Software Engineering:-Unit:-1

Software

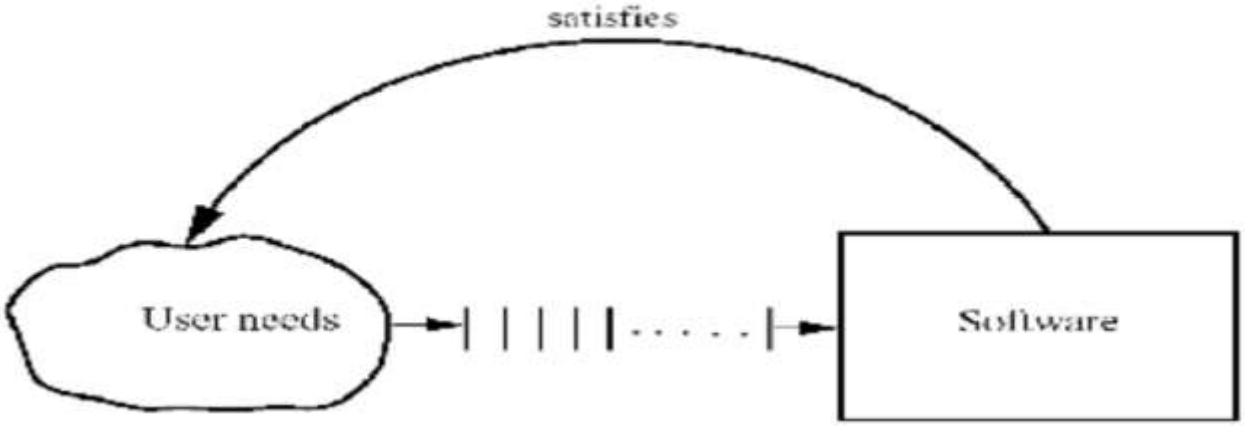
- The software is a set of instructions, technically referred to as programs that perform operations and specific tasks based on the commands of the user.
- Example:-
- System Software: - Operating System, Device driver, Firmware, utilities.
- Application Software: - Word Processors, Database software, Multimedia software, web browsers.
- Programming Software: - Compiler, Debuggers, Linkers, Malware.

Software consists of

• Instructions (computer programs) that when executed provide desired function and performance.

• Data structures that enable the programs to adequately manipulate information.

• Documents that describe the operation and use of the programs.



Software Engineering

• Software engineering is the branch of computer science that deals with designing, developing, testing,

and maintaining software applications.

• Software applies engineering principles and programming language knowledge to build software solutions for end users.

• Example:-

• Computer games, business applications, operating systems, network control system

Software Components

• **Lowest Level-** It mirrors the instruction set of the h/w. makes efficient use of

• memory and optimize programmed execution speed.

• **Machine language-** 1 Level

• **Assembly language**-2 Level

• **Mid Level - 3** Level. Machine independents used to create procedural description

• of the program. More efficient than machine language C, and C++.

• **The highest fourth level** is non-procedural moves the developer further from

• the h/w. it uses geographical icons**.**

A program is a subset of software and it becomes software only if documentation and an operating procedure manual are prepared.

• Program

• Documents

• Software documentation consists of all the descriptions, programs, graphics, and instructions about the design, coding, testing, and preparation of software.

• Operating Procedure (User Manual & Operational Manual)

• Provides information about what software is how to work with it how to install it on your system and how to control all the activities of the software.

There are three components of the software program, documentation, and operating procedure.

• **Program**: - list of instruction that tell a computer what to do.

• **Documentation**:- source information about the product contained in design documents detailed code comment use and maintain the software, such as user manuals and technical guides.

• **Operating Procedures –** Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

• **Code**: the instructions that a computer executes to perform a specific task or set of tasks.

• **Data**: the information that the software uses or manipulates.

• **User** interface: how the user interacts with the software, such as buttons, menus, and text fields.

• **Libraries**: pre-written code that can be reused by the software to perform common tasks.

• **Test cases**: a set of inputs, execution conditions, and expected outputs that are used to test the software for correctness and reliability.

• **Configuration files**: files that contain settings and parameters that are used to configure the software to run in a specific environment.

• **Build and deployment** scripts: scripts or tools that are used to build, package, and deploy the software to different environments.

• **Metadata**: information about the software, such as version numbers, authors, and copyright information.• All these components are important for software development, testing, and deployment.

program	Software
Usually small in size	Large
The author himself is the sole user	A large number of users
Single developer	Team of developers
Lacks proper user interface	Well-designed interface
Lacks proper documentation	Well-documented & user manual prepared
Ad hoc development	Systematic development

Software myths

• Many software problems are due to myths that arise during the initial stages of software developments•

Management myths, user myths, and developers myths.

Management myths	Reality
The members of the organization can acquire all the information they require from a manual that contains standards, procedures and principles.	Standards are rarely used, and standards are often out of date and incomplete.
State of art hardware is the essential ingredient for successful software production.	Software tools are usually more important than hardware tools for achieving quality and productivity.
If the project gets behind the planned schedule increasing the number of programmers can reduce the time.	Adding more manpower to project the planned schedule is difficult as it further delays the project.

User myth	Reality
The initial process is enough to start development detailed requirements can be added at later stages	Insufficient knowledge about requirements is the major cause of software failure
Software is flexible hence software requirement changes can be added during any phases of the development process.	The impact of changes varies according to the time of introduction, early requests for change can be accommodated easily and cheaply.

Developer myth	Reality
The software development is considered complete when the code is delivered	The efforts are expended after the software is delivered to the user.
Software engineering makes unnecessary documentation, which slows down the project.	The success of a project does not only depend on the quality of program documentation and software configuration are also essential.
The quality can be assessed only after the program is executed.	Software quality assurance mechanisms is the formal technical review, which can be applied from the inception of the project.

Characteristics

• Functionality

• Efficiency

• Portability

• Maintainability

• Reliability

• Usability

• **Functionality:**- performance of software to its intended purpose.

• **Reliability:**- the ability of software to perform a required functional condition for a specified period.

• **Usability**:- the software is easy to use.

• **Efficiency:**- ability of software to use system resources most efficiently.

• **Maintainability:**- The software system can be modified to add capabilities, improve system performance, or correct

Software Crisis

Software fails because it

• Programmers have skills for programming but without the engineering mindset about a process discipline

• Crash frequently

• Expensive

• Difficult to alter, debug, enhance

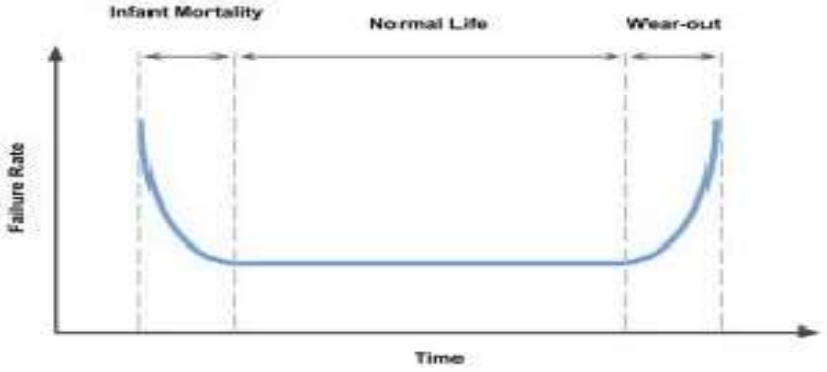
• Often delivered late

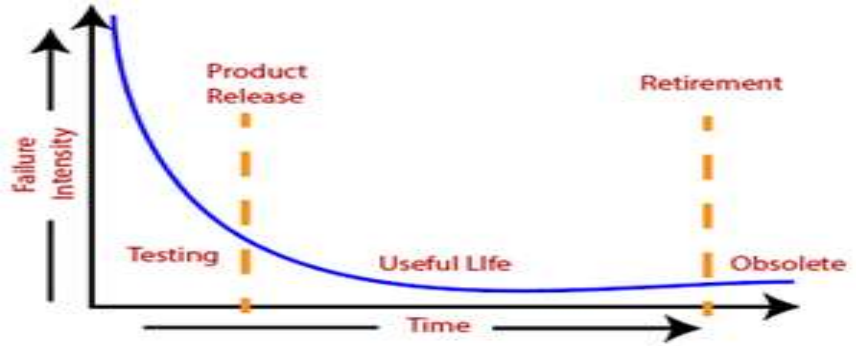
• Not delivered.

