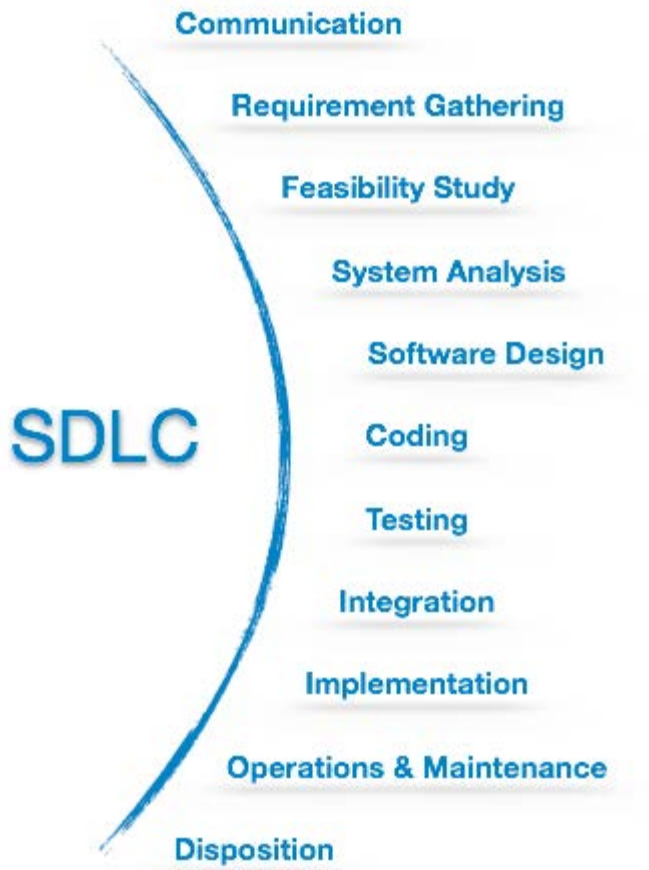
Characteristics : Maintenance

- Modularity
- Maintainability
- Flexibility
- Scalability

Software development life cycle (SDLC)

- The **software development life cycle** (**SDLC**) is a framework defining tasks performed at each step in the software development process.

Steps of SDLC



Software life cycle model

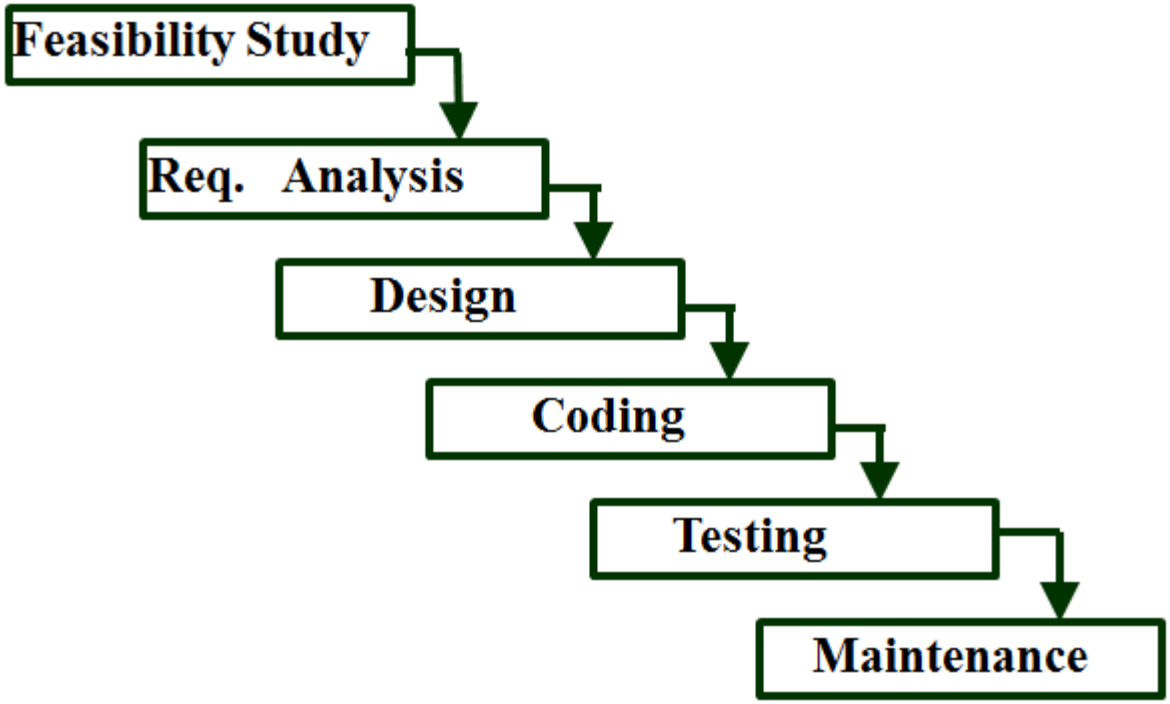
- A descriptive and diagrammatic model of software lifecycle.
- Identifies all the activities required for product development.
- Establishes a precedence ordering among the different activities.
- Divides lifecycle into phases.

