SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

CSE 316 – Software Design With UML

Course Coordinator : Dr. G. Pradeep

UNIT-I **Introduction to Software Development Process**

Part – I

Software Processes

Conventional software process models, Rational Unified Process, and Agile

Objectives

- To introduce software process models
- To describe three generic process models and when they may be used
- To describe outline process models for requirements engineering, software development, testing and evolution
- To explain the Rational Unified Process model
- To explain the Software Characteristics and Quality

Topics covered

- Software process models
- Project and its Characteristics
- Process activities
- The Rational Unified Process and Agile Process Model

The software process

- A software process model is an abstract representation of a process. It presents a description of a process. A process is a collection of activities, actions, and tasks that are performed when some work product is to be created.
- A structured set of activities required to develop a software system
  - Specification;
  - Design;
  - Validation;
  - Evolution.

THE SOFTWARE PROCESS

A generic process framework for software engineering encompasses five activities:

- **Communication**

  important to communicate and collaborate with the customer and other stakeholders. The intent is to understand stakeholders' objectives for the project and to gather requirements

- **Planning**

  planning activity creates a "map" that helps, guide .The map—called a software project plan—defines the software engineering work by describing the technical tasks to be conducted

- **Modeling**

  creating models to better understand software requirements and the design that will achieve requirements.

- **Construction**

  This activity combines code generation (either manual or automated) and the testing that is required to uncover errors in the code.

- **Deployment**

  The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation

Software engineering process framework activities are complemented by a number of umbrella activities

- **Software project tracking and control**

  allows the software team to assess progress against the project plan and take any necessary action to maintain the schedule.

- **Risk management**

  assesses risks that may affect the outcome of the project or the quality of the product.

- **Technical reviews**

  assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.

- **Software quality assurance**

  defines and conducts the activities required to ensure software quality**.**

- **Measurement**

  Measurement—defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs

- **Software configuration management**

  manages the effects of change throughout the software process
- **Reusability management**

  defines criteria for work product reuse (including software components) and establishes mechanisms to achieve reusable components.
- **Work product preparation and production**

  encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

What is a Project?

A project is a temporary venture that exists to produce a defined outcome.

Each project will have agreed and unique objectives as well as its own **project plan, budget, timescale, deliverables** and **tasks**.

A project may involve people from different teams within an organization who are brought together to accomplish a specific goal.

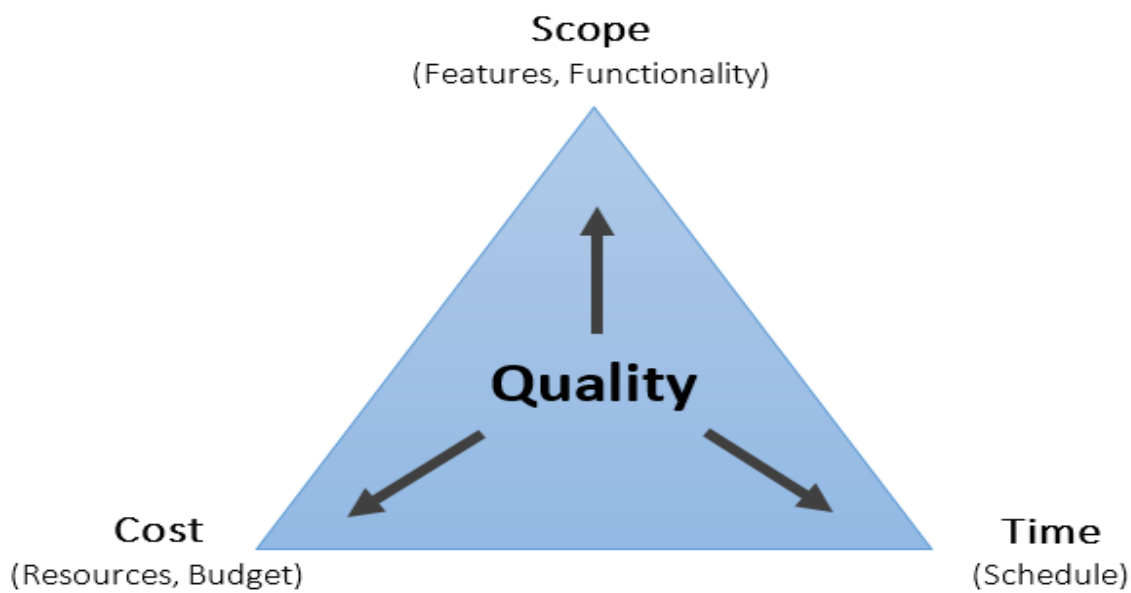
A project must have a **scope, fixed timeline, project plan, and resources.**

On the other hand, a process involves a **series of related tasks** that teams must carry out to achieve a result.

A **project is something that generally has not been done** before within an organization, whereas a **process is regularly repeated**.

Key components of project management

- **Time** – the intended duration of the work
- **Cost** – the budget allocated for the work
- **Scope** – what innovations or changes will be delivered by the project
- **Quality** – the standard of the outcome of the project.



