FOUNDATIONS OF SOFTWARE DEVELOPMENT

Priti Srinivas Sajja

- Name: **Dr. Priti Srinivas Sajja**
- Communication:
  - Email : priti@pritisajja.info
  - Mobile : +91 9824926020
  - URL :**http://pritisajja.info**

- *Academic qualifications* : Ph. D in Computer Science
- *Thesis title*: Knowledge-Based Systems for Socio-
- Economic Rural Development (2000)
- *Subject area of specialization* : Artificial Intelligence
- *Publications* : **216** in Books, Book Chapters, Journals and in Proceedings of International and National Conferences

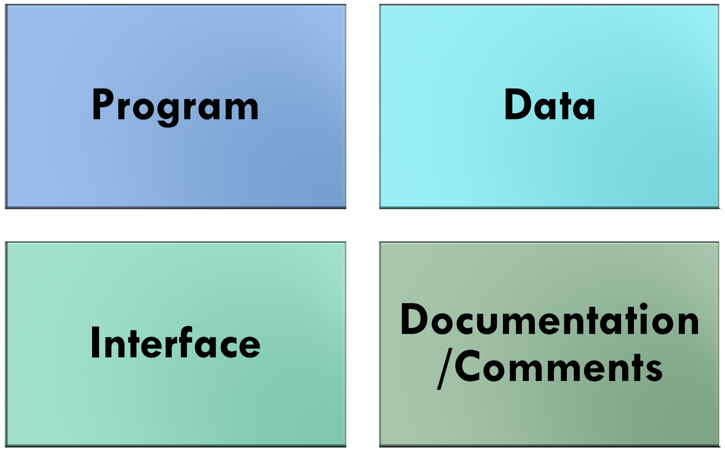
PGDCA I SEMESTER
PS01CDCA35
UNIT 4:SOFTWARE ENGINEERING

- **Software – meaning, general characteristics and applications**
- **Software Engineering – meaning, goal and needs**
- **Software Development Process Models – Waterfall, Iterative, Spiral, etc.**
- **Software Testing – introduction, needs and levels**

SOFTWARE

- Software is a **set of instructions and data** used to operate computers and execute specific tasks.
- Software contains
  - Data
  - Data Structure
  - Comments
  - Other instructions
- Opposite of hardware, which describes the physical aspects of a computer, software is a generic term used to refer to applications, scripts and programs that run on a device.

PROGRAM VS. SOFTWARE



- Software is more than programs. Any program is a subset of software.
- Software also contains documentation & operating procedures manuals.

CHARACTERISTICS OF SOFTWARE

- **Software does not wear out:**
  Different things like clothes, shoes, ornaments do wear out after some time. But, software once created **never wears out**. It can be used for as long as needed and in case of need for any updating, required changes can be made in the same software and then it can be used further with updated features.

- **Software is not manufactured:**
  Software is not manufactured **but is developed**. So, it does not require any raw material for its development.

CHARACTERISTICS OF SOFTWARE

- **Usability of Software:**
  The usability of the software is the **simplicity of the software** in terms of the user. The easier the software is to use for the user, the more is the usability of the software as more number of people will now be able to use it and also due to the ease will use it more willingly.

- **Reusability of components:**
  As the software never wears out, neither do its components, i.e. code segments. So, if any particular segment of code is required in some other software, we can **reuse the existing code** form the software in which it is already present. This reduced our work and also **saves time and money**.

CHARACTERISTICS OF SOFTWARE

- **Flexibility of software:**
  A software is flexible. What this means is that we can **make necessary changes** in the software in the future according to the need of that time and then can use the same software then also.

- **Maintainability of software:**
  Every software is maintainable. This means that if **any errors or bugs** appear in the software, then they **can be fixed**.

CHARACTERISTICS OF SOFTWARE

- **Portability of software:**
  Portability of the software means that we can transfer our software from **one platform to another** that too with ease. Due to this, the sharing of the software among the developers and other members can be done flexibly.