Classical Waterfall Model

Classical waterfall model divides life cycle into phases:

- Feasibility study
- Requirements analysis and specification
- Design
- Coding and unit testing
- Integration and system testing
- Maintenance

Classical Waterfall Model



Feasibility Study

Main aim of feasibility study :

- To determine whether developing the product financially worthwhile and technically feasible.

The collected data are analyzed to arrive at the following :

- An abstract problem definition.
- Formulation of the different strategies for solving the problem.
- Evaluation of the different solution strategies.

Requirement analysis and specification

Aim of this phase:

- To understand the exact requirements of the customer.
- To document them properly.

Consists of two distinct activities:

- Requirements gathering and analysis.
- Requirements specification.

Requirements gathering and analysis

Gathering relevant data:

- To collect all relevant information regarding the product to be developed from the customer with a view to clearly understand the customer requirements.

Requirements analysis :

- To weed out the incompleteness and inconsistencies in these requirements.

Requirements Specification

Aim of this phase:

- To construct the Software Requirements specification (SRS) document.
- Contents of SRS are :
  - The functional requirements.
  - The non-functional requirements.
  - The goals of implementations.

DESIGN

- Design phase transforms requirements specification into a form suitable for implementation in some programming language.
- **Approaches :**
  - Traditional design approach
  - Object-oriented design approach

DESIGN : TRADITIONAL APPROACH

Consists of two activities:

- **Structured analysis :** The detailed structure of the problem is examined.

- **Structured design :** The results of structured analysis are transformed into the software design.

Design : Structured Design

High-level design:

- Decompose the system into modules
- Represent invocation relationships among the modules.

Detailed design:

- Different modules designed in greater detail.
- Data structures and algorithms for each module are designed.

DESIGN : OBJECT-ORIENTED APPROACH

First identify various objects (real-world entities) occurring in the problem:

- Identify the relationships among the objects.
- For example, the objects in a pay-roll software may be:
  - Employees
  - Managers
  - Pay-roll register
  - Departments, etc.

CODING & UNIT TESTING

Aim of this phase :

- To translate the software design into source code.

Coding :

- The design is implemented as a program module.

Unit testing:

- Determines the correct working of the individual modules.

INTEGRATION AND SYSTEM TESTING

Aim of this phase :

- To integrate different modules in a planned manner.

Integration Testing :

- Various modules are integrated incrementally over a number of steps.

System Testing :

- α-testing
- β-testing
- Acceptance testing

MAINTENANCE

Aim of this phase :

