(CT-21015)
Software Engineering
Mini Project Stage- II

Unit I-Software Development Process

(CT-21015) Software Engineering Mini Project Stage- II

- Teaching Scheme
  - Lectures: 2 Hrs / Week
  - Laboratory: 2 Hrs/Week

- Examination Scheme
  - Quiz/Assignments: 20 Marks
  - Theory Exam:40 Marks
  - Project: 40 Marks (Submission in Stages)

UNIT- I Software Development process

Contents

- Software Engineering basics
- Software Crisis and Myths
- Software Process and development
- Software life cycle and Models
- Analysis and comparison of various models
- Agile process

What is Software Engineering?

- **Software:** It is collection of integrated programs. Means it carefully organized instructions and code written by developers on any of various particular computer languages.

- **Engineering:** It is the application of scientific and practical knowledge to invent, design, build, maintain and improve frameworks, processes etc.

- **Software Engineering:** It is an **engineering branch** relate to the evolution of software product using well-defined scientific principles, techniques and procedures. Software engineering is the process of designing, developing, and maintaining software systems. A good software is one that meets the needs of its users, performs its intended functions reliably, and is easy to maintain.

  The result of software engineering is an effective and reliable software product.

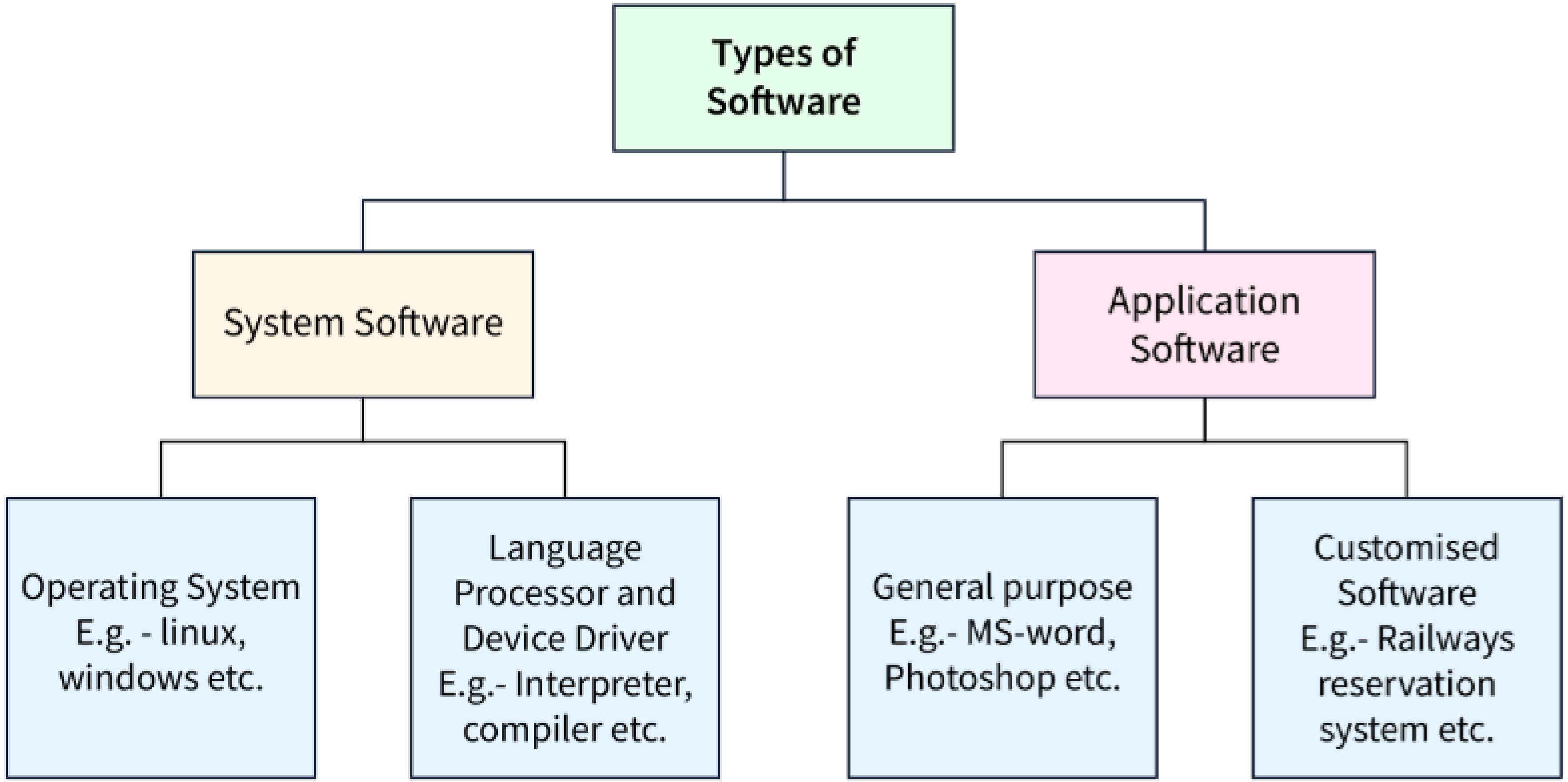
Need of Software Engineering

- **Handling Big Projects:** Corporation must use SE to handle large projects without any issues.
- **To manage the cost:** Software engineering programmers plan everything and reduce all those things that are not required.
- **To decrease time:** It will save a lot of time if you are developing software using a software engineering technique.
- **Reliable software:** It is the company's responsibility to deliver software products on schedule and to address any defects that may exist.
- **Effectiveness:** Effectiveness results from things being created in accordance with the software standards.

Characteristics of Good Software

Operational	Transitional	Maintenance
Budget Efficiency Usability Dependability Correctness Functionality Safety Security	Interoperability Reusability Portability Adaptability	Flexibility Maintainability Modularity Scalability

Nature of Software


Types of Software
System Software
Application Software
Operating System E.g. - linux, windows etc.
Language Processor and Device Driver E.g.- Interpreter, compiler etc.
General purpose E.g.- MS-word, Photoshop etc.
Customised Software E.g.- Railways reservation system etc.

Nature of Software (Cont…)

1. System Software:

- System software is software designed to provide a platform for other software's.
- It is a interaction between hardware & application software.
- Examples: Operating Systems like macOS, Linux, Android and Microsoft Windows.

2. Application Software:

- Application Software is a computer program designed to carry out a specific task as per user & business need.
- Examples: Social medias apps, Gaming apps, Word processing apps, Multimedia apps, Banking apps, Shopping apps, Booking apps etc

Nature of Software (Cont...)

3. Engineering and Scientific Software:

- This software is used to facilitate the engineering function and task take real time.
- It has very high accuracy, complex formula evolution & data analysis.
- Examples: Weather prediction apps, Stock Market apps, Stress Analysis, Body measurement apps

4. Embedded Software:

- Embedded software resides within the system or product and is used to implement and control feature and function for the end-user and for the system itself.
- Example: Switches, Routers, Digital camera, Washing machine functionalities, Traffic control etc.

Nature of Software (Cont…)

5. Web Application:

- It is a client-server computer program which the client runs on the web browser.
- Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics.
- Examples: Online forms, Shopping carts, Gmail, Yahoo, Photo editing, File conversion etc.

6. Artificial Intelligence Software:

- It makes use of a nonnumerical algorithm to solve a complex problem.
- Application within this area includes robotics, expert system, pattern recognition, artificial neural network, theorem proving and game playing.
- Examples: Google Cloud, Azure studio, Tensor Flow, Salesforce etc.

Software Process & Development

- A software process is the set of activities and associated outcome that produce a software product.
- Software engineers mostly carry out these activities.

1. **Software Specifications:** The functionality of the software and constraints on its operation must be defined.
2. **Software Development:** The software to meet the requirement must be produced.
3. **Software Validation:** The software must be validated to ensure that it does what the customer wants.
4. **Software Evolution:** The software must evolve to meet changing client needs.

Software Myths

- Software Myths are **beliefs that do not have any pure evidence**. Software myths may lead to many **misunderstandings, unrealistic expectations, and poor decision-making** in software development projects. Some common software myths include:

➢ **The Myth of Perfect Software:** Assuming that it's possible to create bug-free software. In Reality, software is inherently complex, and it's challenging to eliminate all defects.

➢ **The Myth of Short Development Times:** Assuming that software can be developed quickly without proper planning, design, and testing. In Reality, rushing the development process can lead to better-quality software and missed deadlines.

➢ **The Myth of User-Developer Mind Reading:** It is assumed that developers can only understand user needs with clear and ongoing communication with users. But in reality, user feedback and collaboration are essential for correct software development.

➢ **The Myth of Cost Predictability:** It is thought that the cost of the software can be easily predicted, but in reality, many factors can influence project costs, and estimates are often subject to change. There are many hidden costs available.