Effective software project management focuses on the four P's:

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

Requirement Gathering

- Gathering needs from stakeholders and analyzing them to understand the **project's scope and goals**. It decides the project's quality in terms of **ease of use, performance, functionality, portability, etc**.
- The **Software Requirement Specification** (SRS) document includes a thorough statement of **functional requirement** of the software system, and **limitations on the software system**.

- **Design Phase**

  The design phase transforms the prerequisite specification into a software representation that may be evaluated prior to the coding phase.

- **Implementation**

  It contains coding the software in accordance with the design necessitates. In this phase, unit testing is also utilized to check that every component of the software is functioning properly.

- **Integration and System Testing**

  It combines the software modules in a systematic and planned way. The integration of the components cannot be completed in a single step but rather requires several iterations.

- **Deployment and Maintenance**

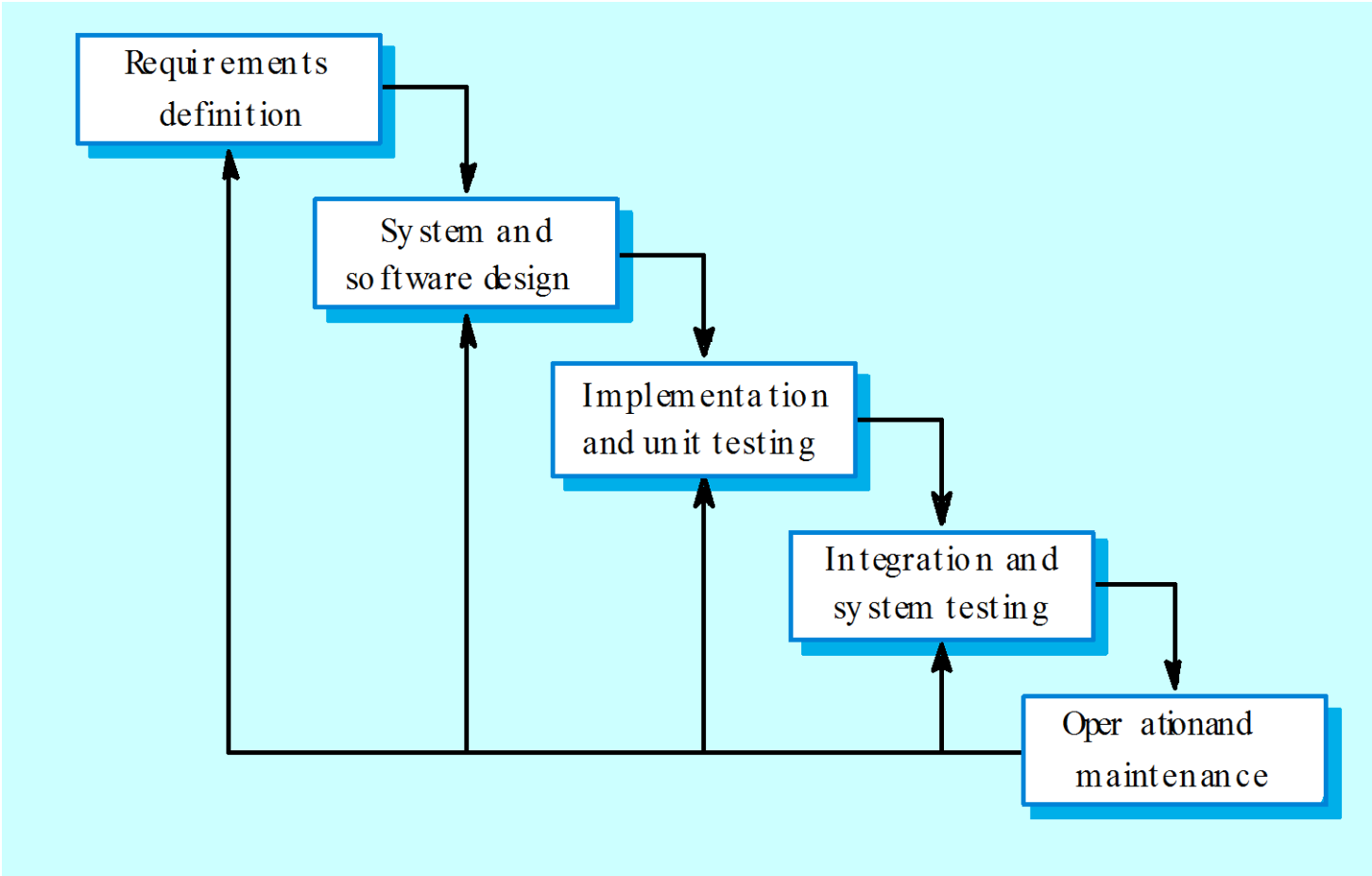
  Once the software has been tested and authorized, it is deployed to the production environment.

  After the software is delivered, it is analyzed for a period of time to find and resolve the errors that do not appear in the starting phases of software, which is known as maintenance.

Generic software process models

- The waterfall model
  - Separate and distinct phases of specification and development.
- Evolutionary development
  - Specification, development and validation are interleaved.
- Component-based software engineering
  - The system is assembled from existing components.
- There are many variants of these models e.g. formal development where a waterfall-like process is used but the specification is a formal specification that is refined through several stages to an implementable design.

Waterfall model



Waterfall model phases

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance
- The main drawback of the waterfall model is the difficulty of **accommodating change after the process is underway. One phase has to be complete before moving onto the next phase**.