- Corrective
- Perfective
- Adaptive

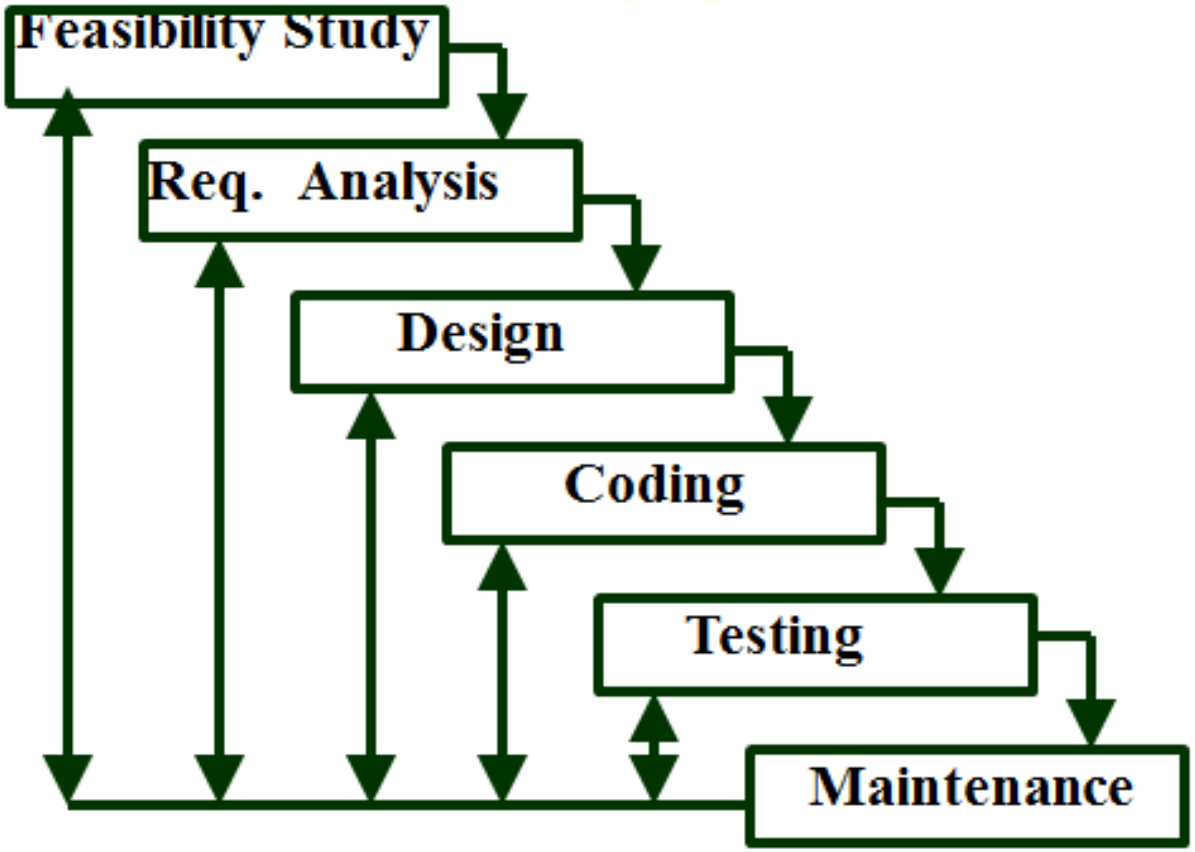
Limitation of classical waterfall model

- More ideal than practical.
- Assumes that all requirements are correct at the beginning.
- Freezing requirement at the beginning means freezing hardware which may become obsolete in due course of a large project.
- Assumes that all the phases are sequential.
- The model is heavily document driven as formal document is required after end of each phase.

Iterative waterfall model

- Provides feedback path in the classical waterfall model

ITERATIVE WATERFALL MODEL



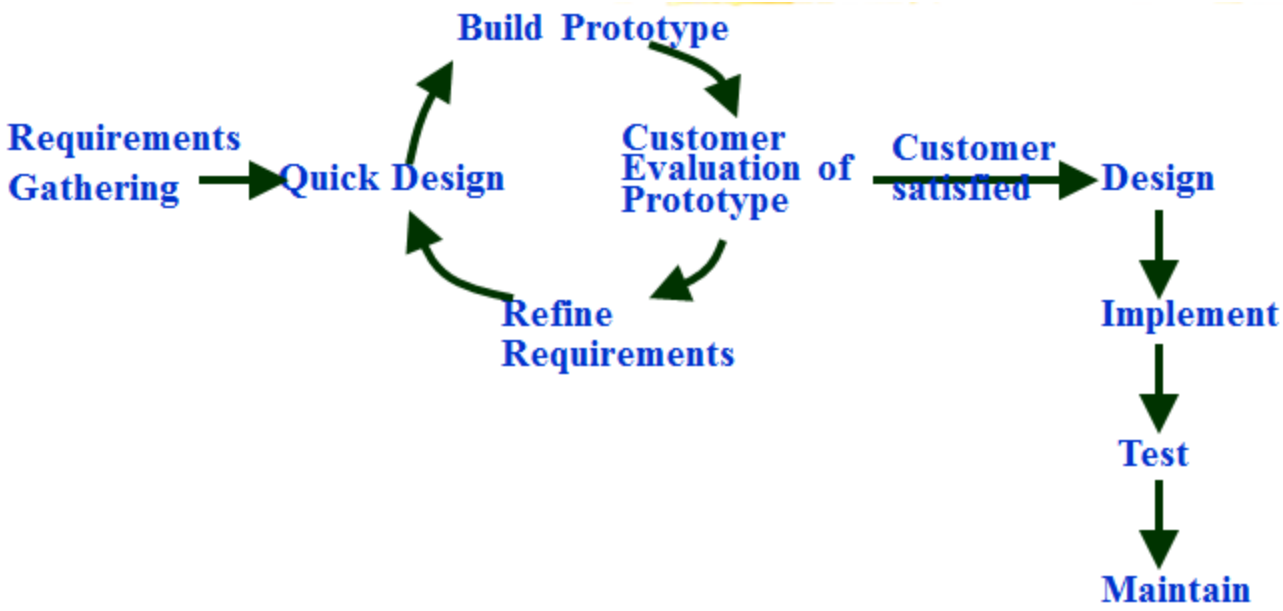
Limitations of Iterative waterfall model

- Cannot handle different types of risks that a real-life software project may suffer from.
- Rigid phase sequence.