➢ **The Myth of Endless Features:** It is believed that adding more features to software will make it better. But in reality, adding more features to the software can make it complex and harder to use and maintain. It may often lead to a worse user experience.

- **The Myth of No Testing Needed:** It is assumed that there is no need to test the software if the coder is skilled or the code looks good. But in reality, through testing is essential to catch hidden defects and ensure software reliability.
- **The Myth of "We'll Fix It Later":** It is assumed that a bug can be fixed at a later stage. But in reality, as the code gets longer and bigger, it takes a lot of work to find and fix the bug. These issues can lead to increased costs and project delays.
- **The Myth of No User Training Required:** It is assumed that users will understand and use new software without any training. But in reality, users need training and documentation to use the new software because the different methods used by different developers can be unique.
- **Perfect Software Is Possible:** The Idea of creating completely bug-free software is a myth. In Reality, software development is complex, and it's challenging to eliminate all defects.

Many more…

Understanding and addressing these software myths is important for successful software development projects. It helps in setting realistic expectations, improving communication, and making more informed decisions throughout the development process.

Software Crisis

There are various software crisis which include:

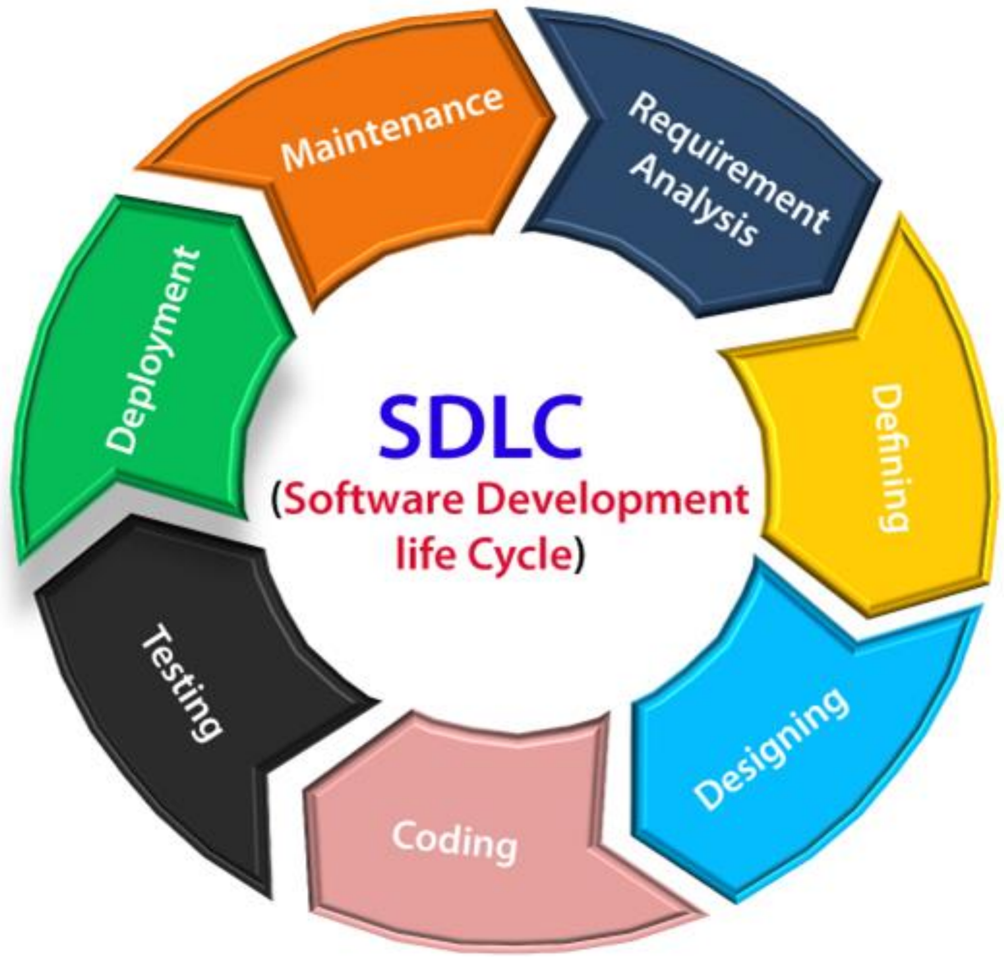
1. Over budget
2. Not delivering product on time
3. Poor Quality product
4. Software product not meeting the customer requirement

Software Development Life Cycle (SDLC)

What are the different phases of SDLC?

SDLC mainly has 7 stages

- Requirement Analysis
- Defining
- Designing
- Coding
- Testing
- Deployment
- Maintenance



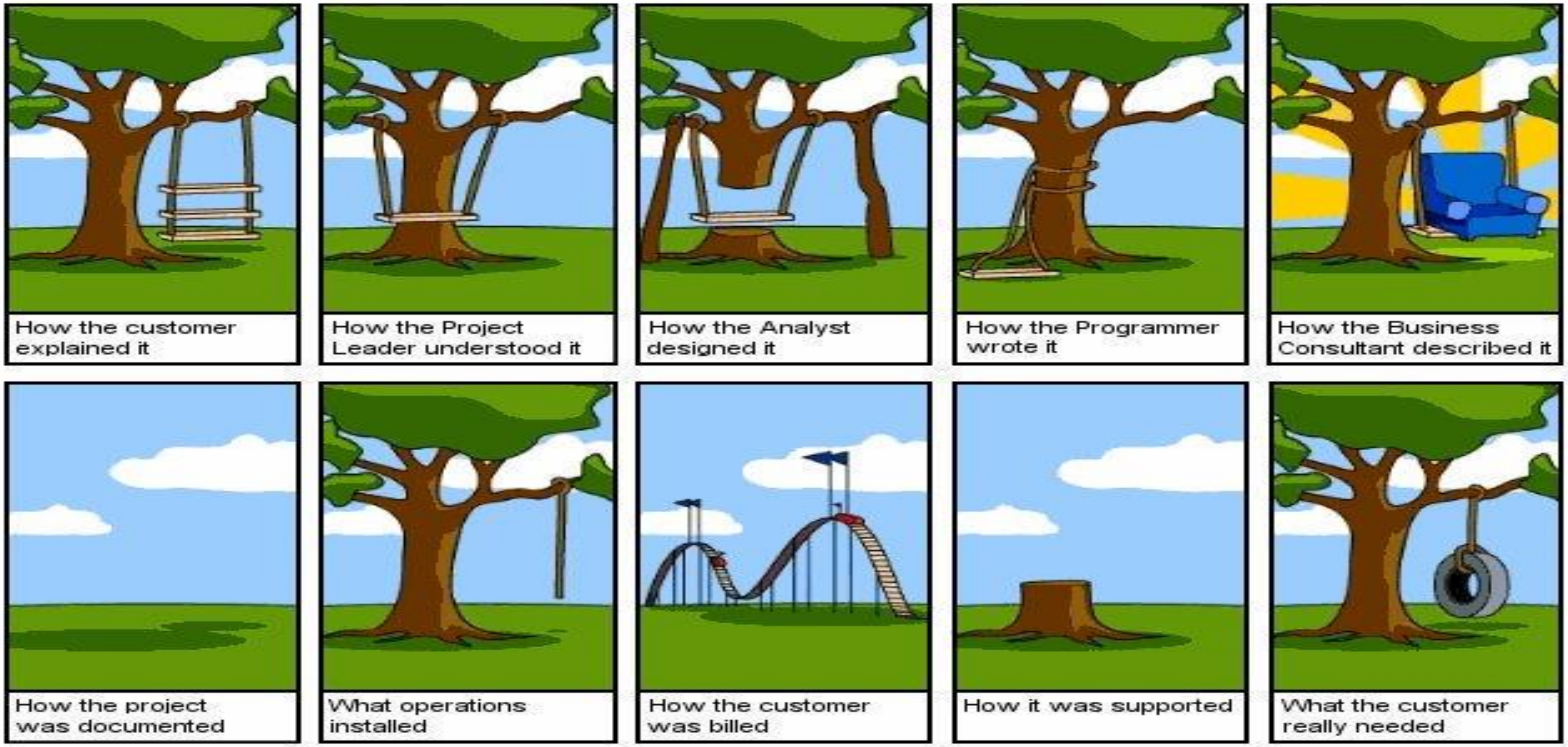
What is SDLC?

- Workflow process which defines activities performed at each stage of a software development project.
- Process used by the software industry to design, develop and test high-quality software’s.
- SDLC stands for Software Development Life Cycle is a **set of structured processes and defined phases** used **for the development of High-Quality Software with defined budget, schedule and resources for meeting or exceeding the Customers expectations.**
- SDLC covers all the phases of Software Development right from start to end.
- SDLC once defined, it is then documented and shared with the respective teams so as to keep track of the overall project.

Why do we need SDLC?