Advantages

- The Waterfall model is simple to understand and use, and it is a better choice for software development projects.
- It functions well for smaller tasks and projects with well-defined requirements.
- It is a dependable and predictable technique for developing software.
- It offers a clear picture of the end product's appearance and functionality.
- It is a sequential, linear strategy that makes it simpler to estimate the time and resources needed for every project phase.

Disadvantages

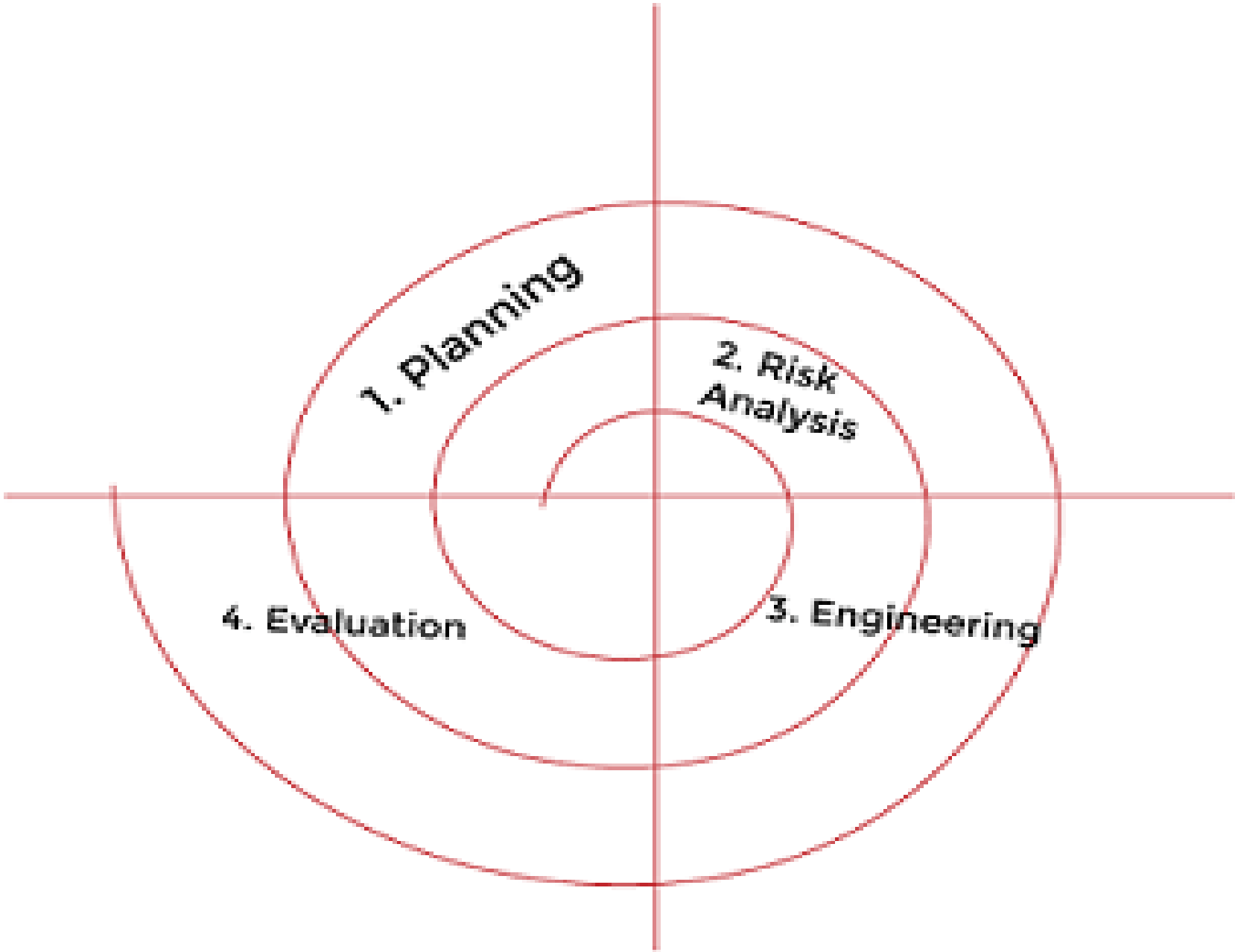
- It does not enable end-user feedback.
- A new phase starts only after the prior phase has been finished. However, it cannot be kept in real-world projects. Phases may overlap to enhance efficiency and decrease costs.
- It is unsuitable for complicated projects because its linear and sequential nature complicates handling numerous dependencies and interrelated components.
- Testing is usually done toward the end of the development process in the Waterfall Model. It means that defects cannot be found until late in the development process, which may be costly and time-consuming to resolve.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