PROTOTYPING MODEL

- Before starting actual development a working prototype of the system should first be built.
- **A prototype is a toy implementation of a system:**
  - limited functional capabilities
  - low reliability
  - inefficient performance

Prototyping model



Benefits of Prototyping model

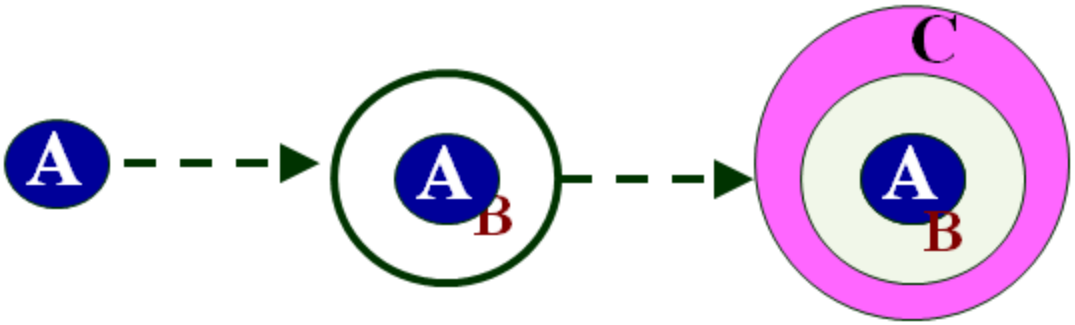
- Misunderstanding between software developer and user may be identified.
- Missing user services may be detected.
- Difficult to use or confusing user services may be identified and refined.
- It helps in gaining user confidence.
- It helps in writing the specification.

EVOLUTIONARY OR INCREMENTAL MODEL

Evolutionary model (incremental model):

- The system is broken down into several modules which can be incrementally implemented and delivered.
- First develop the core modules of the system.
- The initial product skeleton is refined into increasing levels of capability:
  - by adding new functionalities in successive versions.

EVOLUTIONARY OR INCREMENTAL MODEL



A,B,C are modules of a software product that are incrementally developed and delivered.

Advantages of Evolutionary Model

Users get a chance to experiment with a partially developed system:

- Much before the full working version is released.

Helps finding exact user requirements:

- Much before fully working system is developed.

Core modules get tested thoroughly:

- Reduces chances of errors in final product.

DISADVANTAGES OF EVOLUTIONARY MODEL

Often, difficult to subdivide problems into functional units:

- Can be incrementally implemented and delivered.
- Useful for very large problems.
- Easier to find modules for incremental implementation.