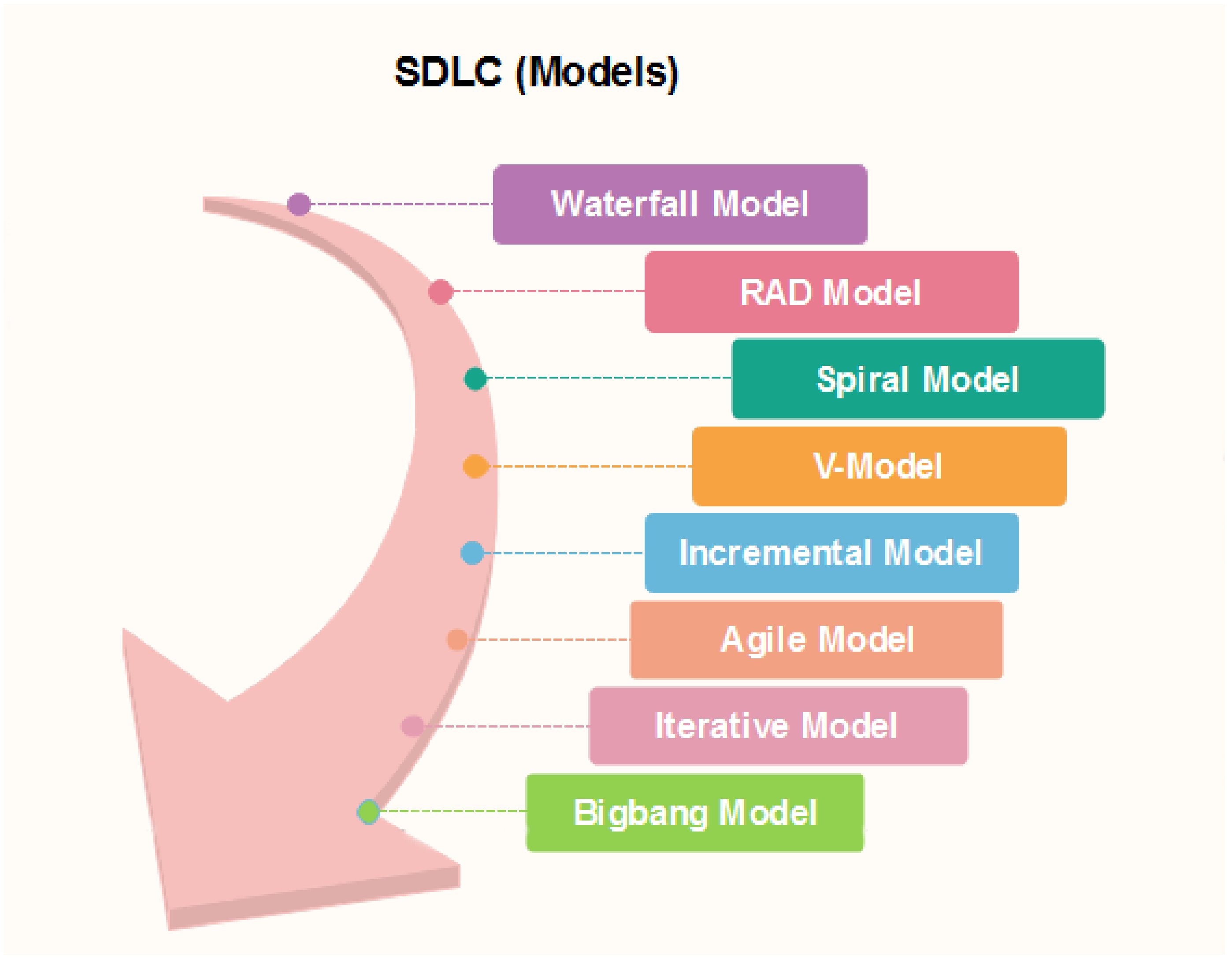
- Software Development involves various tasks that need to be completed in a well-defined and sequential manner for its successful completion.
- Input of one phase serves as an output for the following phases

For Example, A Software has to be developed and there are multiple teams working on the project, it is obvious that development cannot be done until requirement gathering is completed and testing cannot be done until development is completed, so to orchestrate the tasks of the team in a manner that each team is able to perform their tasks uninterrupted and with clear set of goals and inputs SDLC is required to be an essential part of the project.

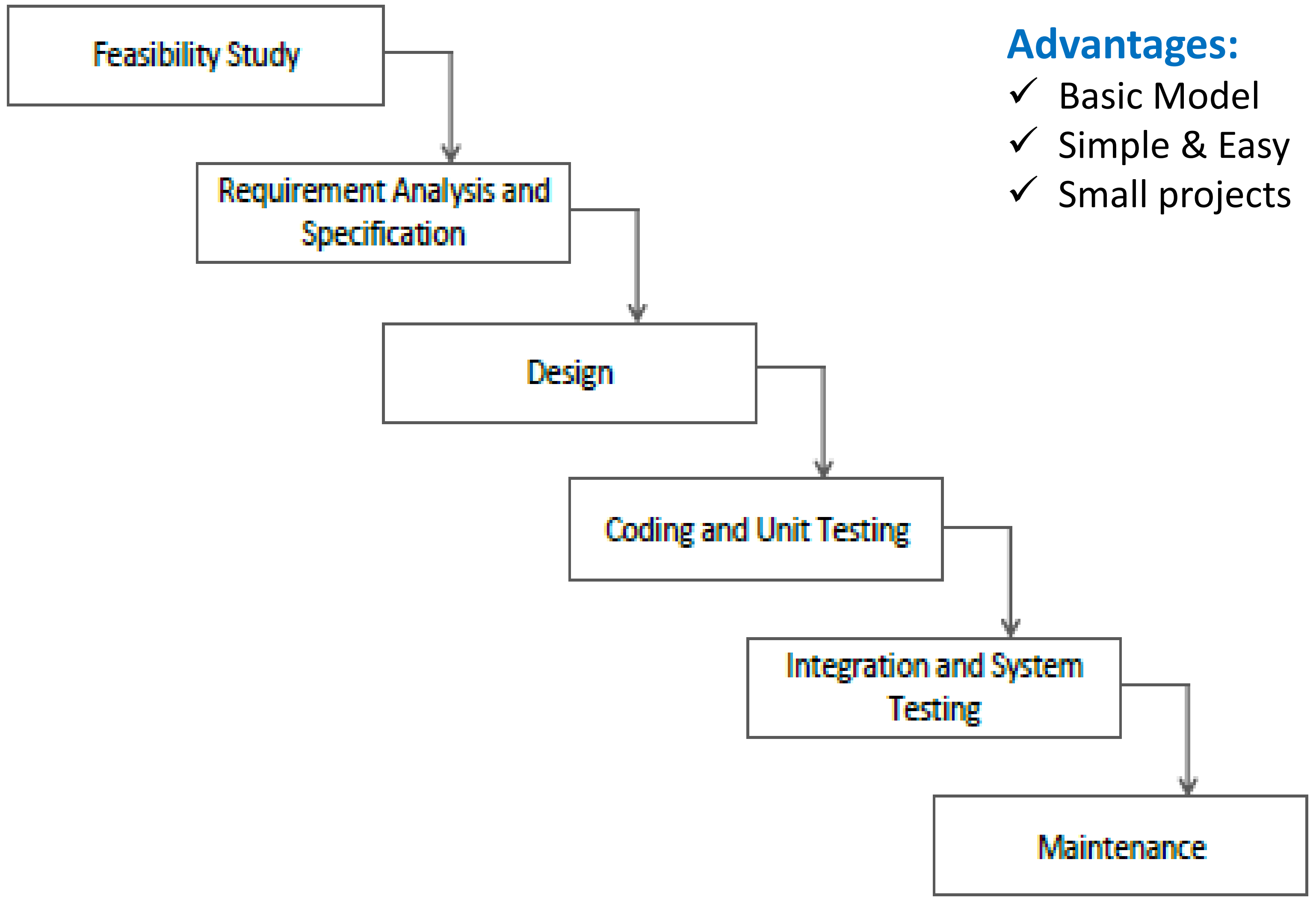

How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

The software development life cycle. (v/Project Cartoon)

Software Development Life Cycle (SDLC) MODELS



Classical Waterfall Model


Feasibility Study
Requirement Analysis and Specification
Design
Coding and Unit Testing
Integration and System Testing
Maintenance

Advantages:

- ✓ Basic Model
- ✓ Simple & Easy
- ✓ Small projects

Disadvantages:

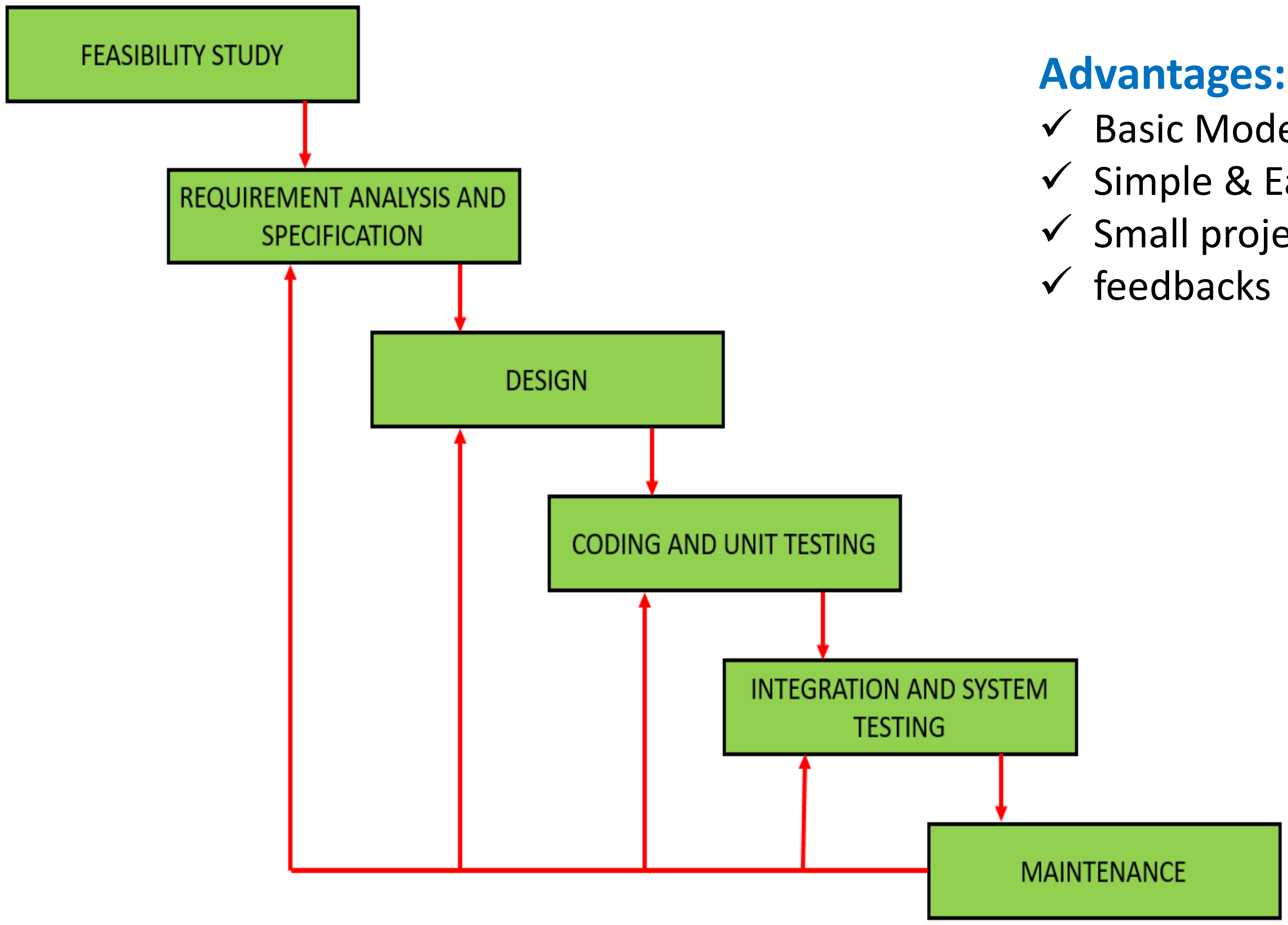
- ✓ No feedback
- ✓ No experiment
- ✓ No parallelism
- ✓ High risk
- ✓ 60% efforts in maintenance

The Waterfall model follows a linear approach to the software development life cycle (SDLC) in which progress flows in a single direction, like a cascading waterfall. In other words, each phase of the waterfall model must be completed before the next phase can begin, and there is no overlap or going back and forth between phases. The Waterfall model consists of the following phases:

- Requirements gathering: In this phase, the requirements for the software are defined and documented by meeting and discussion with the client.
- Design: In this phase, the system design is created based on the requirements gathered in the previous phase.
- Implementation: In this phase, the actual code for the software is written.
- Testing: In this phase, the software developed is tested to ensure that it meets the requirements defined in the first phase.
- Deployment: In this phase, the software is deployed and made available to users.
- Maintenance: In this phase, the software is maintained and updated as needed.

One of the main advantages of the Waterfall model is that it is simple, straightforward, and easy to understand, as each phase has well-defined goals and deliverables. However, it is inflexible to changes and may not be suitable for projects that require flexibility or rapid changes

Iterative Waterfall Model


FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Advantages:

- ✓ Basic Model
- ✓ Simple & Easy
- ✓ Small projects
- ✓ feedbacks

Disadvantages:

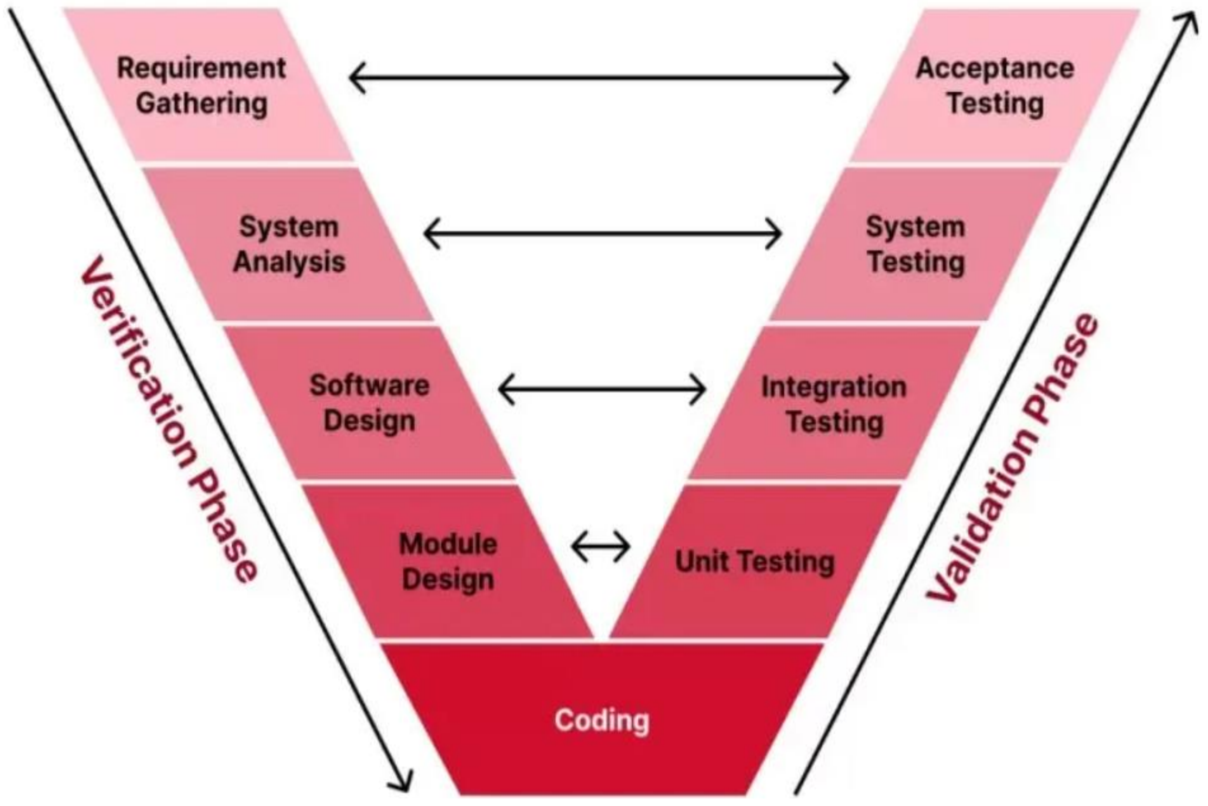
- ✓ No phase overlapping
- ✓ No intermediate delivery
- ✓ Rigidity (No Changes)
- ✓ Less customer interaction

The Iterative Waterfall Model is a software development approach that combines the sequential steps of the traditional Waterfall Model with the flexibility of iterative design. It allows for improvements and changes to be made at each stage of the development process, instead of waiting until the end of the project. The Iterative Waterfall Model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical Waterfall Model.

1. When errors are detected at some later phase, these feedback paths allow for correcting errors committed by programmers during some phase.
2. The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected in the later phases.
3. While the feasibility study phase often doesn't involve frequent feedback, the iterative nature of this model allows for revisiting earlier stages, including feasibility, if new requirements or changes arise during later phases.
4. It is good to detect errors in the same phase in which they are committed.
5. It reduces the effort and time required to correct the errors.
6. A real-life example could be building a new website for a small business.