• Use resources non-optimally

• Software professionals lack engineering training

For hardware same production cost every time, for software production cost only for the first time and then only the maintenance cost. This life cycle of the hardware follows the bathtub curve, while in the life cycle of software failure intensity goes down with time.





(b) Software Product

Software engineering Process

The concept of software Engineering was evolved in 1986 by BOHEM to reduce the effect of his software crisis because software engineering is defined as a scientific and systematic approach to developing, operating, and maintaining software projects to meet a given specific object it beautifully envisages the concept of the life cycle of any software project

through the following five phases:-

- Requirement analysis
- Design
- Coding
- Testing
- Maintenance

Software Engineering	Software Programming
Single Developer	Teams of developers with multiple roles
Small Application(Toy)	Complex System
Short lifespan	Indefinite lifespan
Single or few stakeholders	Numerous stakeholders
One-of-a-kind systems	System families

Similarities between software processes and conventional engineering process

- Software Engineering Process is an engineering process that is mainly related to computers and programming and developing different kinds of applications.
- Conventional Engineering Process is an engineering process that is highly based on empirical knowledge and is about building machines, and hardware. It is a process that mainly involves science and mathematics.

Both Software Engineering and Conventional Engineering Processes become automated after some time.

- These processes are making our day-to-day place better.• These processes have a fixed working time.
- Both processes must consist of deeper knowledge.

Dereferences

Software Engineering Process	Conventional Engineering Process
Majorly involves computer science, information technology, and discrete mathematics.	The conventional Engineering Process is a process that majorly involves science, mathematics, and empirical knowledge.
It is mainly related to computers, programming, and writing codes for building applications	It is about building machines, hardware, buildings, etc.
In Software Engineering Process construction and development costs are low.	In Conventional Engineering Process construction and development cost is high.
It can involve the application of new and untested elements in software projects.	It usually applies only known and tested principles to meet product requirements.
In the Software Engineering Process, most development effort goes into building new designs and features.	In Conventional Engineering Processes, most development efforts are required to change old designs.
It majorly emphasizes quality.	It majorly emphasizes mass production

Software Quality Attributes

- Software Quality is a term used to measure the degree of excellence of software. Software Quality attributes are extremely important while designing a software application
- There is a misconception that if the software application is bug-free then the quality of the software is high. However, a bug-free product is just one of the Software Quality Attributes. The quality of the software also depends on the user requirements, satisfaction, clearer design, and usability.
- Software Quality attributes help to measure the quality of the software from different angles.
- Software Quality attributes can be broadly classified into 5 types:
- Design, Runtime, System, User, and Non-runtime qualities**.**
- Reliability
- Maintainability
- Usability
- Portability
- Efficiency
- Security
- Flexibility
- Scalability
- Compatibility

- Supportability
- Reusability
- Interoperability

• Reliability is the ability of software applications to behave as expected and function under the maximum possible load.

• Components:- Availability, Recoverability, Fault Tolerance.

• Maintainability refers to how easily software developers can add new features and update existing features with new technologies.

• Usability is tied to application performance, application, design, and accessibility.

• The portability quality attribute refers to how easily the system can be ported or migrated to other environments containing different hardware or operating system specifications.

• Correctness refers to the ability to behave or function as per software requirement specifications.

• Efficiency can be defined as the time taken by the system to complete a specific task, and performance of the application**.**

• Security attribute focuses on the ability to safeguard applications, data, and information from unauthorized entities.

• The software and log a defect and how easy it is to automate the software applications. Application design should focus on making the testing easier and faster.

- Flexibility refers to how quickly an application can adapt to future and current technology demands.
- Scalability is how easily the system can handle increasing demands without affecting the application's performance.
- Compatibility focuses on the ability to work software systems on different operating systems, and browsers seamlessly as expected without affecting any functionality.
- Supportability is the degree to which a software system can provide useful information for identifying and resolving the issues when the application/functionality stops working.
- Reusability is the degree to which software components can be reused in another application or the same application. Reusable software components reduce the development cost and effort.
- Interoperability refers to the ability to communicate or exchange data between different systems.

Software Development Life Cycle (SDLC) Models

- Software-development life-cycle is used to facilitate the development of a large software product in a systematic, well-defined, and cost-effective way.

- The software development process comprises different phases. These phases work into the bottom approach, implying that the phase table input from the previous phases adds features, and then produces output. The output from different phases is referred to as the intermediate product, work product, or derivable.

- Various reasons for using a life-cycle model include:
  - Helps to understand the entire process
  - Enforces a structured approach to development
  - Enables planning of resources in advance
  - Enables subsequent controls of them
  - Aids management to track the progress of the system
  - **Activities undertaken during feasibility study: -** The main aim of the feasibility study is to determine whether it would be financially and technically feasible to develop the product.

- **Activities undertaken during requirements analysis and specification: -** The requirements analysis and specification phase aims to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely Requirements gathering and analysis, this phase ends with the preparation of Software Requirement Specification (SRS).

- **Activities undertaken during design: -** The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language. Design specification Document is outcome of this phase.

- **Activities undertaken during coding and unit testing:-**The purpose of the coding and unit testing phase of software development is to translate the software design into source code. Each component of the design is implemented as a program module. The end-product of this phase is a set of program modules that have been individually tested. Code Listings are generated after this phase.

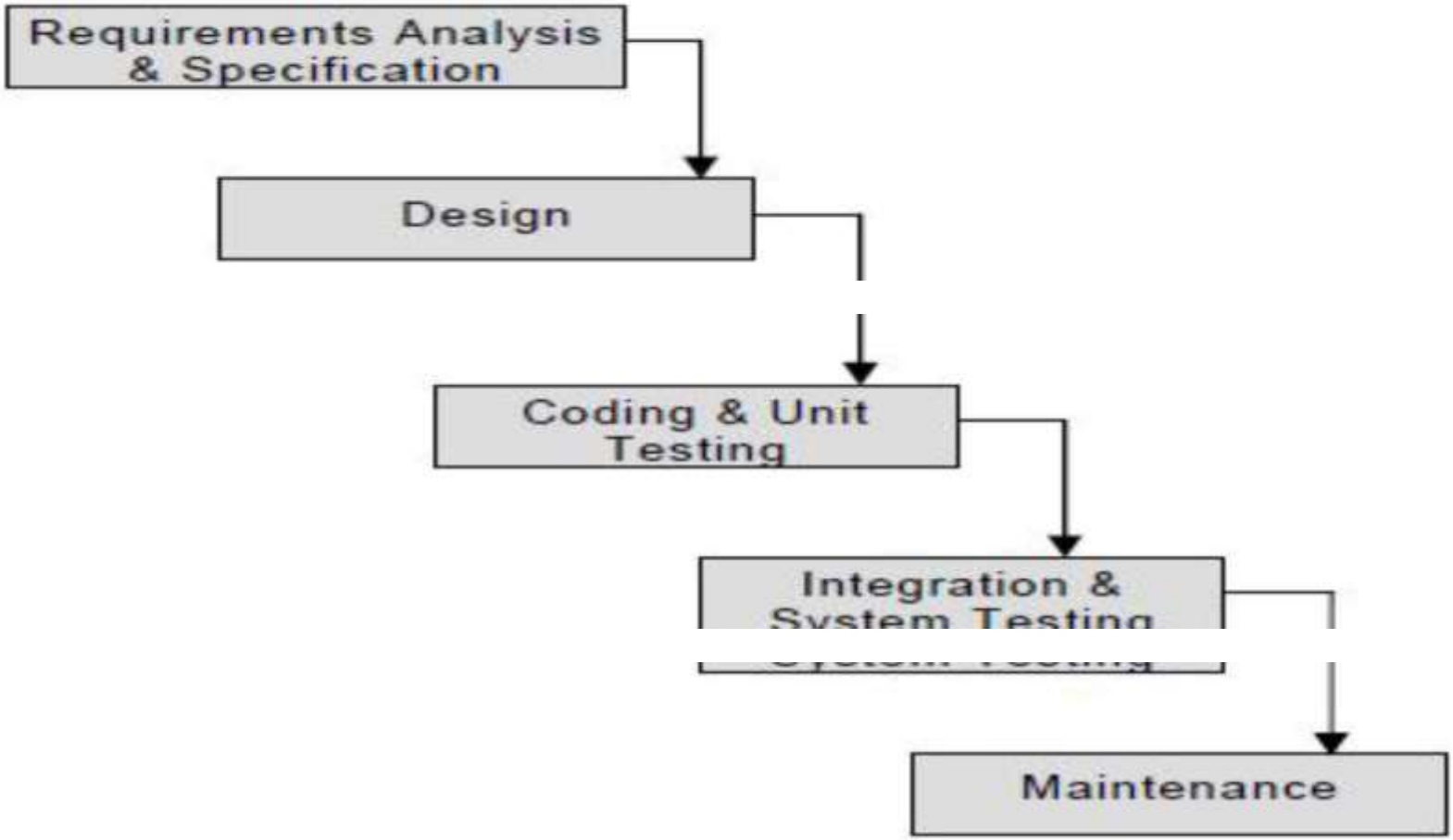
- **Activities undertaken during integration and system testing: -** Integration of different modules is undertaken once they have been coded and unit tested during each integration step, the partially integrated system is tested and a set of previously planned modules are added to it. Finally, when all the modules have been successfully integrated tested, system testing is carried out. The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document. Test reports are generated after this phase.