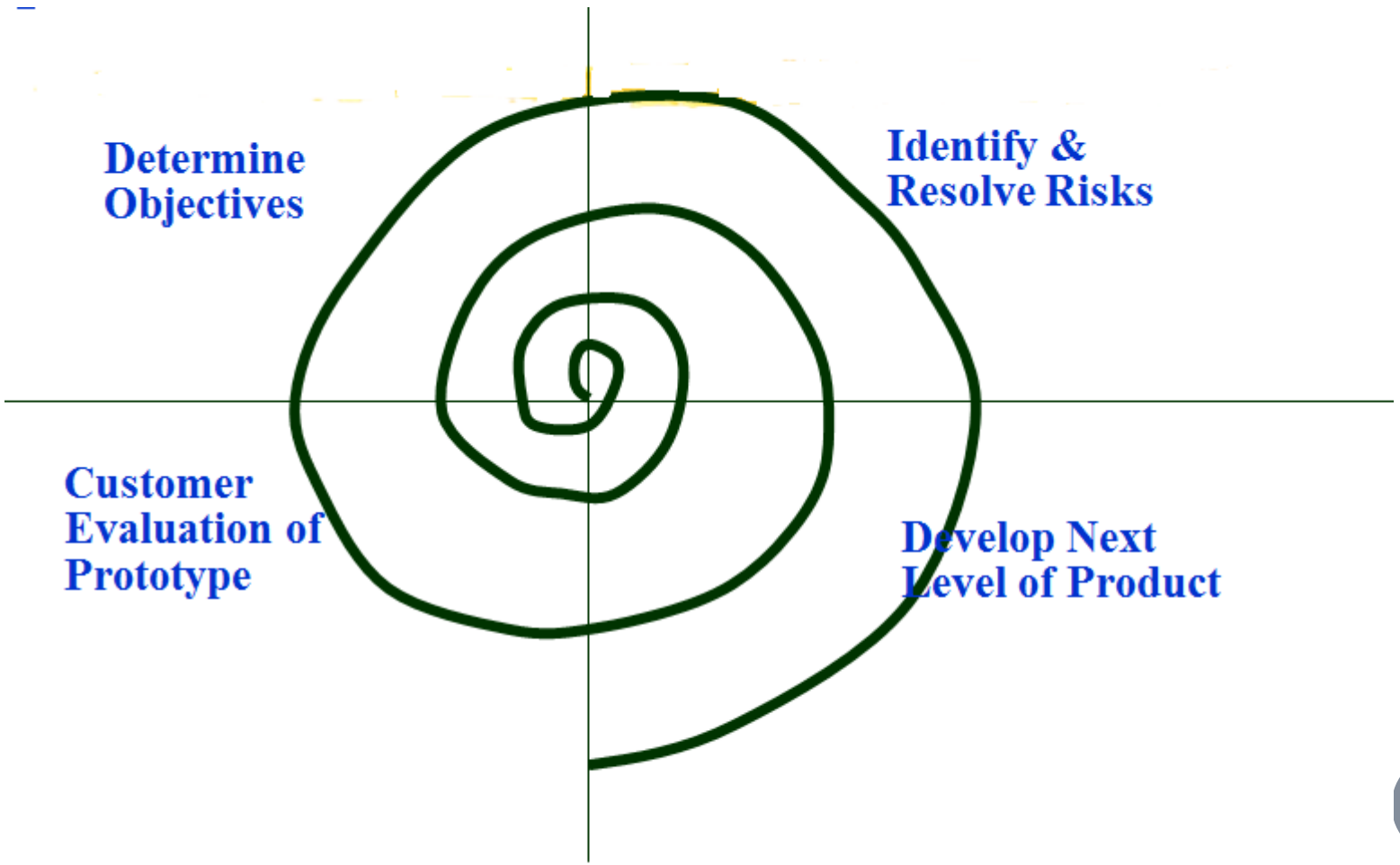
Spiral Model

- Proposed by Boehm in 1988.
- Each loop of the spiral represents a phase of the software process:
  - the innermost loop might be concerned with system feasibility,
  - the next loop with system requirements definition,
  - the next one with system design, and so on.
- There are no fixed phases in this model, the phases shown in the figure are just examples.

SPIRAL MODEL CONT.

- The team must decide how to structure the project into phases.
- Start work using some generic model.
- Add extra phases for specific projects or when problems are identified during a project.
- Each loop in the spiral is split into four sectors (quadrants).

Spiral Model



Objective Setting (First Quadrant)

- Identify objectives of the phase,
- Examine the risks associated with these objectives.
- **Risk:**
  - any adverse circumstance that might hamper successful completion of a software project.
- Find alternate solutions possible.

Risk Assessment and Reduction (Second Quadrant)

- For each identified project risk, a detailed analysis is carried out.
- Steps are taken to reduce the risk.
- For example, if there is a risk that the requirements are inappropriate then a prototype system may be developed.

Development and Validation (Third quadrant):

- Develop and validate the next level of the product.

Review and Planning (Fourth quadrant):

- Review the results achieved so far with the customer and plan the next iteration around the spiral.

Advantages of Spiral Model

- Spiral Life Cycle Model is one of the most flexible SDLC models in place.
- Project monitoring is very easy and effective.
- Risk management is one of the in-built features of the model.
- Changes can be introduced later in the life cycle as well.
- It is suitable for high risk projects, where business needs may be unstable.

Disadvantages of Spiral Model

- Cost involved in this model is usually high.
- It is a complicated approach especially for projects with a clear SRS.
- Skills required, to evaluate and review project from time to time, need expertise.
- Rules and protocols should be followed properly to effectively implement this model.
- Due to various customizations allowed from the client, using the same prototype in other projects, in future, is difficult.

Comparison of Different Life Cycle Models

Iterative waterfall model

- Most widely used model.
- But, suitable only for well-understood problems.

Prototype model is suitable for projects not well understood:

- User requirements
- Technical aspects

Comparison of Different Life Cycle Models

Evolutionary model is suitable for large problems:

- Can be decomposed into a set of modules that can be incrementally implemented,
- Incremental delivery of the system is acceptable to the customer.

The spiral model:

- Suitable for development of technically challenging software products that are subject to several kinds of risks.

THANK YOU