V-Shaped Model

- Also known as Verification & Validation Model
- Extension of waterfall model
- Testing is associated with every phase of lifecycle.
- Verification phase (Requirement Analysis, System Design, Architecture Design, Module Design)
- Validation Phase (Unit Testing, Integration, System, Acceptance testing)



Advantages:

✓ Time saving
✓ Good understanding of project in the beginning
✓ Every component must be testable.
✓ Progress can be tracked easily.
✓ Proactive defect tracking.

Disadvantages:

✓ No feedback so less scope of changes
✓ Risk analysis not done
✓ Not good for big or object-oriented projects

2 Phases of the V Model

Verification Phases

The verification phase checks the product development process to ensure the team meets the criteria. Verification involves 5 steps that are business requirement analysis, system analysis, software architecture design, module design, and coding.

•**Business requirement analysis:** Business requirement analysis helps the team understand customer product requirements.

•**System analysis:** System engineers will use the user requirements document to analyze and interpret the proposed system's business requirements.

•**Software architecture design:** The team chooses the software architecture based on the list of modules, their brief functionality, interface relationships, dependencies, database tables, architectural diagrams, technological information, and more during software architecture design. This phase creates the integration testing model.

•**Module design:** The development team divides the system into tiny modules and specifies their low-level designs during module design.

•**Coding:** The development team chooses a programming language depending on design and product needs. Coding standards and procedures are in place, and the code will be thoroughly reviewed to ensure performance.

Validation Phases

The validation step uses dynamic analysis and testing to guarantee the program satisfies customer needs. This phase including comprises unit, integration, system, and acceptability testing.

•**Comprising unit:** The team creates and runs unit test plans to find code or unit issues. Program modules are tested to ensure they work properly when isolated from the rest of the code.

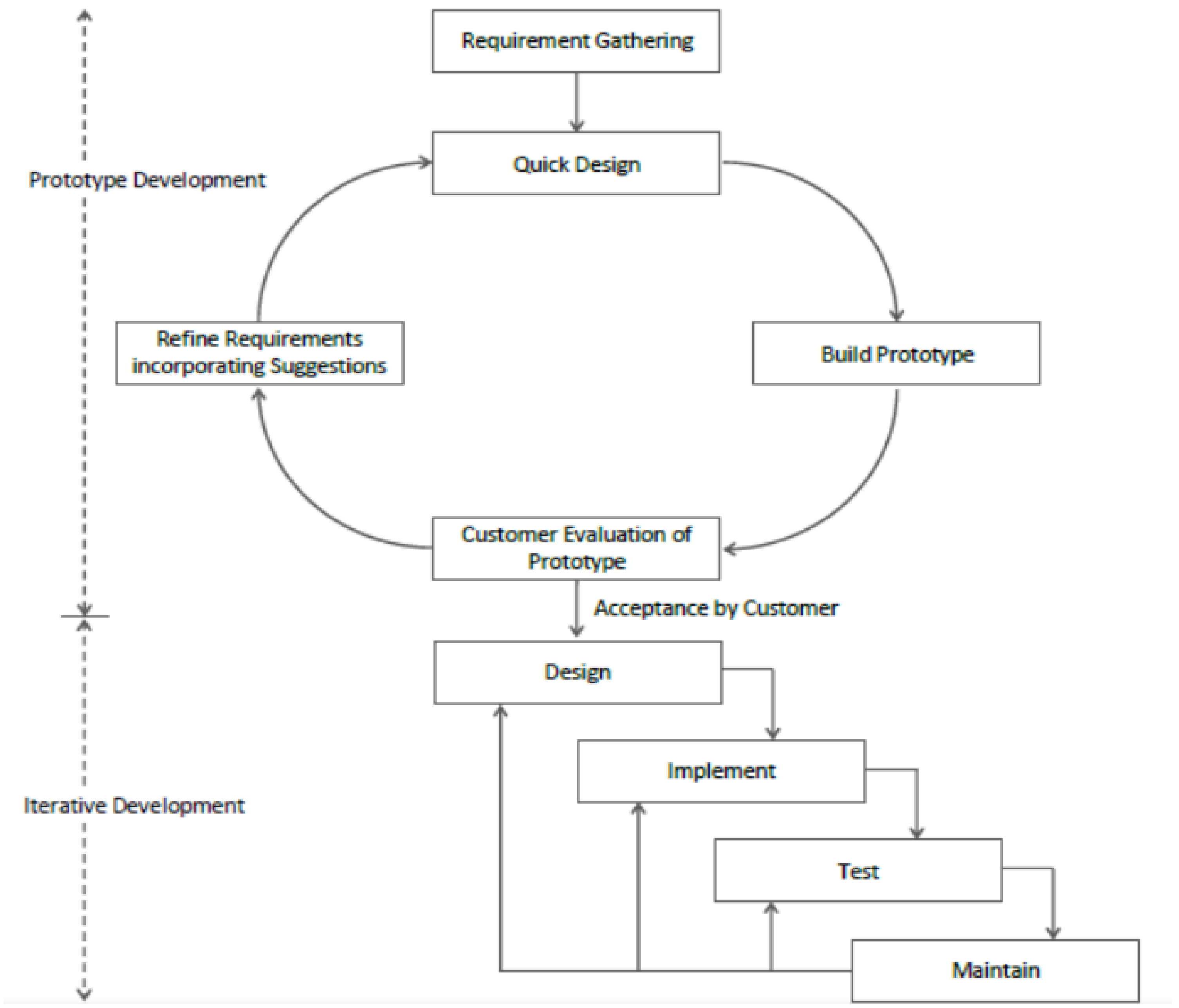
•**Integration:** Integration testing verifies that groups formed and tested independently may coexist and interact by performing integration test plans developed during architectural design.

•**System testing:** System test plans created by the client's business team during system design are executed during system testing. System testing ensures the team meets application developer objectives.

•**Acceptability testing:** Based on the business requirement analysis element of the V-model, acceptance testing entails evaluating the software product in the user environment to find compatibility concerns with other systems. In the real user environment, acceptance testing finds non-functional concerns like load and performance faults.

Prototyping Model

- Customer not clear with idea
- Throwaway model
- Good for technical and requirement risk
- Increase in cost of development



- In terms of Software Development, a prototype is an early sample or model of a product that is used for the purpose of demonstration of a solution.
- It is mainly used during the design and development phases to visually and functionally for giving demos to the client/ customer so as to demonstrate functional working of the product to better understand the developed product.
- The prototype can be used to gather feedback and make any necessary changes before the final product is developed.
- The prototype model in SDLC is an iterative process, where multiple prototypes are developed and refined until the final product meets the desired requirements.
- This process continue until the customer is satisfied with the system. Once a user is satisfied, the prototype is converted to the actual system with all considerations for quality and security.

Advantages of Prototype Model

- **Demo working model**: Customer get demo working model of actual product which help them to give a better understanding and attain a high level of satisfaction.
- **New requirement**: Based on the customer feedback, the requirements are redefined and the prototype is suitably modified till final approval.
- **Missing functionality**: can be easily established.
- **Easy error detection**: It saves time and cost in developing the prototype and enhance the quality of the final product.
- **Flexibility**: in the development phase.

Disadvantages of Prototype Model

- **Time-consuming:** As the prototype is being modified time to time according to customer requirement which usually increases the time of completion of the product.
- **Complexity**: Change in the requirement usually expand the scope of the product beyond its original plan and thus increase the complexity.
- **Poor Documentation**: Continuous changing of requirement can lead to poor documentation.
- **Unpredictability of no of iteration**: It is difficult to determine the no of iteration required before the prototype is finally accepted by the customer.
- **Confusion**: Customer can confuse between the actual product and prototype.

Incremental Model

- Large projects
- Module by module working
- Customer Interaction Maximum
- Early release product
- Flexible to changes