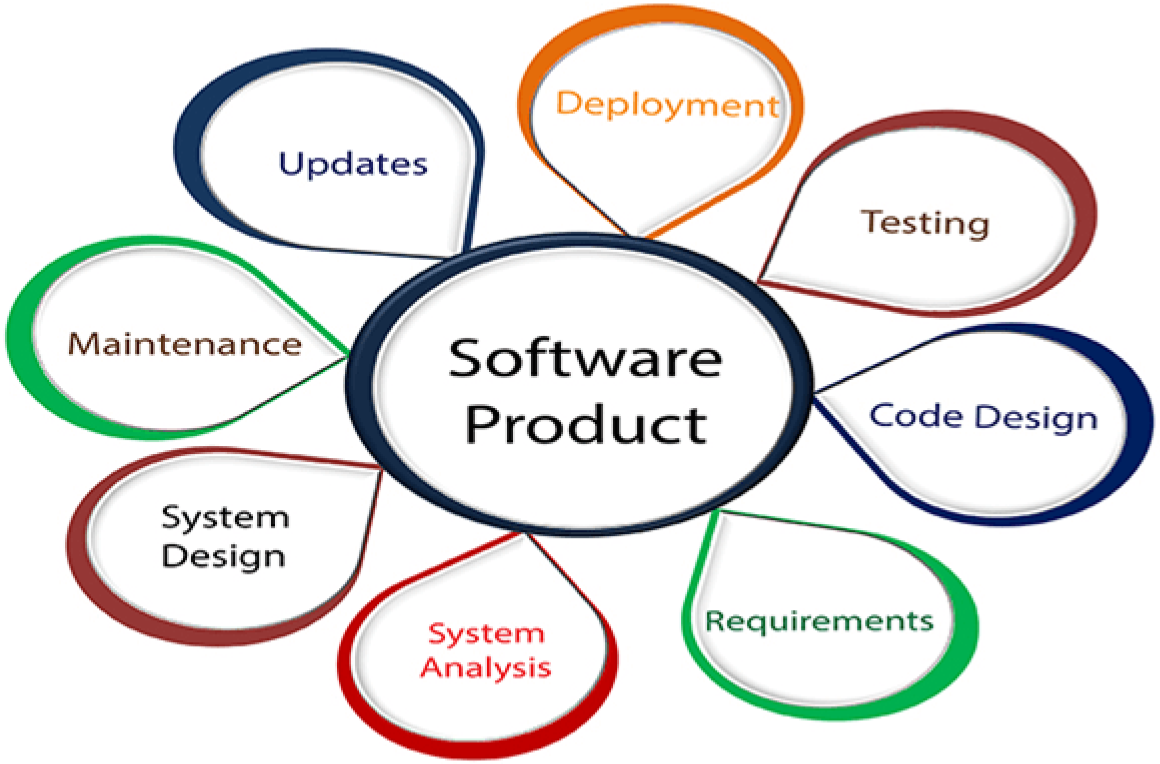
- **Reliability of Software:**
  This is the ability of the software to provide the **desired functionalities under every condition**. This means that our software should work properly in each condition.

TYPES OF SOFTWARE

- **System Software:** A system software aids the user and the hardware to function and interact with each other. Eg. OS, Device Drivers and utility software such as translators.
- **Applications Software:** It is a program or group of programs designed for end-users. It can be generic or domain specific. Eg. Attendance System.
- **Firmware:** Firmware is the permanent software that is embedded into a read-only memory. It is a set of instructions permanently stored on a hardware device.

MEANING: SOFTWARE ENGINEERING

- The term **software engineering** is the product of two words, **software**, and **engineering**.
- As stated software contains documentation & operating procedures manuals.
- Engineering is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.



Software Engineering is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.