Spiral development

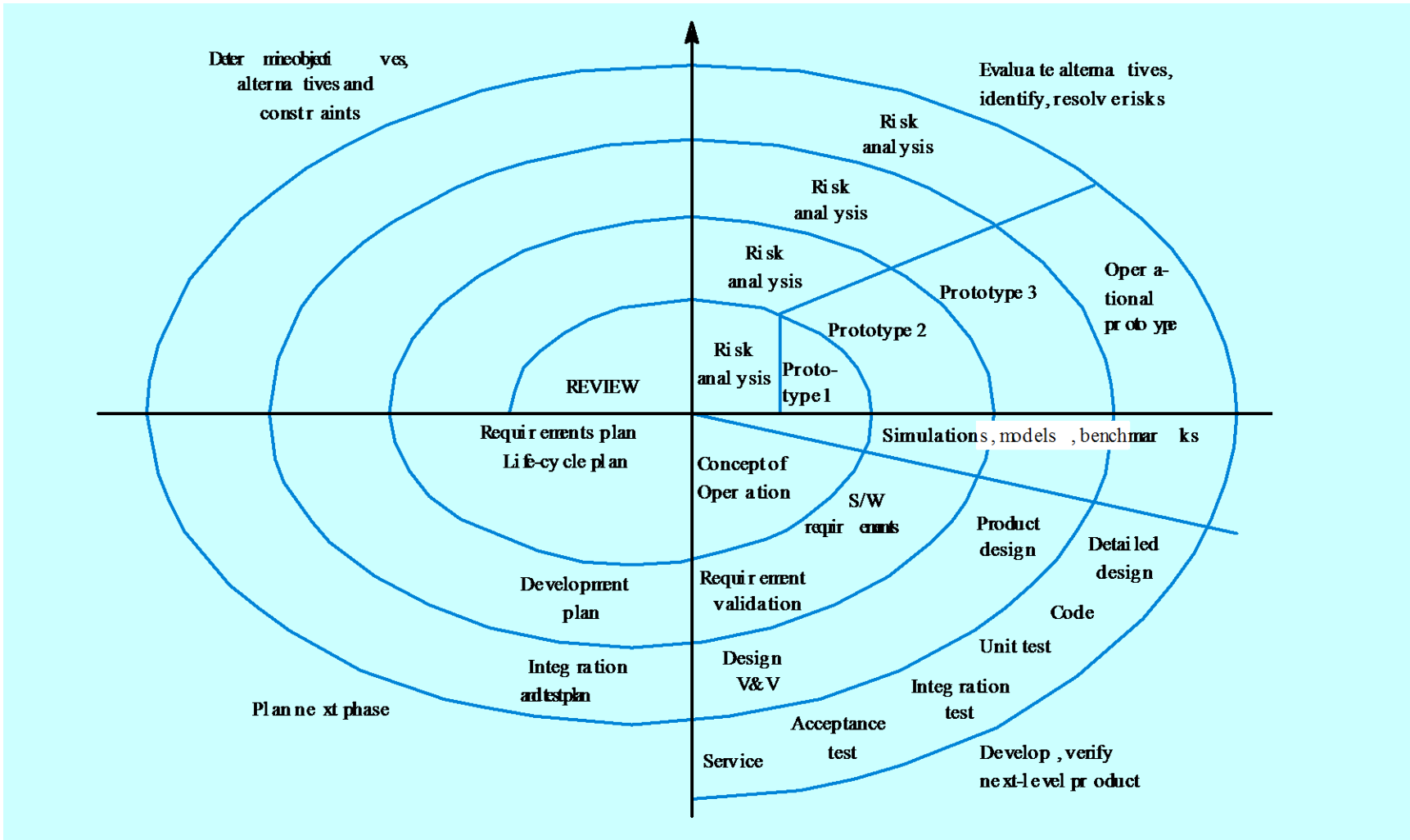
- Evolutionary method
- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.



Spiral Model

- Identifying and understanding requirements,
- performing risk analysis,
- building the prototype and
- evaluation of the software's performance.