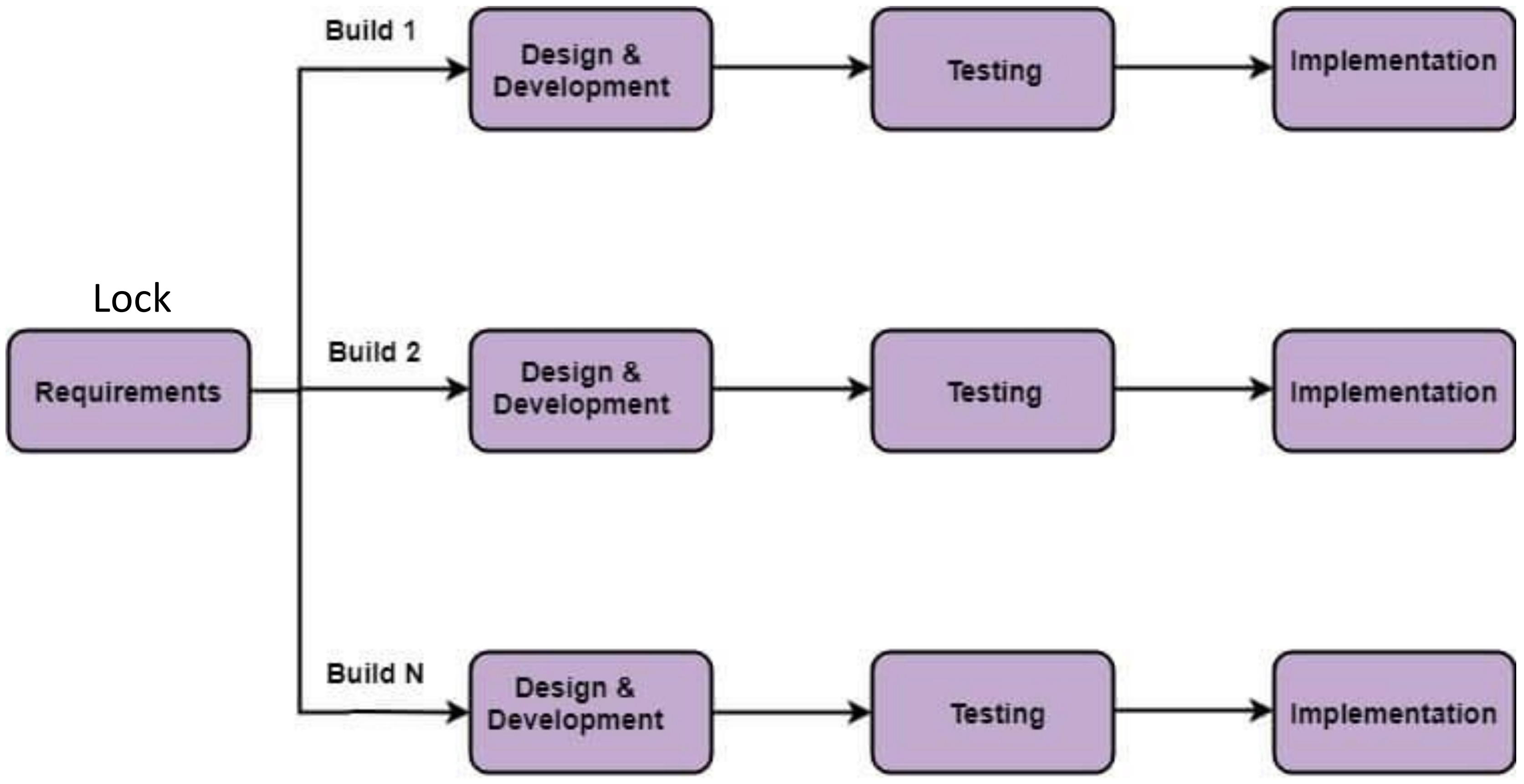


Fig: Incremental Model

When we use the Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

Evolutionary Model

- Combination of Iterative and Incremental model of SDLC.
- Incremental model first implement a few basic features and deliver to customer. Then build the next part and deliver it again and repeat this step until the desired system is fully realized.
- No long-term plans are made.
- Iterative model main advantage is its feedback process in every phase.
- Also known as "Design a little, build a little, test a little, deploy a little model".

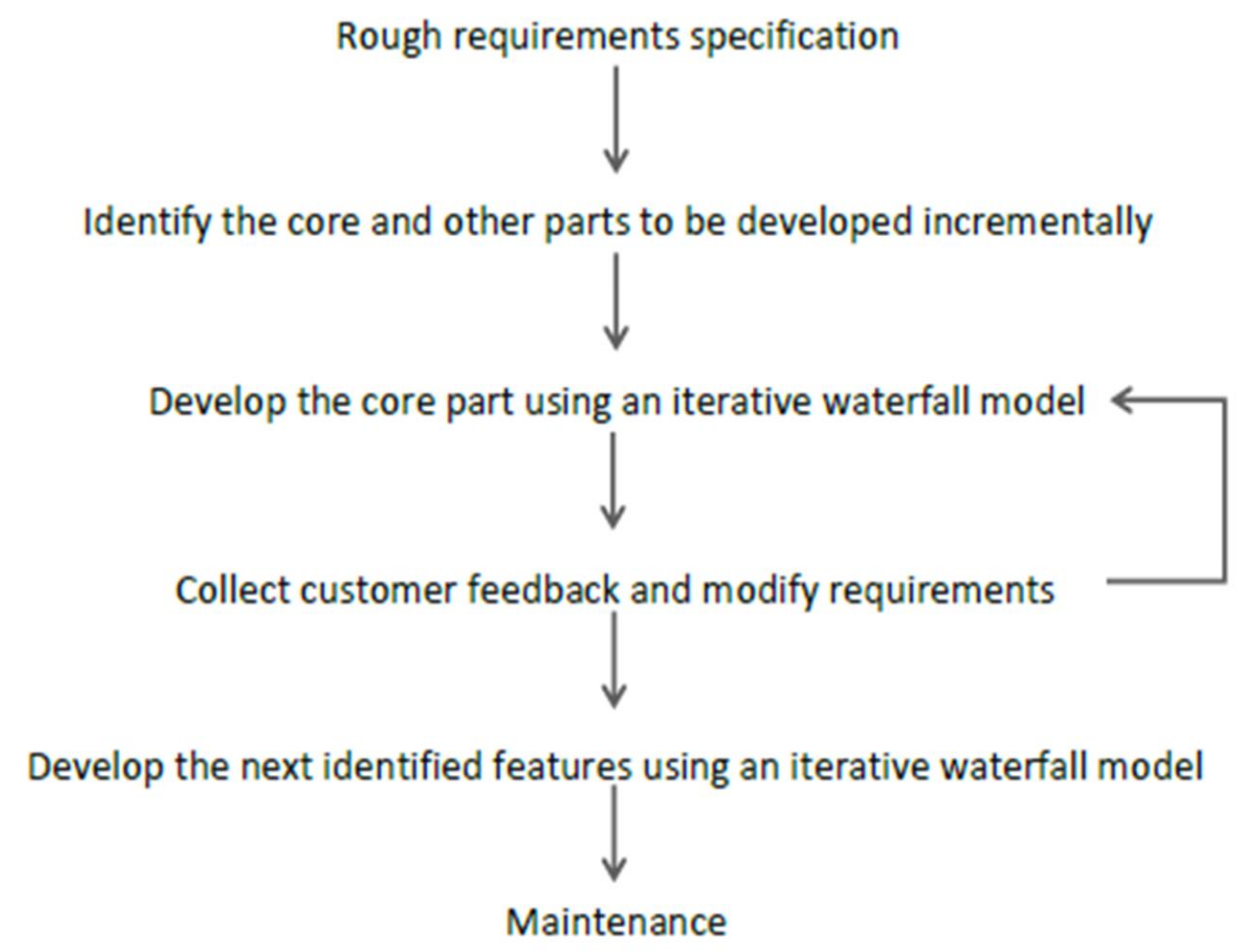

Rough requirements specification
Identify the core and other parts to be developed incrementally
Develop the core part using an iterative waterfall model
Collect customer feedback and modify requirements
Develop the next identified features using an iterative waterfall model
Maintenance

Figure 6: Evolutionary Model of Software Development

Advantages of Evolutionary Model

- Customer requirements are clearly specified
- Risk analysis is better
- It supports changing environment
- Initial operating time is less.