- **Activities undertaken during maintenance:** - Maintenance of a typical software product requires much more than the effort necessary to develop the product itself. Many studies carried out in the past confirm this and indicate that the relative effort the development of a typical software product to its maintenance. This phase continues till the software is in use.

1. Water Fall Model

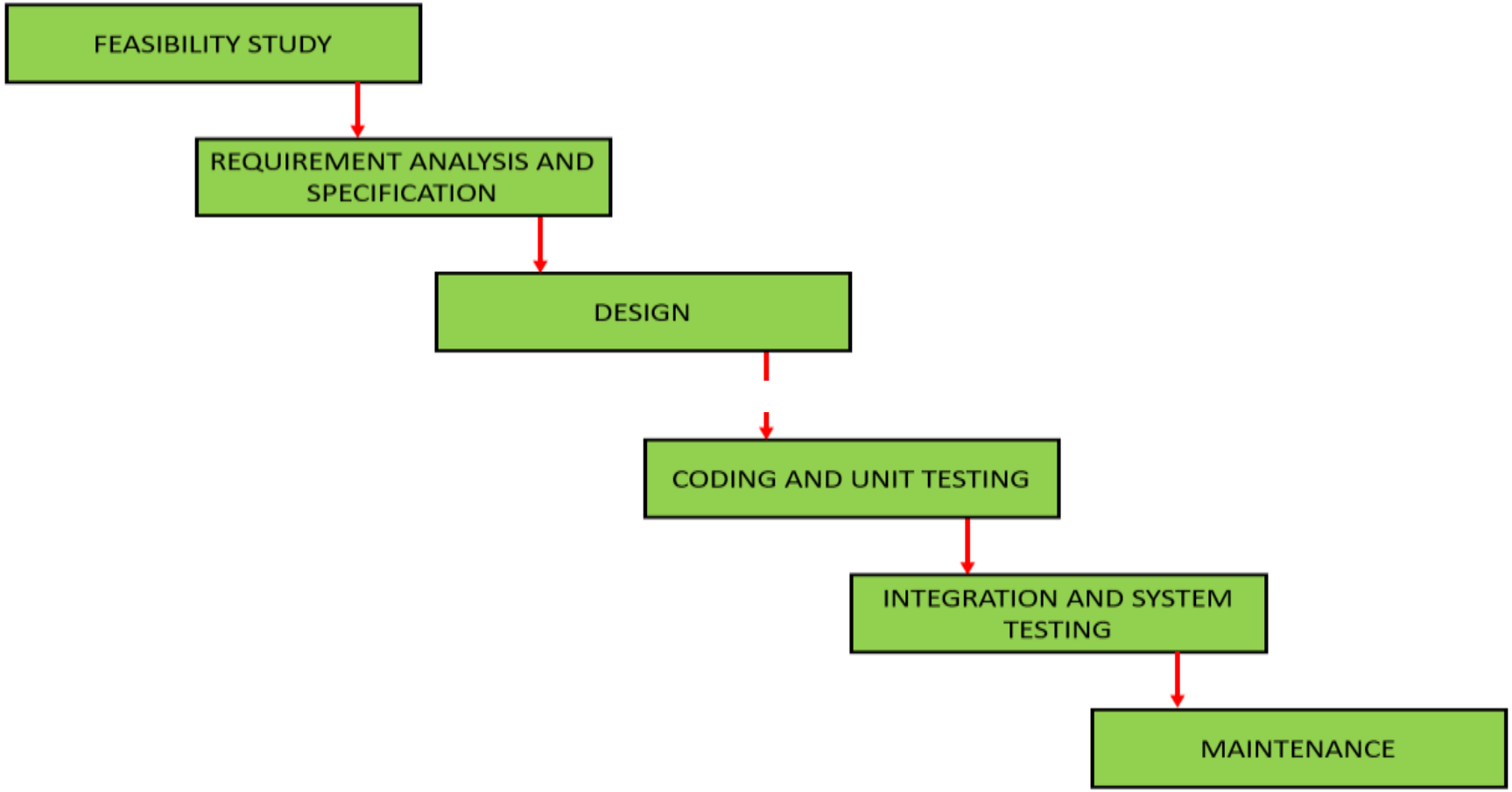
- Linear Sequential Life Cycle Model.
- It is very simple to understand and use.

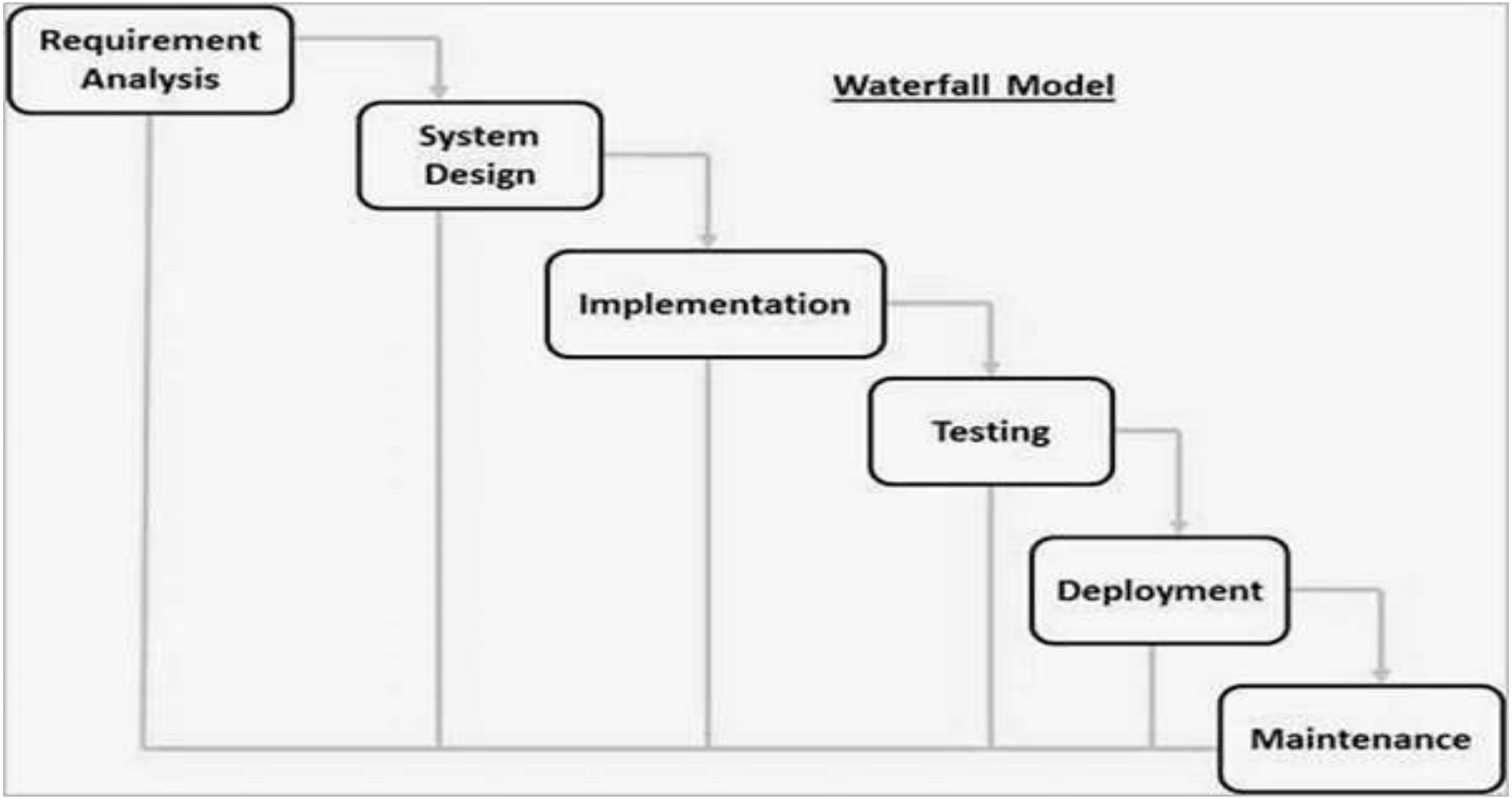
- In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.