Spiral model of the software process

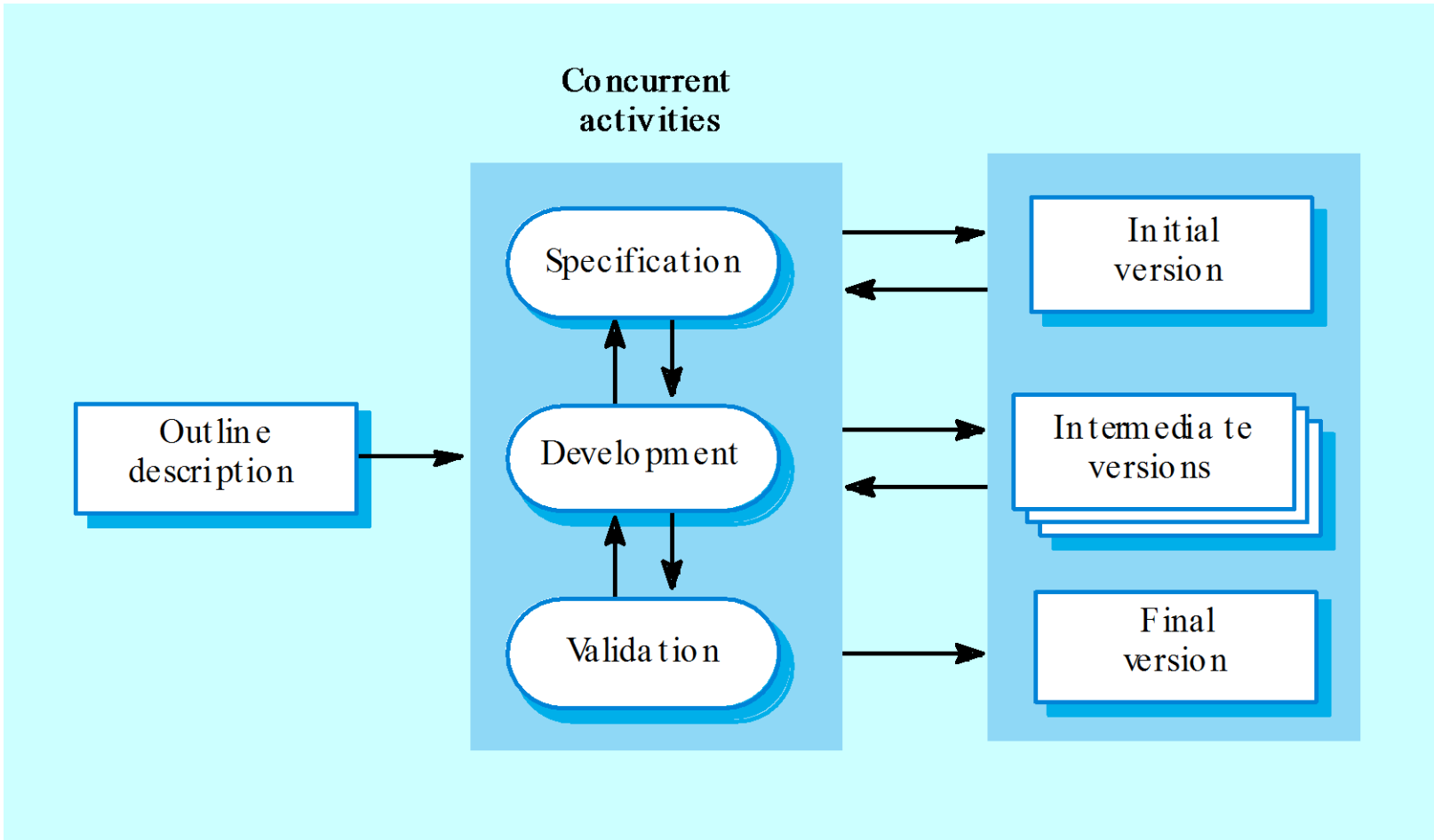


Features	Waterfall Model	Spiral Model
Definition	It is a software development process model that follows a linear sequential flow.	It is an evolutionary method of SDLC, and it is a combination of the prototype model and the waterfall model.
Ease to learn	It is simple and easy.	It is more complex.
Working	It works in a sequential method.	It works in an evolutionary method.
Suitable for	It is suitable for small projects with clear goals.	It is suitable for larger, more complicated projects.
Risk	There is a high risk than in other models.	There is a low risk than in other models.
Cost	It is less costly than the spiral model.	It is very expensive than the waterfall model.

Evolutionary development

- **Exploratory development**
  - Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.
- **Throw-away prototyping**
  - Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