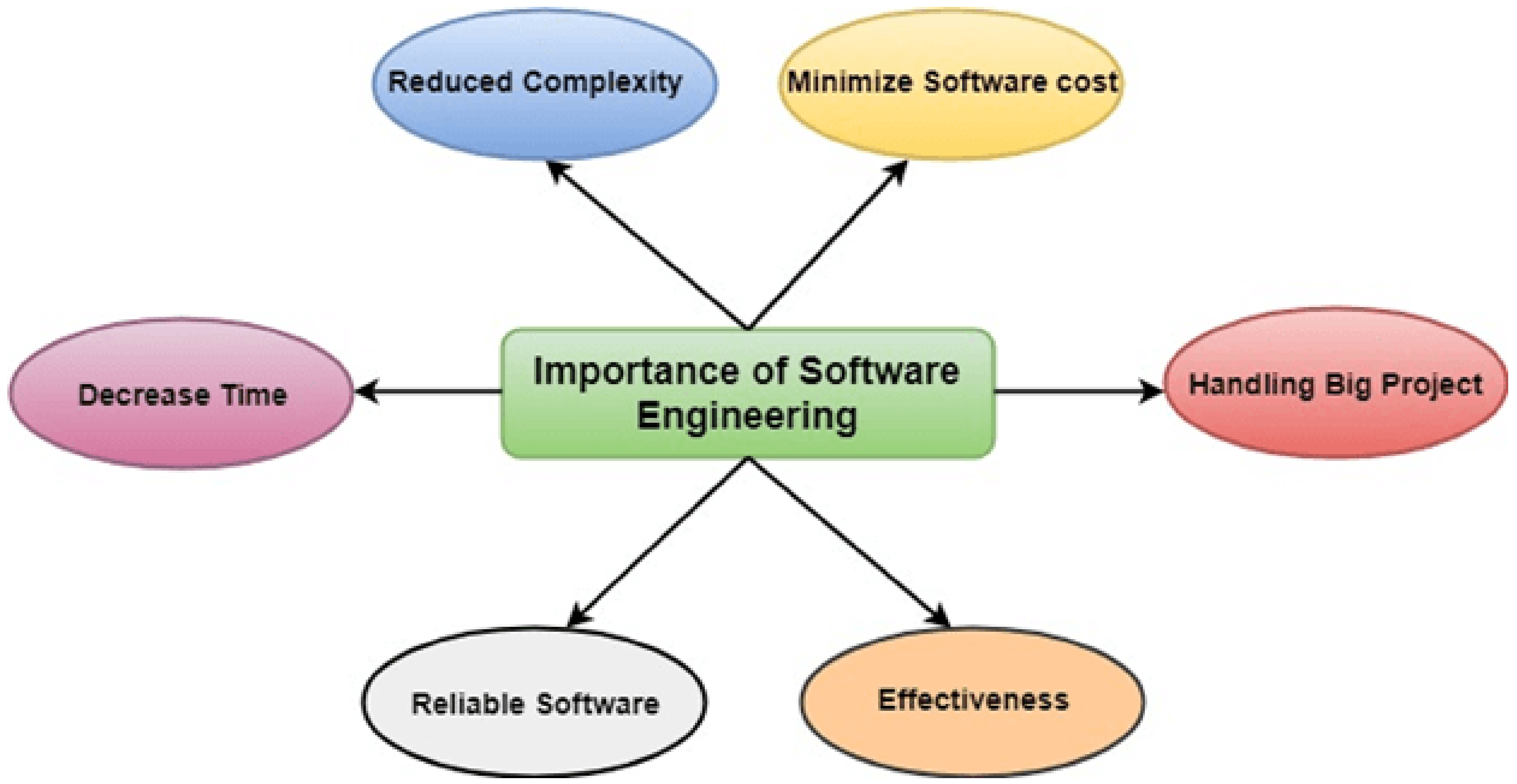
GOALS: WHY IS SOFTWARE ENGINEERING REQUIRED?