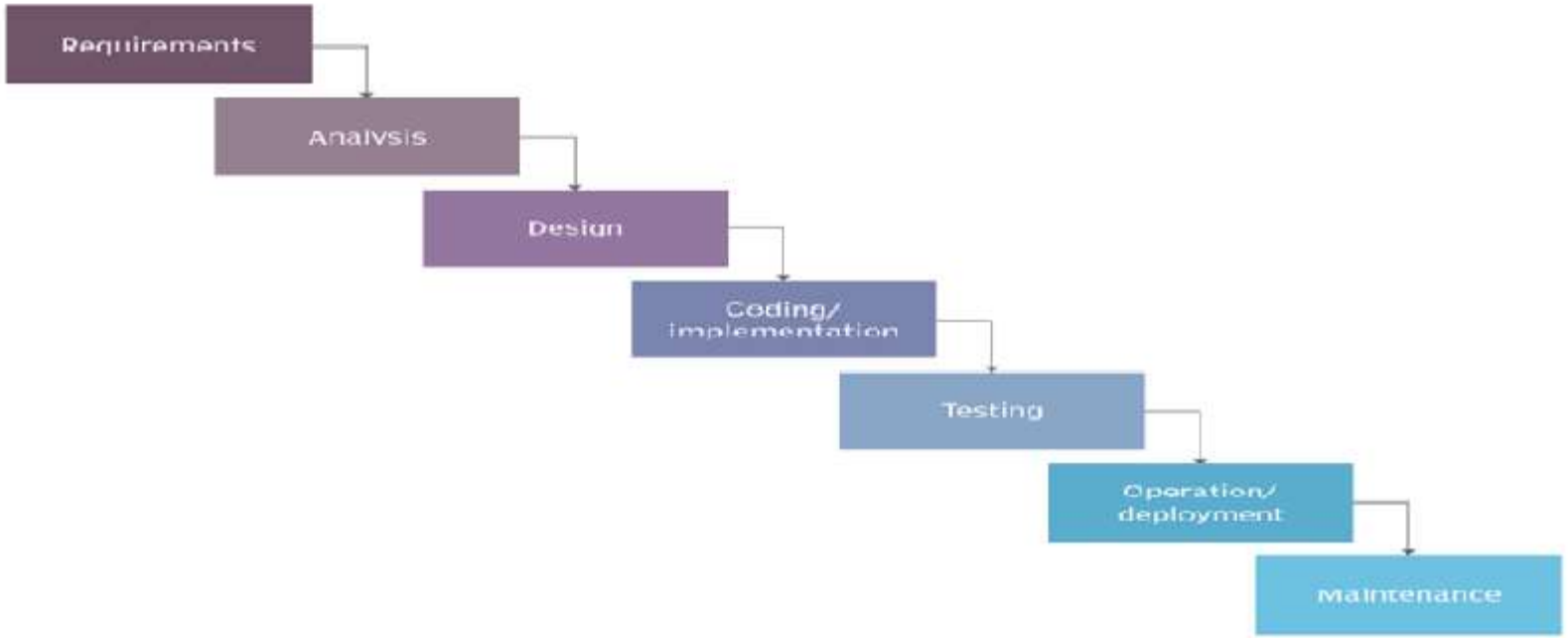
Classical Waterfall Model

Iterative Water fall Model




Waterfall Model
Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

Waterfall model



Waterfall Model

• In water fall model each step is dependent on the output of previous one.

• This model is good to develop a project according to fixed or unchanging requirements set forth at the beginning of the project.

• The waterfall model are well defined, predictable and have specific documentation.

Characteristics

• Fixed requirements;
• Ample resources;
• Established timeline;
• Well-understood technology;
• Unlikely to require significant changes;

Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. 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 very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Disadvantages

• Not a good model for complex and object-oriented projects.• It is difficult to measure progress within stages
• High amounts of risk and uncertainty.

• It doesn't consider error correction.

• Limited Flexibility

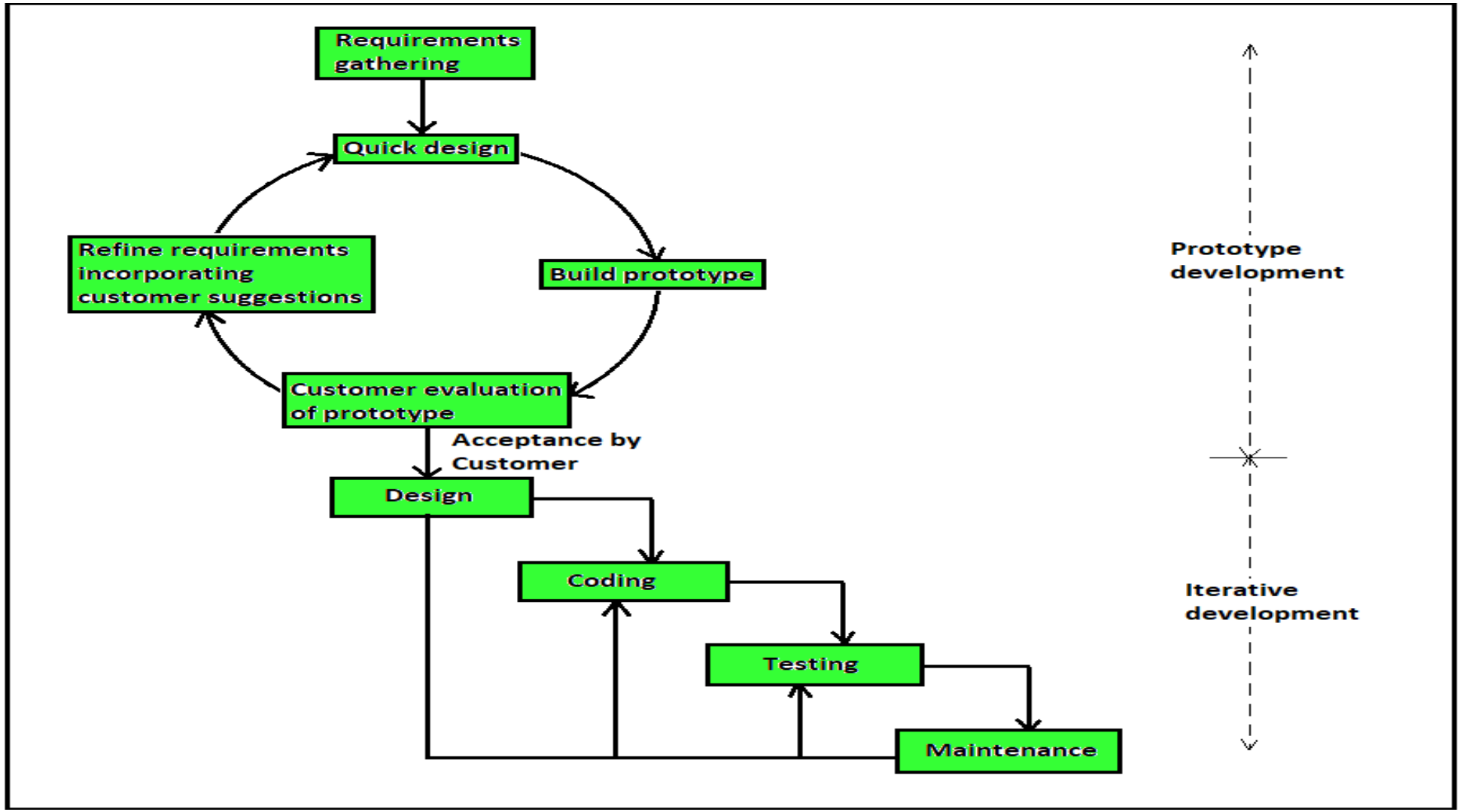
2. Prototype Model

- The prototype model is applied when there is an absence of information regarding input and output requirements in the software.