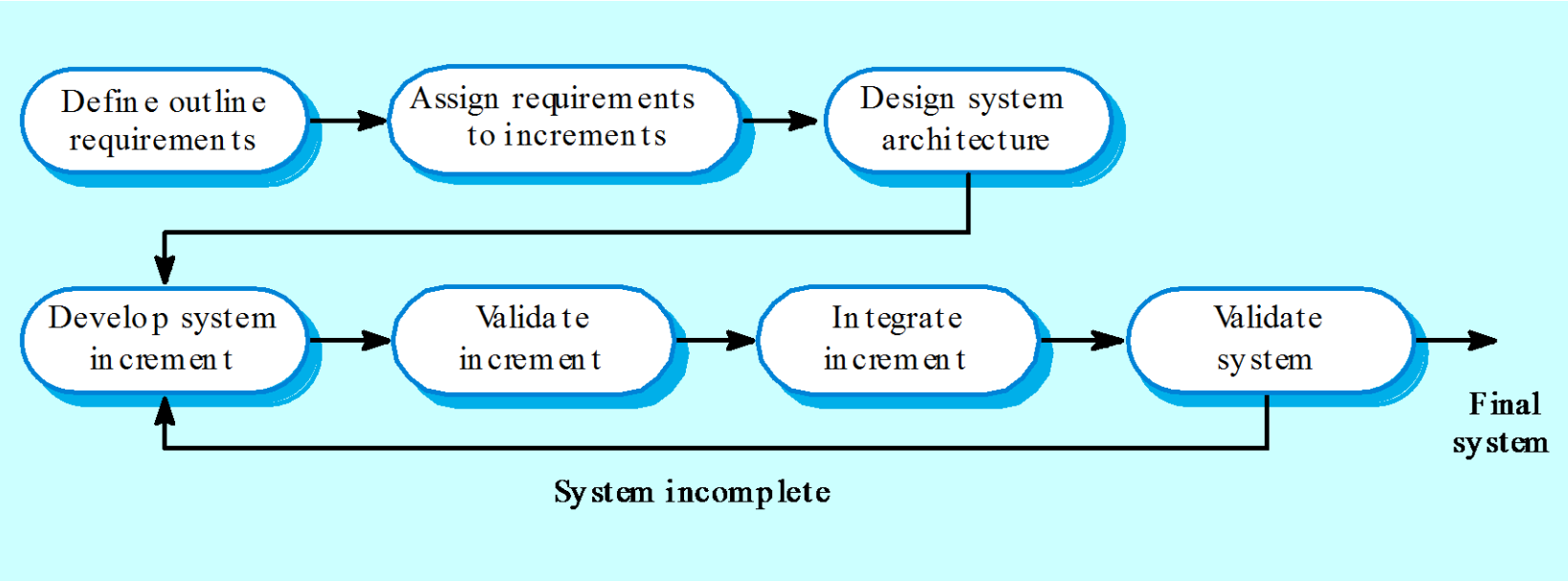
Evolutionary development



Evolutionary development

- Problems
  - Lack of process visibility;
  - Systems are often poorly structured;
  - Special skills (e.g. in languages for rapid prototyping) may be required.
- Applicability
  - For small or medium-size interactive systems;
  - For parts of large systems (e.g. the user interface);
  - For short-lifetime systems.

Incremental development



Incremental Development

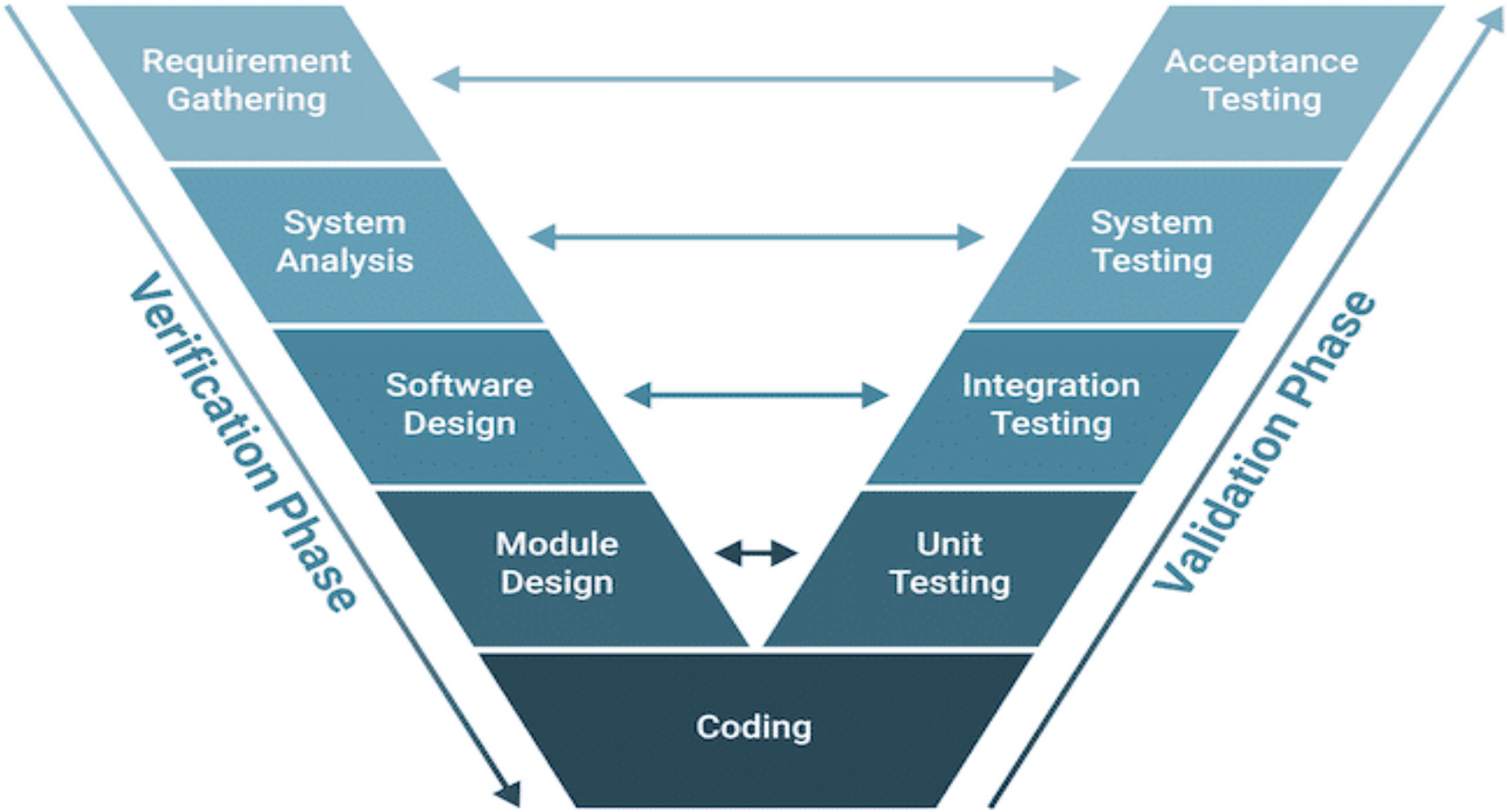
- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental development advantages