Software Engineering is required due to the following reasons:

- To manage Large software
- For more Scalability, Effectiveness and Reliability
- Cost Management
- To manage the dynamic nature of software
- For better quality Management

NEED OF SOFTWARE ENGINEERING

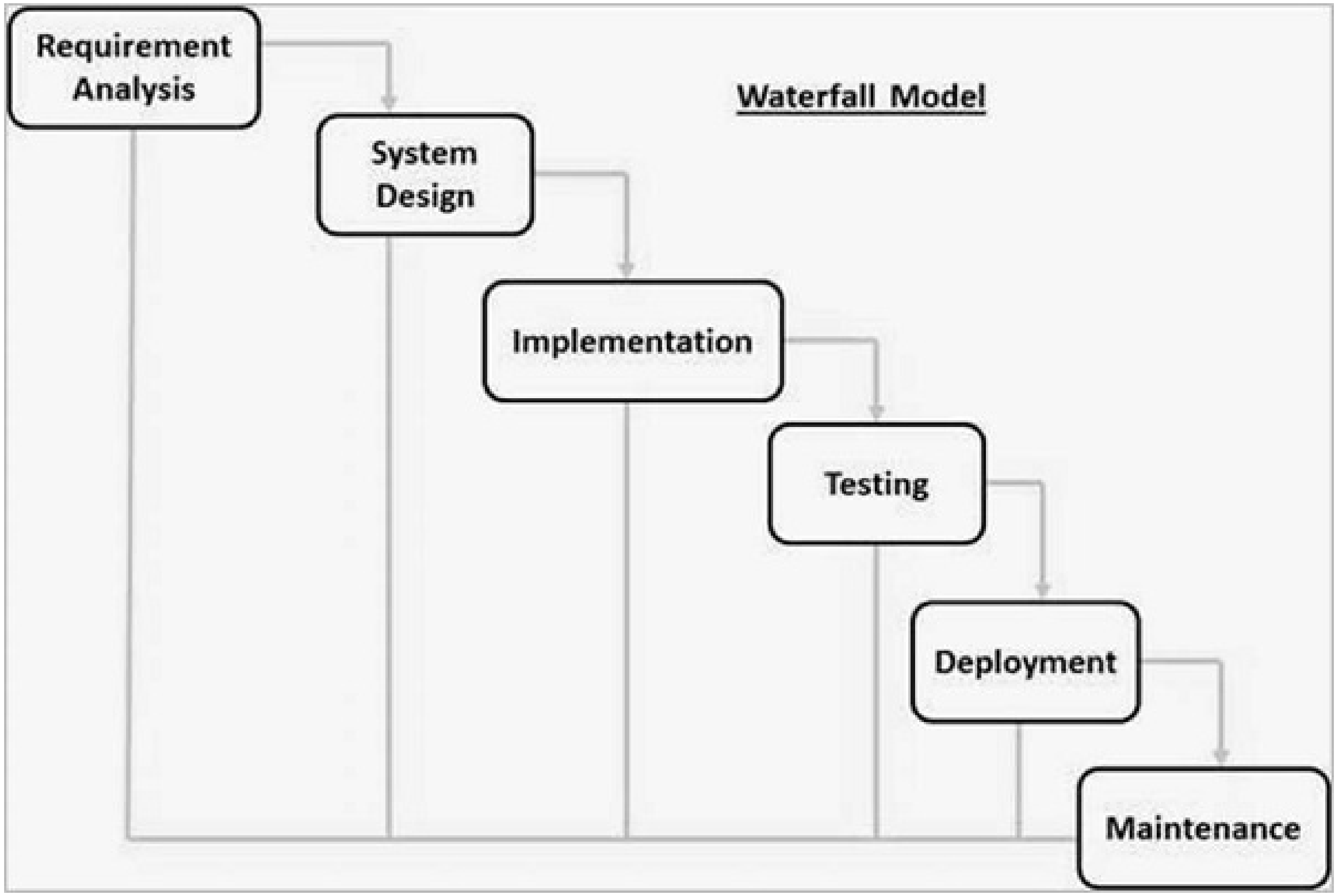
- **Increased expectations** and higher/dynamic requirements from users.
- **Huge Programming and big systems require a systematic approach:** It is simpler to manufacture a wall than to a house or building, similarly, as the measure of programming become extensive engineering has to step to give it a scientific process.
- **Adaptability:** If the software procedure were not based on scientific and engineering ideas, it would be simpler to re-create new software than to scale an existing one.
- **Cost:** Cost of software development can be reduced.
- **Quality Management:** Better procedure of software development provides a better and quality software product.