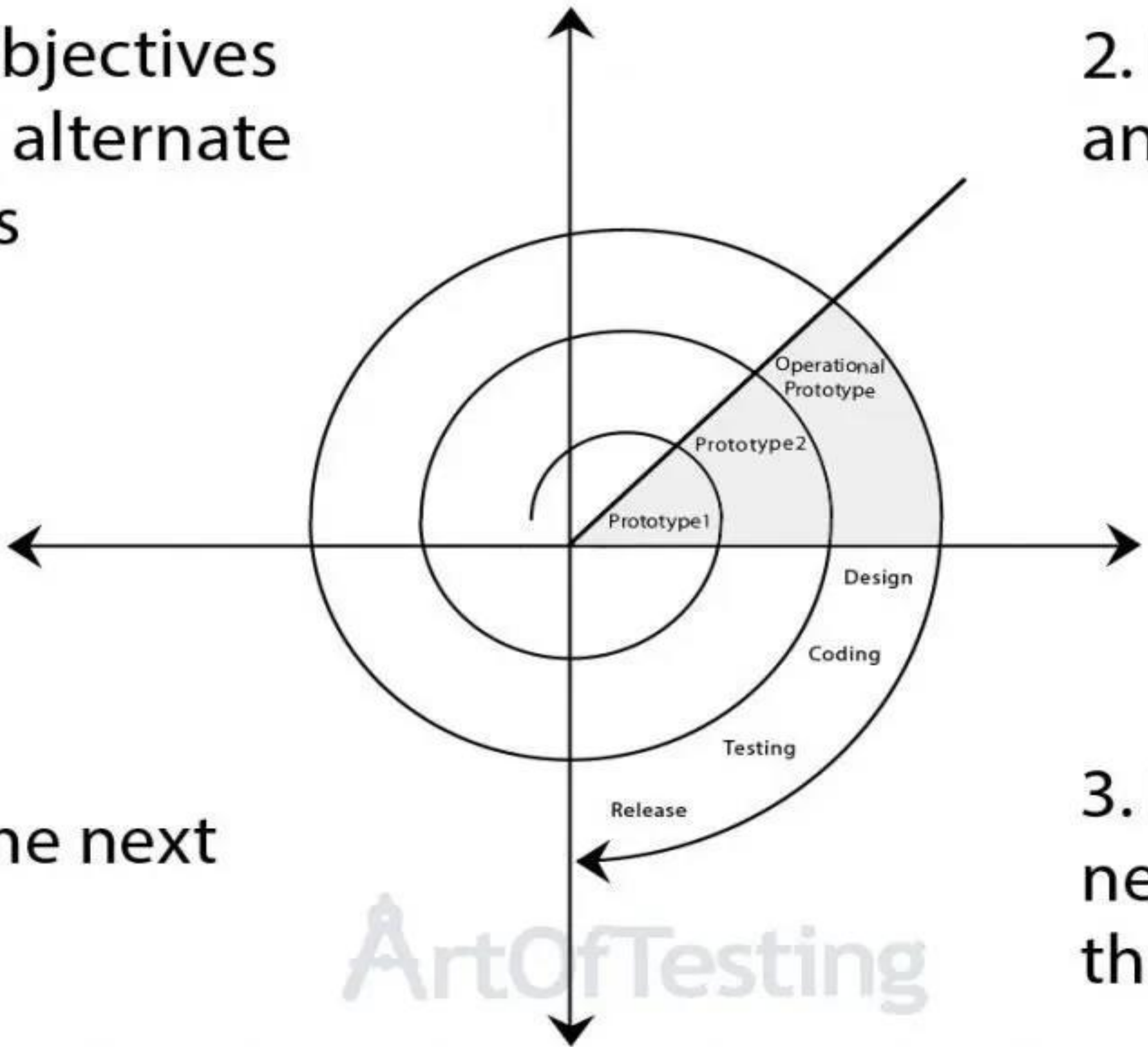
Disadvantages of Evolutionary Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Spiral Model

1. Plan objectives and find alternate solutions

2. Risk analysis and resolving



3. Develop the next version of the product

4. Plan the next phase

Spiral Model (Cont….)

- **Meta Model:** Combines elements of an iterative model, an incremental model and prototype model.
- ***Risk Handling***
- Radius of spiral = cost
- Angular dimension = progress

Disadvantages of Spiral Model

- Complex
- Expensive
- Too much risk analysis
- ***Time – overcome by RAD***

Advantages of Spiral Model

- Risk Handling
- Large & Complex Projects
- Flexible
- Implementation of change requests (CRs)
- Customer Satisfaction.

Spiral Model Phases

It has four stages or phases: The planning of objectives, risk analysis, engineering or development, and finally review. A project passes through all these stages repeatedly and the phases are known as a Spiral in the model.

Determine objectives and find alternate solutions – This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.

Risk Analysis and resolving – In this quadrant, all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.

Develop and test: This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.

Review and planning of the next phase – In this phase, the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

Spiral Model Advantages

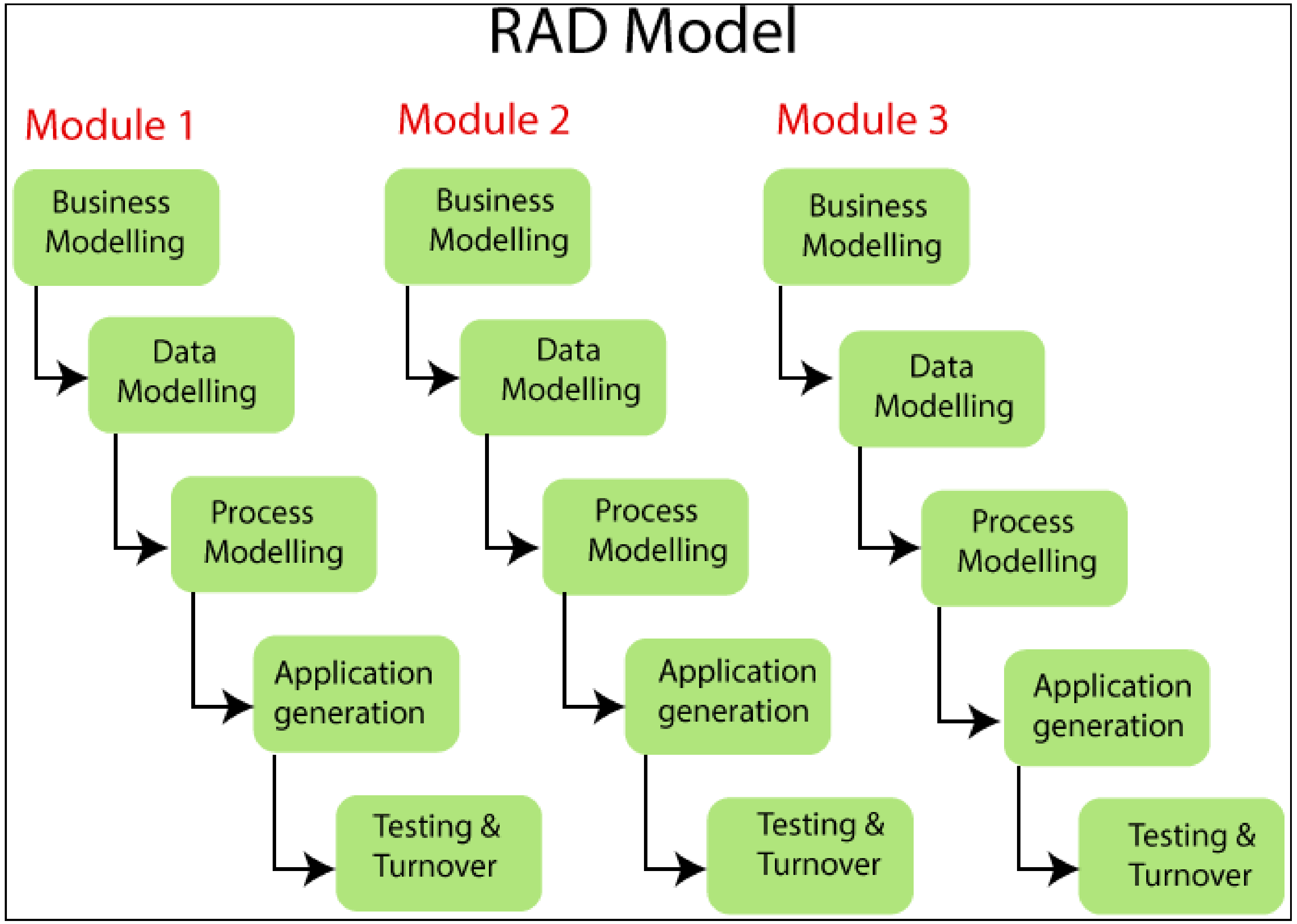
1. The spiral model is perfect for projects that are **large and complex** in nature as continuous prototyping and evaluation help in mitigating any risk.
2. Because of its **risk handling ability**, the model is best suited for projects which are very critical like software related to the health domain, space exploration, etc.
3. This model supports the client feedback and **implementation of change requests** (CRs) which is not possible in conventional models like a waterfall.
4. Since customer gets to see a prototype in each phase, so there are higher chances of customer satisfaction.

Spiral Model Disadvantages

1. Because of the prototype development and risk analysis in each phase, it is very **expensive and time taking**.
2. It is **not suitable for a simpler and smaller** project because of multiple phases.
3. It requires **more documentation** as compared to other models.
4. Project **deadlines can be missed** since the number of phases is unknown in the beginning and frequent prototyping and risk analysis can make things worse.

RAD Model – Rapid Application Development

- ✓ Quick Prototyping
- ✓ Iterative Development
- ✓ Incremental Releases
- ✓ User Involvement
- ✓ Time-Boxing
- ✓ Parallel Development



RAD Model (Cont….)

- First introduced by IBM in the 1980s was the rapid application development model. The use of effective production tools and strategies is a core aspect of this model.
- This model is best suited for projects with well-defined and understood requirements and limited project scope. By implementing the RAD process, development teams can create a fully functional system within a short timeframe.
- **SDLC RAD Model Phases**
- **Business Modelling –** During the Business Modeling phase, the primary task is to determine the information flow among various business functions by answering fundamental questions related to the source of data that drives the business process, where the generated data comes from, its destination, the individuals involved in the processing of this data, and other pertinent details. This step is crucial to establish a clear understanding of the business operations and the interdependence of its functions.
- **Data Modeling** is a phase where data collected from business modeling is transformed into a set of data objects, or entities, that are essential to support business activities. During this phase, the attributes of each entity are identified, and the relationship between these data objects is defined. This process helps to ensure that the data can be effectively utilized by the software application being developed.
- **Process Modeling** involves transforming the information objects defined in the data modeling phase to facilitate the required data flow for executing a business function. To enable the addition, modification, deletion, or retrieval of a data object, processing descriptions are generated.
- **Application Generation** In the application generation phase, developers employ automated tools to streamline the software construction process. These tools may leverage fourth generation language (4GL) techniques to improve efficiency and productivity.
- **Testing & Turnover –** The testing and turnover phase involves the evaluation of programming components, many of which have already undergone testing due to the emphasis on reuse in rapid application development (RAD). As a result, the overall testing time is reduced. However, any new components must be thoroughly tested, and all interfaces must be fully exercised to ensure optimal performance.