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

V-Model

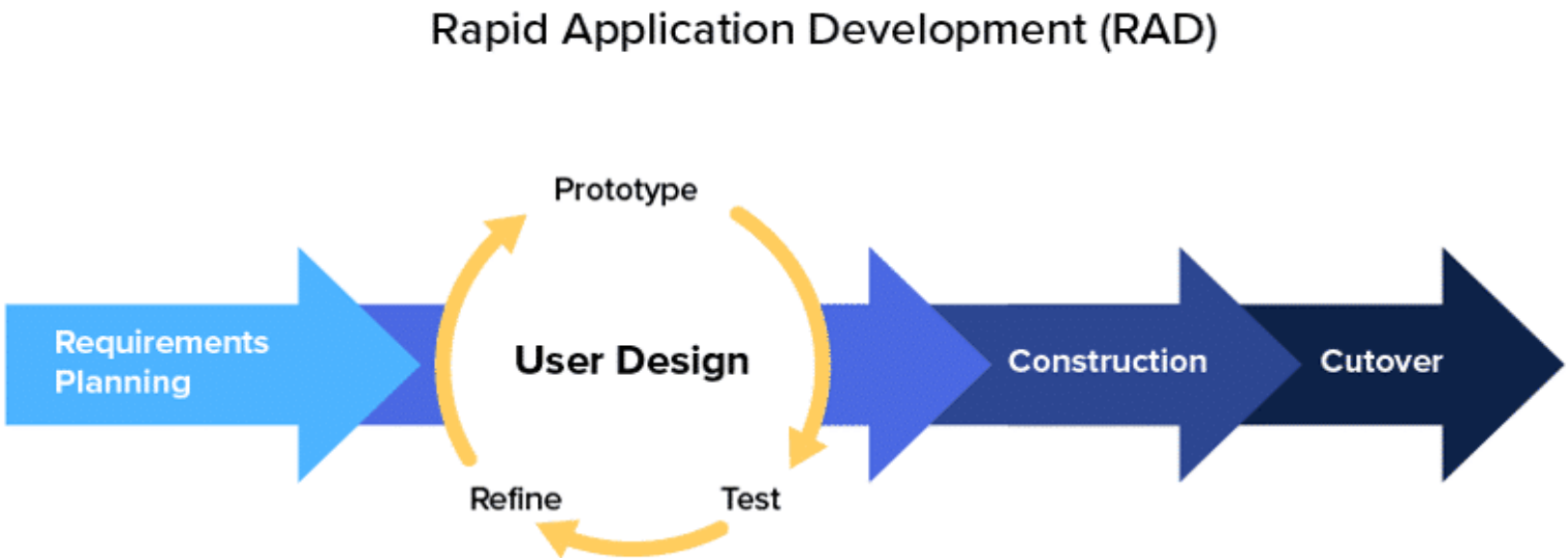
- Referred to as the **Verification and Validation** model
- offers a structured approach to software development.
- It builds upon the Waterfall model by introducing testing phases corresponding to each development stage, shaping a V-shaped process.
- This methodology underscores the significance of testing at every step, from requirements gathering to maintenance, to ensure quality and reliability.


Requirement Gathering
System Analysis
Software Design
Module Design
Coding
Unit Testing
Integration Testing
System Testing
Acceptance Testing
Verification Phase
Validation Phase

Waterfall Vs V Model

- Waterfall model, which reserves testing for post-development stages,
- V-Model incorporates testing throughout the process, thereby improving product quality.
- **However, this thorough testing approach can extend project timelines and inflate costs**.

RAD Model


Rapid Application Development (RAD)
Prototype
Requirements Planning
User Design
Construction
Cutover
Refine
Test

RAD Model

- Designed for speed and adaptability.
- It emphasizes swift prototyping over extensive upfront planning, facilitating rapid progress.
- In RAD, teams concurrently tackle various components, enabling the early deployment of essential features.
- Through iterative cycles, modifications and improvements are seamlessly integrated.

Software Crisis

Software crisis refers to **the challenges faced in developing efficient and useful computer programs due to** increasing complexity and demands.

- Limited Reusability
- Complexity Overhead
- Absence of Standards
- Lack of formalization
- Fail to meet performance metrics

Causes of Software Crisis

- Project running over budget
- Project running over time
- Inefficient Software
- Software of Low Quality
- Software did not match the user requirements
- Software never delivered
- Lack of Skilled Personnel: The demand for skilled software professionals often exceeds supply, impacting the quality and speed of software development.
- Code difficult to maintain

Factors Contributing to Software Crisis

- **Poor Project Management**: Ineffective management practices lead to mismanaged resources, unclear objectives, and unrealistic timelines.
- **Inadequate Testing:** Software with insufficient testing often has bugs and performance issues, affecting usability and reliability.
- **Hardware Limitations**: Sometimes, software does not align well with existing hardware capabilities, leading to performance issues.

Solution of Software Crisis

- **Adopting Agile Methodologies**: Agile focuses on iterative development, adapting to changing requirements effectively.
- **Enhanced Skill Development:** Continuous learning and development can equip professionals with up-to-date skills and knowledge.
- **Effective Project Management:** Robust project management ensures better planning, resource allocation, and timeline management.

SOFTWARE ENGINEERING PRACTICE

- **The Essence of Practice**
  - *Understand the problem* (communication and analysis).
  - *Plan a solution* (modeling and software design).
  - *Carry out the plan* (code generation).
  - *Examine the result for accuracy* (testing and quality assurance).
- **General Principles**

1.The Reason It All Exists

A software system exists for one reason: *to provide value to its users* "Does this add real value to the system?" If the answer is "no," don't do it.