Reduced Complexity
Minimize Software cost
Decrease Time
Importance of Software Engineering
Handling Big Project
Reliable Software
Effectiveness

SOFTWARE DEVELOPMENT PROCESS MODELS

- Waterfall
- Iterative
- Spiral

WATERFALL MODEL



WATERFALL MODEL

- The Waterfall model is based on SDLC /Linear/ Classical approach that was used for software development.
- **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 as well as detailed designs.
- **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.

WATERFALL MODEL

- **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 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.
  - **Corrective, perfective, and adaptive maintenance**

WATERFALL MODEL: ADVANTAGES

The major advantages of the Waterfall Model are as follows −

- **Simple and easy to understand, use, and manage.**
- **Well understood milestones.**
- **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 clear.**
- **Clearly defined stages.**
- **Process and results are well documented.**

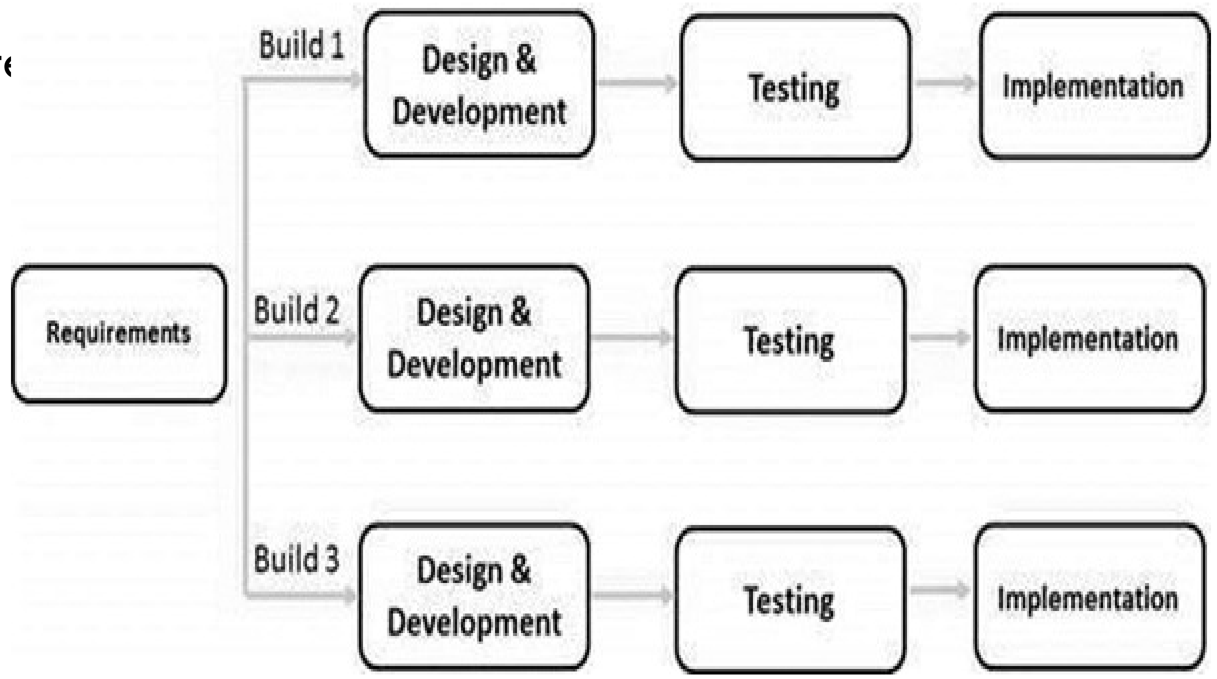
WATERFALL MODEL: DISADVANTAGES

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Cannot accommodate changing requirements.