RAD Model (Cont….)

When the Rapid Application Development Model is Used?

- ✓ When there is a need to complete a project within a short timeframe of 2-3 months and modularization is required.
- ✓ When the technical risks associated with the project are limited and can be easily managed.
- ✓ The RAD model should only be utilized if the project budget permits the use of automated code generating tools.
- ✓ When the project requirements are clearly defined and well-understood.
- ✓ When there is a need to develop a modularized system within a timeframe of 2-3 months.

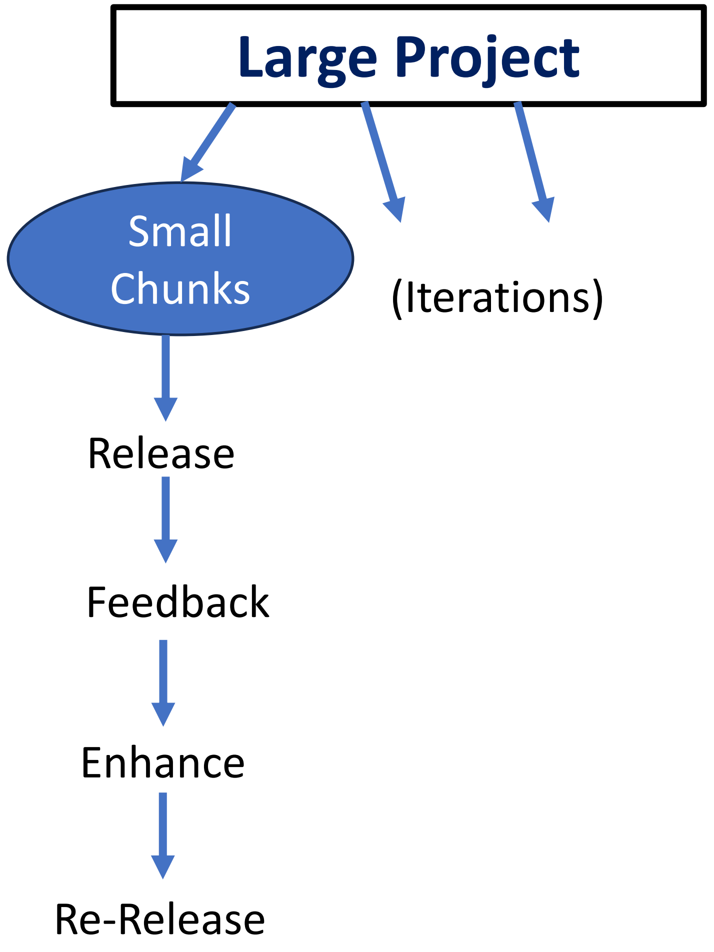
Advantages of RAD Model

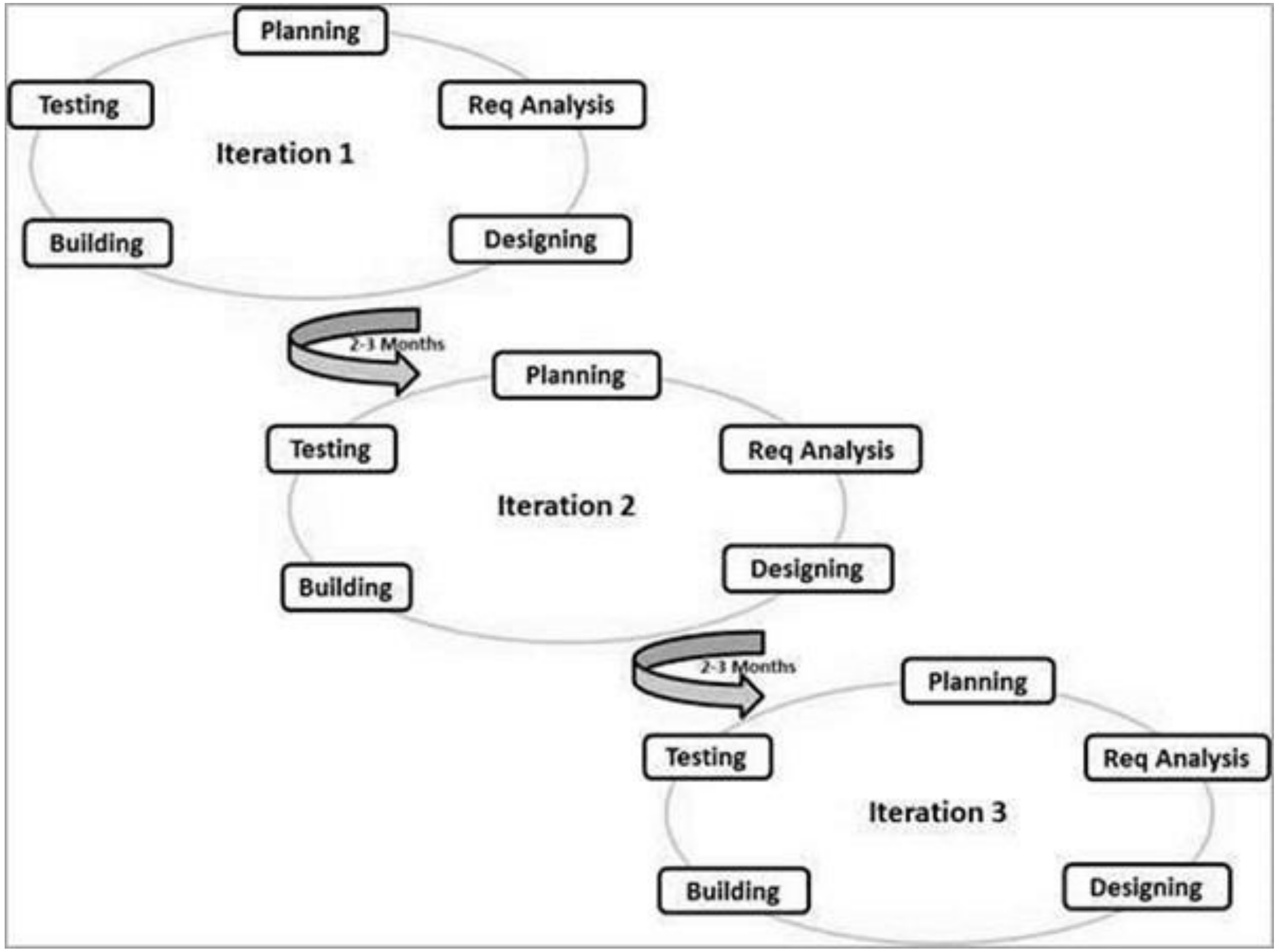
- ➢ The RAD model is highly flexible and can easily accommodate changes throughout the development process.
- ➢ Each phase of RAD focuses on delivering the highest priority functionalities to the customer.
- ➢ The RAD model emphasizes the reusability of features, which can increase efficiency and productivity in future development projects.
- ➢ Changes can be readily adopted and implemented in the RAD model.
- ➢ The RAD model significantly reduces development time compared to traditional models.

Disadvantages of RAD Model

- ➢ Requires highly skilled designers to effectively implement the process.
- ➢ Demands significant user involvement, which can be challenging to maintain throughout the project's lifecycle.
- ➢ May not be suitable for smaller projects
- ➢ Not all applications are compatible with the RAD model, as it is primarily suited for specific types of software projects.
- ➢ May not be appropriate for high technical risk projects as it may lead to quality and reliability issues.

Agile Model


Large Project
Small Chunks
(Iterations)
Release
Feedback
Enhance
Re-Release


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 2
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 3
Building
Designing

Advantages of Agile Model

- Frequent Delivery
- Face to Face communication with client
- changes
- Time

Disadvantages of Agile Model

- Less Documentation
- Maintainance

- The meaning of Agile is swift or versatile.“ Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.
- **Phases of Agile Model:**
    1. Requirements gathering
    2. Design the requirements
    3. Construction/ iteration
    4. Testing/ Quality assurance
    5. Deployment
    6. Feedback
- **Agile Testing Methods:**
    - Scrum
    - Crystal
    - Dynamic Software Development Method(DSDM)
    - Feature Driven Development(FDD)
    - Lean Software Development
    - eXtreme Programming(XP)