• This model is developed on the assumption that it is difficult to know all the requirements at the beginning of a project.

• Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.

• By creating major user interfaces without any substantive coding in the background to give the users a feel of what the system will look like.


Requirements gathering
Quick design
Refine requirements incorporating customer suggestions
Build prototype
Customer evaluation of prototype
Acceptance by Customer
Design
Coding
Testing
Maintenance
Prototype development
Iterative development

Prototype model life cycle

• Requirement gathering and Analysis

• Quick design

• Build prototype

• Assessment or User evaluation

• Prototype Redefine

Advantages

• Provides a working model to the user early in the process, enabling early assessment and increasing user confidence.

• Developer gains experience and insight by developing a prototype resulting in better implementation of requirements.

• There is a great involvement of users in software development.

• Helps in reducing risks associated with the project.

Disadvantages

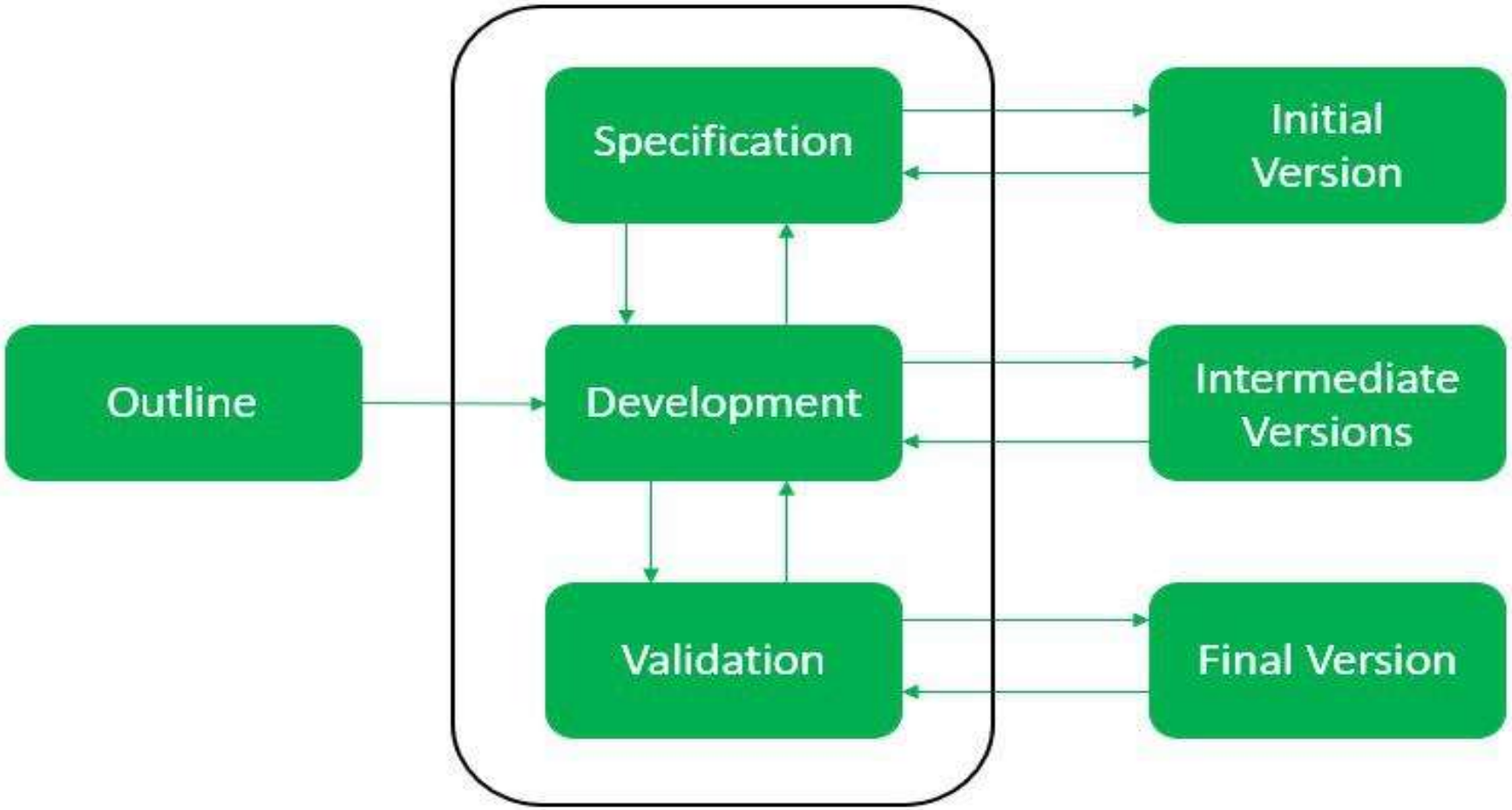
- It is difficult to find all the requirements of the software initially.
- It is very difficult to predict how the system will work after development

3. Evolutionary Process Model

• The evolutionary process model resembles the iterative enhancement model.

• The same phases are defined for the waterfall model occurs here in a cyclical fashion.

• In evolutionary development, requirements are implemented by category rather than by priority.


Specification
Initial Version
Outline
Development
Intermediate Versions
Validation
Final Version

- The phases of the software construction are interleaved
- Feedback from the user is used throughout the entire process
- The software product is refined through many versions.

Advantages

- Deals constantly with changes
- Provides quickly an initial version of the system
- Involves all development teams

- **Disadvantages**

- Quick fixes may be involved
- Invisible process, not well-supported by documentation
- The system’s structure can be corrupted by continuous change