ITERATIVE MODEL

- Iterative process starts with a **simple implementation of a subset of the software requirements** and **iteratively enhances** the evolving versions until the full system is implemented.
- Its ite



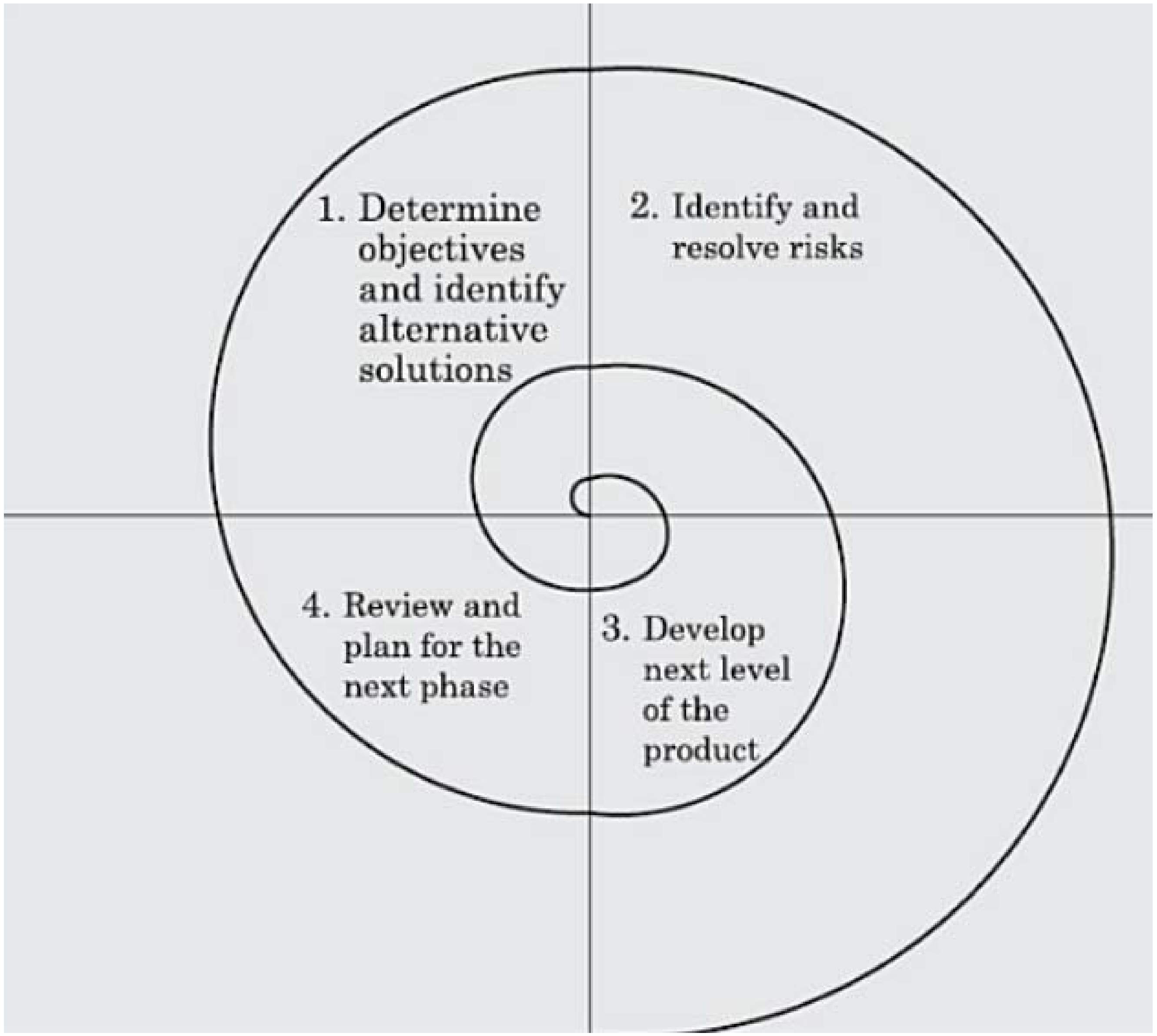
ITERATIVE MODEL: ADVANTAGES

- **Some working functionality can be developed quickly and early** in the life cycle. So, some results are obtained early and periodically. **With every increment, operational product is delivered.**
- **Parallel development** can be planned.
- **Less costly to change** the scope/requirements. It supports changing requirements.
- **Testing** and debugging during **smaller iteration** is easy.
- **Easier to manage risk -** High risk part is done first.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- 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.

ITERATIVE MODEL: DISADVANTAGES

- **More resources may** be required.
- Although cost of **change is lesser, but it is not very suitable** for changing requirements.
- Management complexity is more. 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.
- Not suitable for smaller projects.
- **End of project may not be known** which is a risk.
- Projects progress is highly dependent upon the risk analysis phase.

SPIRAL MODEL


1. Determine objectives and identify alternative solutions
2. Identify and resolve risks
3. Develop next level of the product
4. Review and plan for the next phase

SPIRAL MODEL

- **Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**.
- In its diagrammatic representation, it looks like a spiral with many loops.
- The exact number of loops of the spiral is unknown and can vary from project to project.
- **Each loop of the spiral is called a Phase of the software development process.**
- The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

SPIRAL MODEL

The spiral model has four phases.

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

SPIRAL MODEL

Phase 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 is identified and the risks are resolved using the best possible strategy.
- At the end of this quadrant, Prototype is built for the best possible solution.

SPIRAL MODEL

Phase 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.

Phase 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.

SPIRAL MODEL

Risk Handling in Spiral Model

- A risk is any adverse situation that might affect the successful completion of a software project.