4. Iterative Enhancement Model

- In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

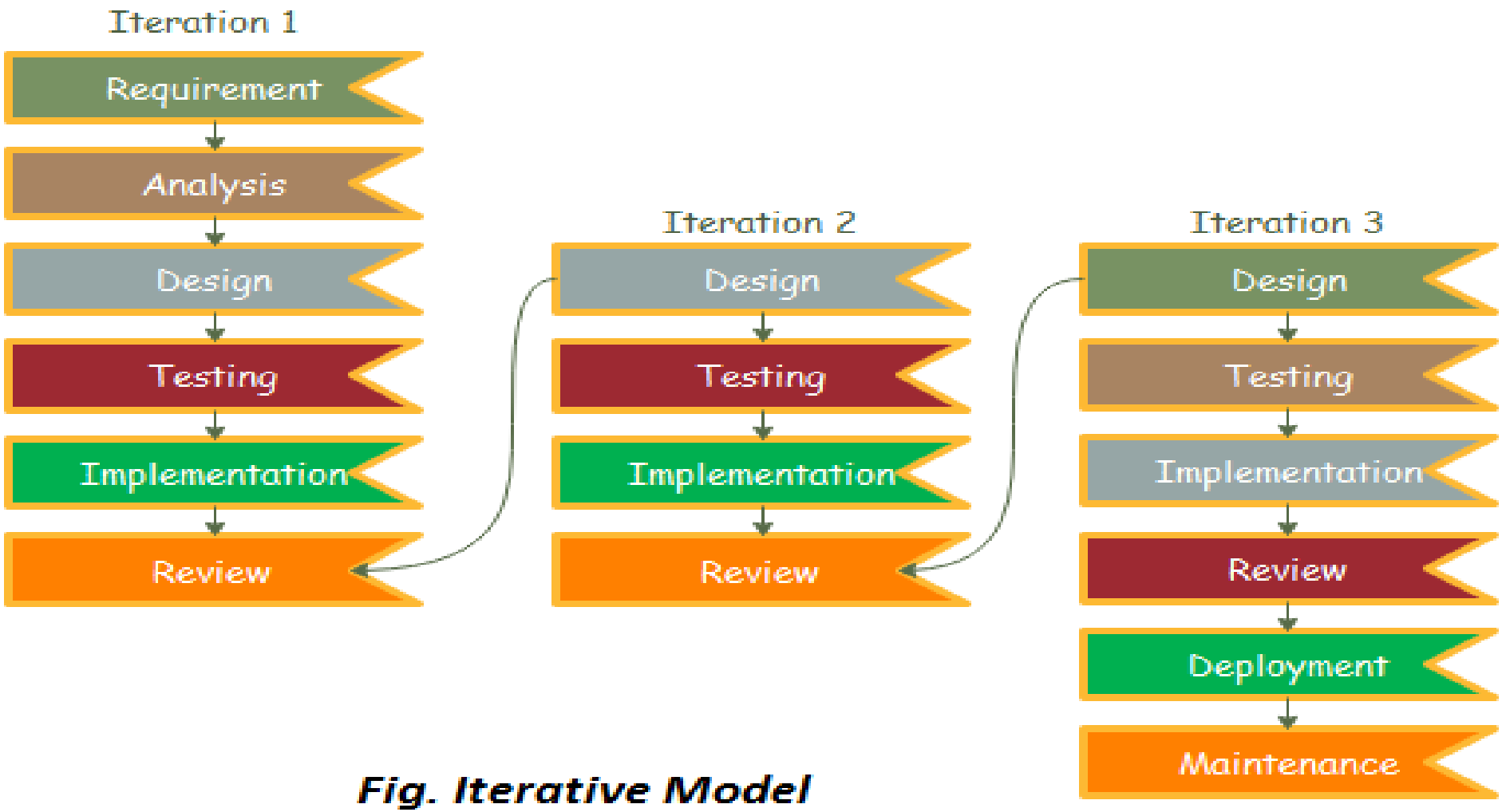


Fig. Iterative Model

Advantages

• Testing and debugging during smaller iteration is easy.

• A Parallel development can be planned.

• It is easily acceptable to the ever-changing needs of the project.

• Risks are identified and resolved during iteration.

• Limited time spent on documentation and extra time on designing.

Disadvantages

• It is not suitable for smaller projects.

• More Resources may be required.

• Design can be changed again and again because of imperfect requirements.

• Requirement changes can cause over budget.

• Project completion date not confirmed because of changing requirements.

5. Spiral Model

• In the spiral model activities are organized in a spiral, which has many cycles.

• This model combines the features of the prototyping model and waterfall model.

• It is advantageous for large, complex, and expensive models.

• The spiral model determines requirements and problems in developing the prototypes.

• Spiral model guides and measures the need for risk management in each cycle of the spiral model.

• This model appears like a spiral with many loops. The exact number of loops in the spiral is not fixed. Each loop of the spiral represents a phase of the software process.

• The innermost loop might be concerned with a feasibility study. The next loop is with requirements specification, the next one is with design, and so on. Each phase in this model is split into four sectors (or quadrants).

First quadrant (Objective Setting)

• During the first quadrant, it is needed to identify the objectives of the phase.• Examine the risks associated with these objectives.

Second Quadrant (Risk Assessment and Reduction)

• A detailed analysis is carried out for each identified project risk.

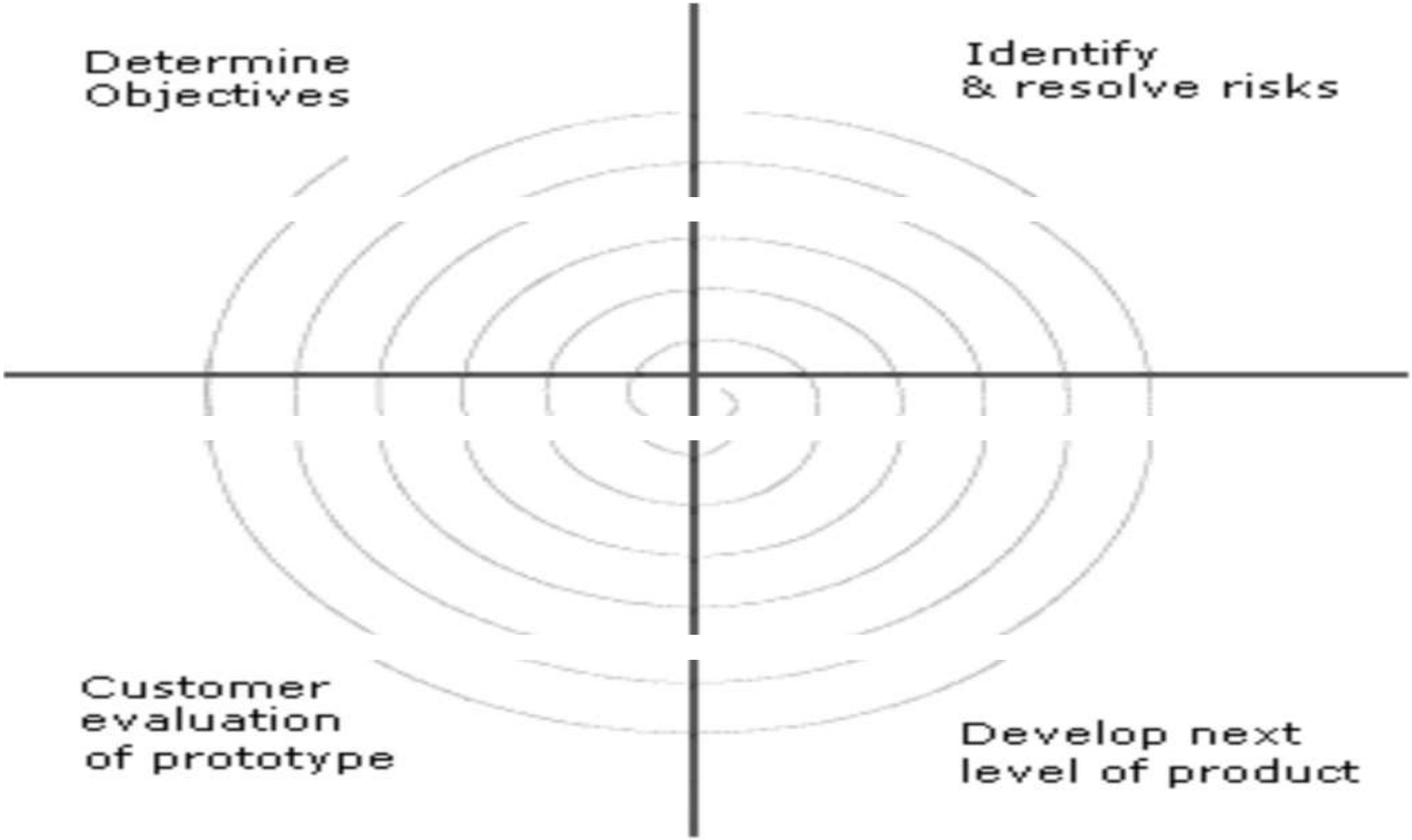
• Steps are taken to reduce the risks.

Third Quadrant (Development and Validation)

• Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning)

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


Determine
Objectives
Identify
& resolve risks
Customer
evaluation
of prototype
Develop next
level of product

Advantages of Spiral Model

• It is risk-driven model.

• It is very flexible.

• Less documentation is needed.

• It uses prototyping

Disadvantages of Spiral Model

• No strict standards for software development.

• No particular beginning or end of a particular phase.

Difference

Water Fall Model	Spiral Model
Separate and distinct phases of specification and development.	The process is represented as a spiral rather than as a sequence of activities with backtracking
After every cycle, a useable product is given to the customer.	Each loop in the spiral represents a phase in the process
Effective in situations where requirements are defined precisely and there is no confusion about the functionality of the final product.	No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
Risks are never explicitly assessed and resolved throughout the process	Risks are explicitly assessed and resolved throughout the process

6. RAD Model (Rapid Application Development): -

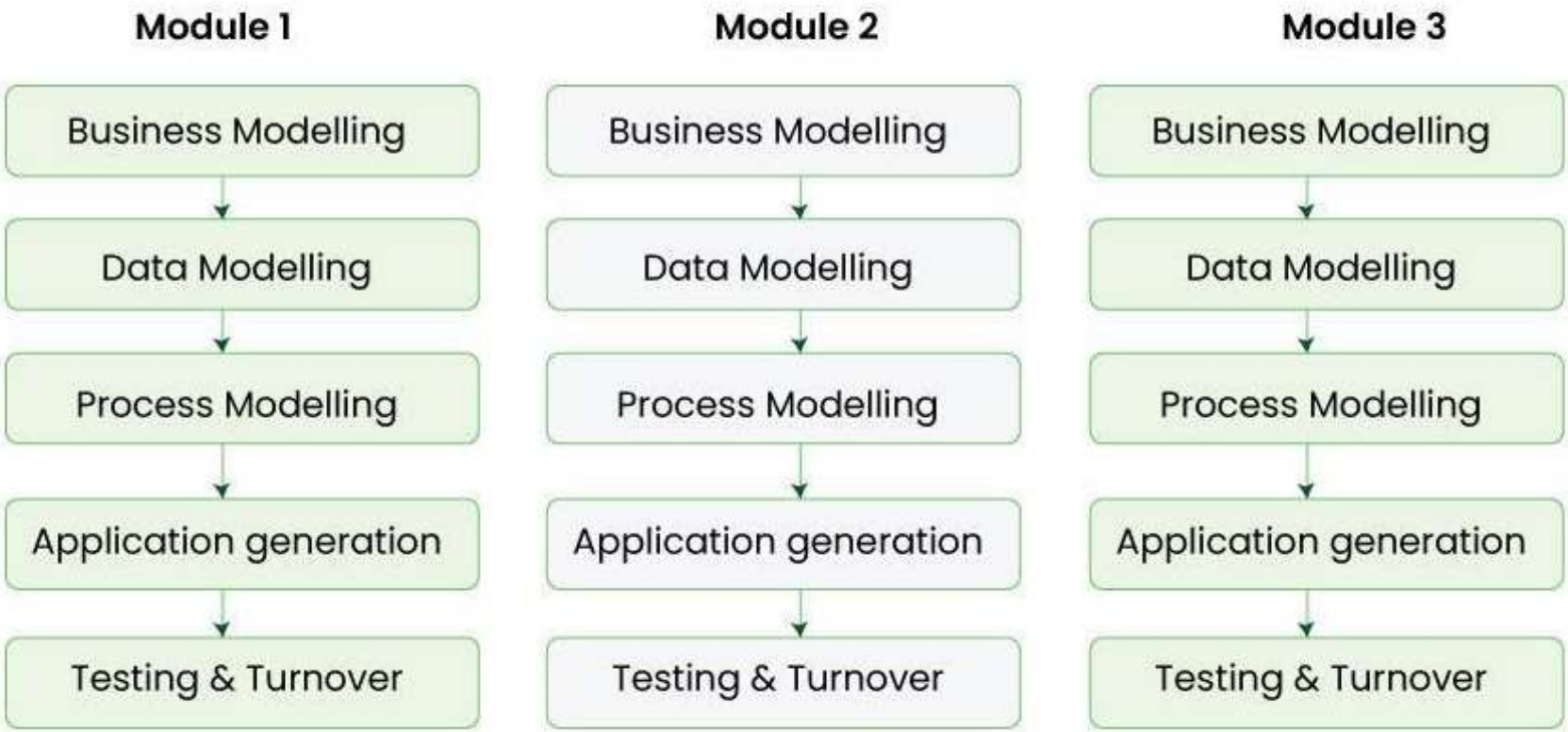
RAD model stands for rapid application development model.

The methodology of the RAD model is similar to that of the incremental or waterfall model.

It is used for small projects.

If the project is large then it is divided into many small projects and these small projects are planned one by one and completed. In this way, by completing small projects, the large project gets ready quickly.

- In the RAD model, the project is completed within the given time and all the requirements are collected before starting the project. It is very fast and there are very less errors in it.



Advantages of RAD Model

- It reduces the time taken in development.
- In this, the components are reused.
- It is flexible and it is easy to make any changes in it.
- It is easy to transfer like scripts because high-level abstraction and intermediate codes are used in it.

- There are very few defects in it because it is a prototype by nature.
- In this, productivity can be increased in less time with fewer people.
- It is cost-effective.
- It is suitable for small projects.

Disadvantages of the RAD Model

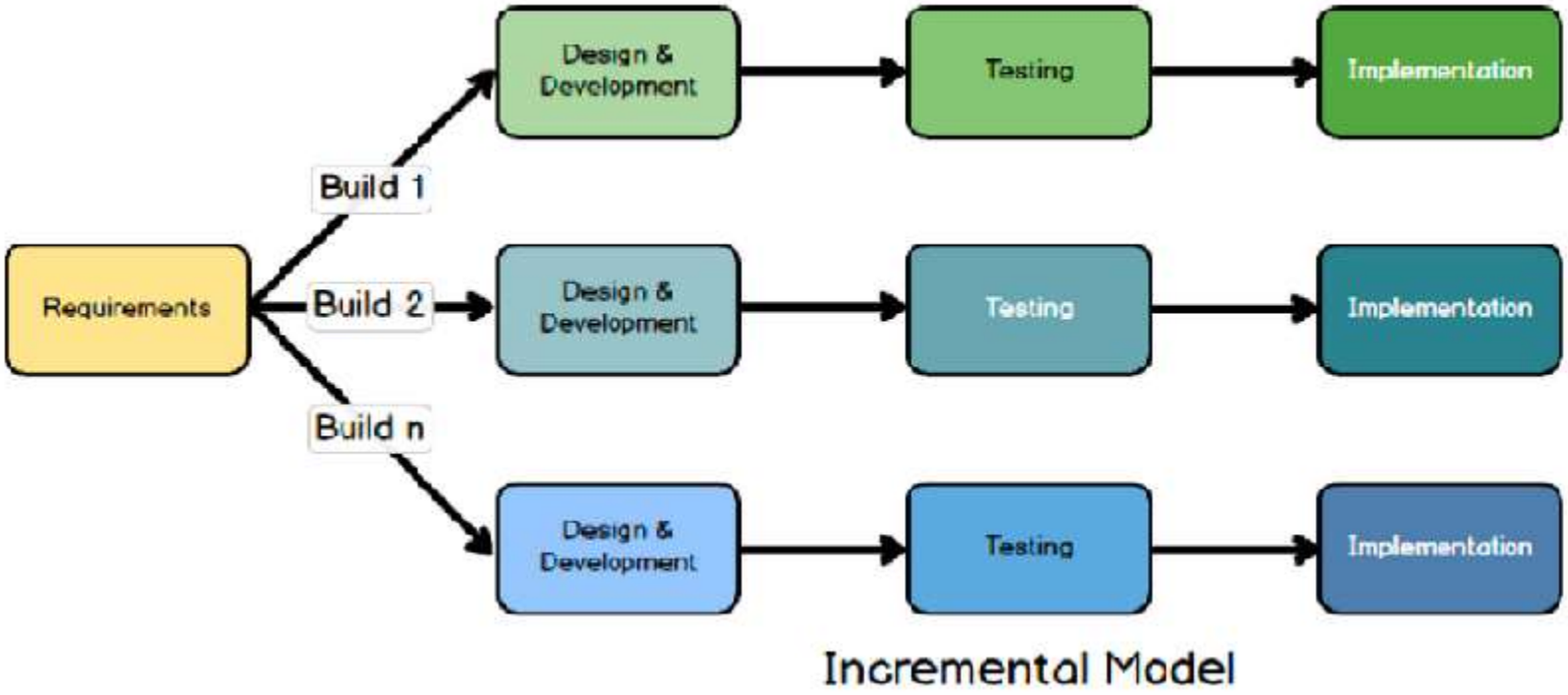
- In this, we need highly skilled developers and designers.
- It is very difficult to manage.
- It is not suitable for projects that are complex and take a long time.
- In this, feedback from the client is required for the development of each phase.
- Automated code generation is very expensive.
- This model is suitable only for component-based and scalable systems.

7. Incremental Process / Incremental Development Model

Incremental development is based on the idea of developing an initial implementation exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.

• In the Incremental model the whole requirement is divided into various builds. Multiple

- Development cycles take place here, making the life cycle a “multi-waterfall” cycle.
- Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, implementation, and testing phases.
- A working version of the software is produced during the first module, so you have working software early on during the software life cycle.
- Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.
- Generally, the early increments of the system should include the most important or most urgently required functionality.
- tries to combine the benefits of both prototyping and the waterfall model



Incremental Model

Advantages of the Incremental model:

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customers can respond to each build.
- Easier to manage risk because risky pieces are identified and handled during their iteration.