- The most important feature of the spiral model is handling these unknown risks after the project has started.
- Such risk resolutions are easier done by developing a prototype.
- The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development.

SOFTWARE TESTING

- Testing is the process of evaluating a system or its component(s) with the intent to find **whether it satisfies the specified requirements or not.**
- Testing is executing a system in order **to identify any gaps, errors, or missing requirements** in contrary to the actual requirements.
- Benefits are
  - Cost-effectiveness
  - Security
  - Quality
  - Users satisfaction

SOFTWARE TESTING

Various types/strategies are :

- Unit Testing
- Integration Testing
- System Testing
- Special Systems Tests
- Black Box Testing
- Glass Box Testing

UNIT TESTING

- **UNIT TESTING** is a type of software **testing** where individual units or components of a software are tested.
- The purpose is to validate that each **unit** of the software code performs as expected.
- **Unit Testing** is done generally during the development (coding phase) of an application by the developers.
- Done by developer, tester and end users
- It can be white box or black box testing

UNIT TESTING

- **UNIT TESTING** is a type of software **testing** where individual units or components of a software are tested.
- The purpose is to validate that each **unit** of the software code performs as expected.
- **Unit Testing** is done generally during the development (coding phase) of an application by the developers.
- Done by developer, tester and end users
- It can be white box or black box testing

INTEGRATED TESTING

- **INTEGRATION TESTING** is defined as a type of testing where software modules are integrated logically and tested as a group.
- A typical software project consists of multiple software modules, coded by different programmers.
- The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

SYSTEM TESTING

- **SYSTEM TESTING** is a level of **testing** that validates the complete and fully integrated software product.
- The purpose of a **system test** is to evaluate the end-to-end **system** specifications.
- Acceptance testing
- Verification (Static) and Validation (Dynamic)
- Special systems tests
- Functional (actual working) and non-functional testing (performance, load, scalability, security, etc.)