Disadvantages of the Incremental model

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.

• Total cost is higher than waterfall

8. Agile Model

The Agile model is a combination of iterative and incremental models, that is, it is made up of iterative and incremental models.

- In the Agile model, focus is given to process adaptability and customer satisfaction
- In earlier times, the iterative waterfall model was used to create software. But in today's time, developers have to face many problems. The biggest problem is that in the middle of software development, the customer asks to make changes to the software
    - Agile Testing Methods:
    - Scrum
    - Crystal
    - Dynamic Software Development Method(DSDM)
    - Feature Driven Development(FDD)
    - Lean Software Development
    - Extreme Programming(XP)

Scrum

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

There are three roles in it, and their responsibilities are:

- Scrum Master: The scrum can set up the master team, arrange the meeting and remove obstacles for the process
- Product owner: The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
- Scrum Team: The team manages its work and organizes the work to complete the sprint or cycle.

eXtreme Programming(XP)

This type of methodology is used when customers are constantly changing demands or requirements, or when they are not sure about the system's performance.

Crystal:

There are three concepts of this method-

1. Chartering: Multi activities are involved in this phase such as making a development team, performing feasibility analysis, developing plans, etc.
2. Cyclic delivery: under this, two more cycles consist, these are:
    - Team updates the release plan.
    - Integrated product delivers to the users.
3. Wrap-up: According to the user environment, this phase performs deployment and post-deployment.

Dynamic Software Development Method (DSDM):

DSDM is a rapid application development strategy for software development and gives an agile project distribution structure. The essential features of DSDM are that users must be actively connected, and teams must be given the right to make decisions. The techniques used in DSDM are:

1. Time Boxing
2. Moscow Rules
3. Prototyping

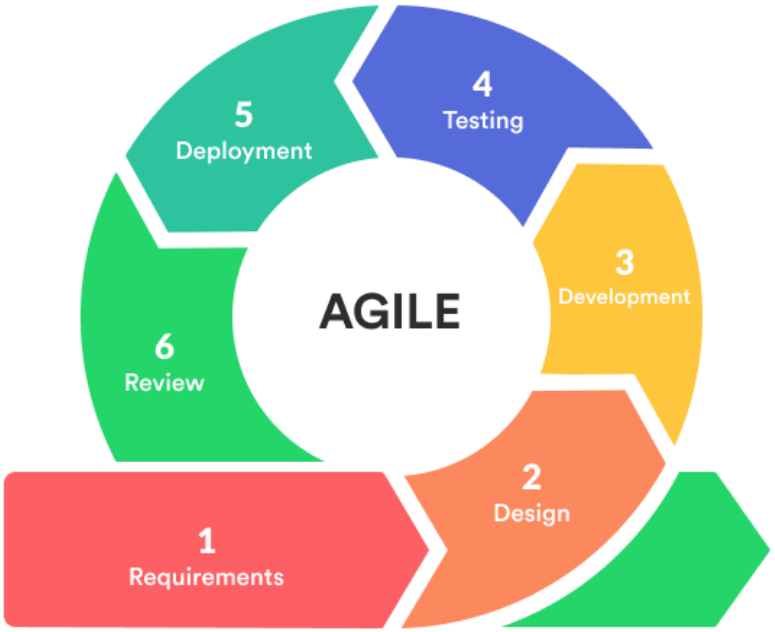
The DSDM project contains seven stages:

1. Pre-project

2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

Feature Driven Development (FDD):

This method focuses on "Designing and Building" features. In contrast to other smart methods, FDD describes the small steps of the work that should be obtained separately per function.



Advantages (Pros) of the Agile Method:

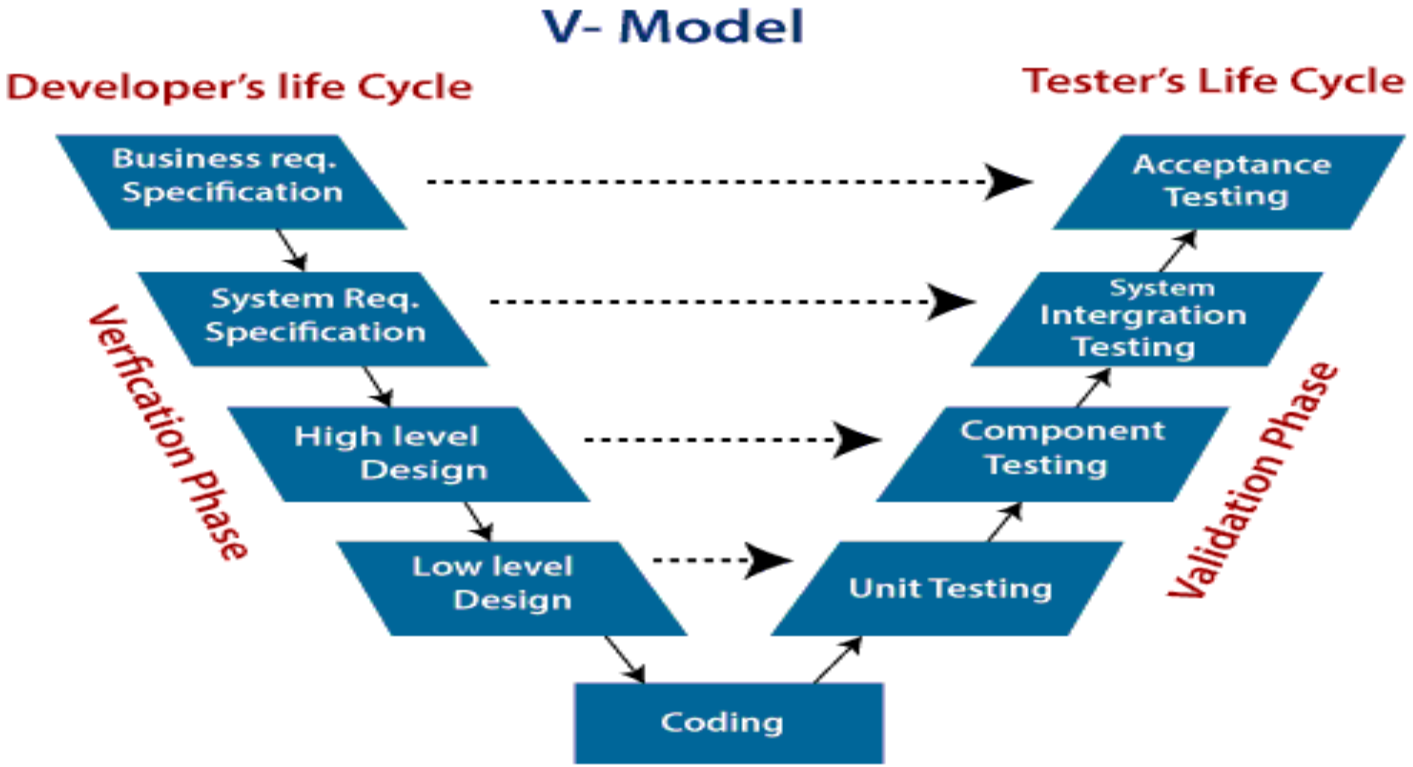
- Frequent Delivery
- Face-to-face communication with clients.
- Efficient design and fulfills the business requirement.
- Anytime changes are acceptable.
- It reduces total development time.

Disadvantages (Cons) of the Agile Model:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project is completed and the developers allotted to another project, maintenance of the finished project can become difficult.

9. V Model (Validation Model): -

V-Model is also referred to as the Verification and Validation Model. In this, each phase of SDLC must be completed before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.



Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements are met.

Validation: It involves a dynamic analysis method (functional, non-functional), and testing is done by executing code. Validation is the process of classifying the software after the completion of the development process to determine whether the software meets the customer's expectations and requirements.

So V-Model contains Verification phases on one side the Validation phases on the other side. The verification and Validation process is joined by the coding phase in a V-shape. Thus it is known as V-Model.

Advantage (Pros) of V-Model:

1. Easy to Understand.
2. Testing Methods like planning, and test designing happen well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, have to be updated.