2. ***KISS (Keep It Simple, Stupid!)***

   All design should be as simple as possible, but no simpler. This facilitates having a more easily understood and easily maintained system.

3. ***Maintain the Vision***

   A clear vision is essential to the success of a software project

4. *What You Produce, Others Will Consume*

   always specify, design, and implement knowing someone else will have to understand what you are doing

5. *Be Open to the Future*

   system with a long lifetime has more value. systems must be ready to adapt to these and other changes. *Never design yourself into a corner. Always ask "what if," and prepare for all possible answers by creating systems that solve the general problem.*

6. *Plan Ahead for Reuse*

   Reuse saves time and effort. Achieving a high level of reuse is arguably the hardest goal to accomplish in developing a software system. *Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated.*

7. *Think!*

   Placing clear, complete thought before action almost always produces better results. When you think about something, you are more likely to do it right. You also gain knowledge about how to do it right again

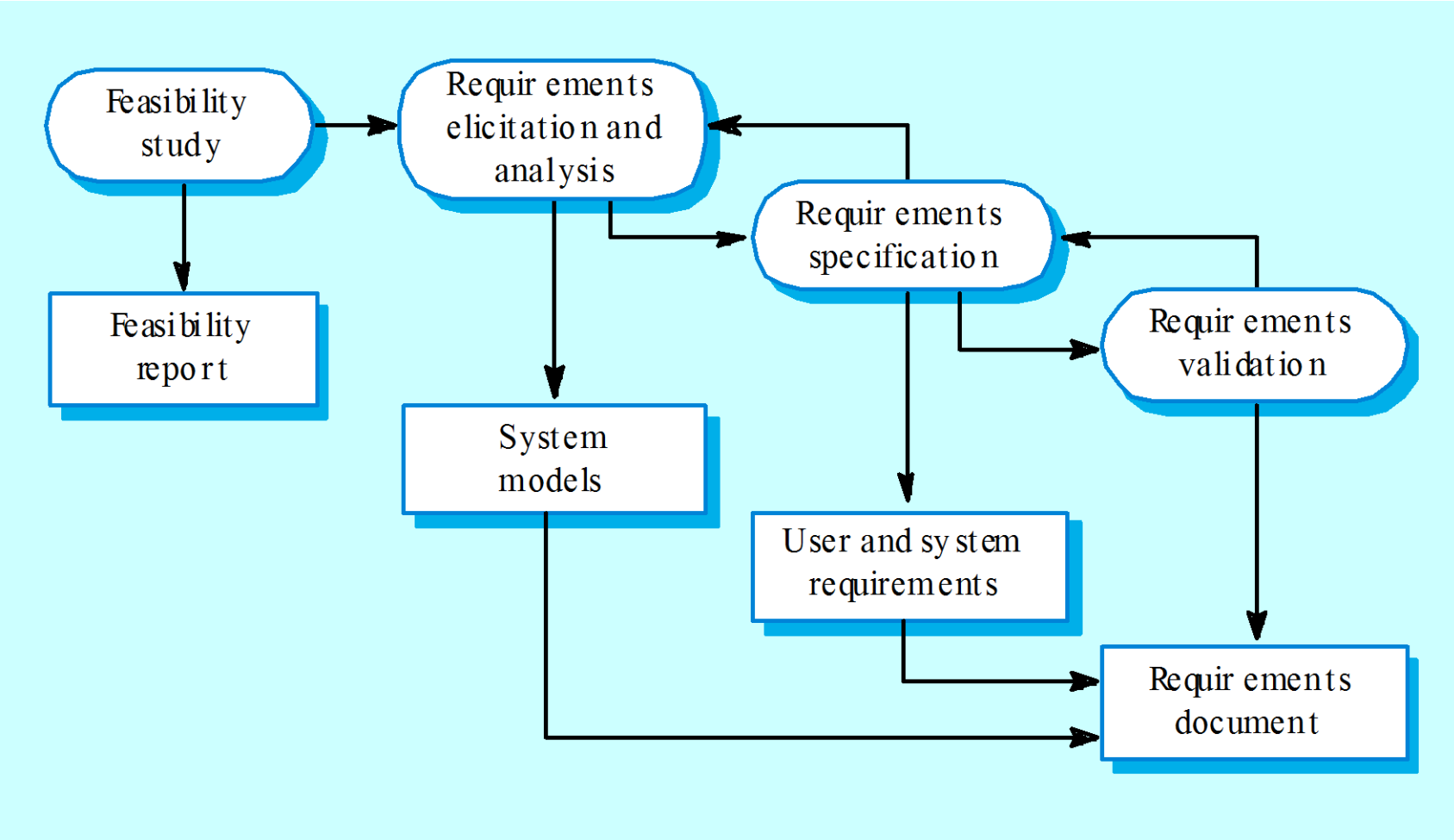
Process activities

- Software specification
- Software design and implementation
- Software validation
- Software evolution

Software specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - Feasibility study;
  - Requirements elicitation and analysis;
  - Requirements specification;
  - Requirements validation.

The requirements engineering process



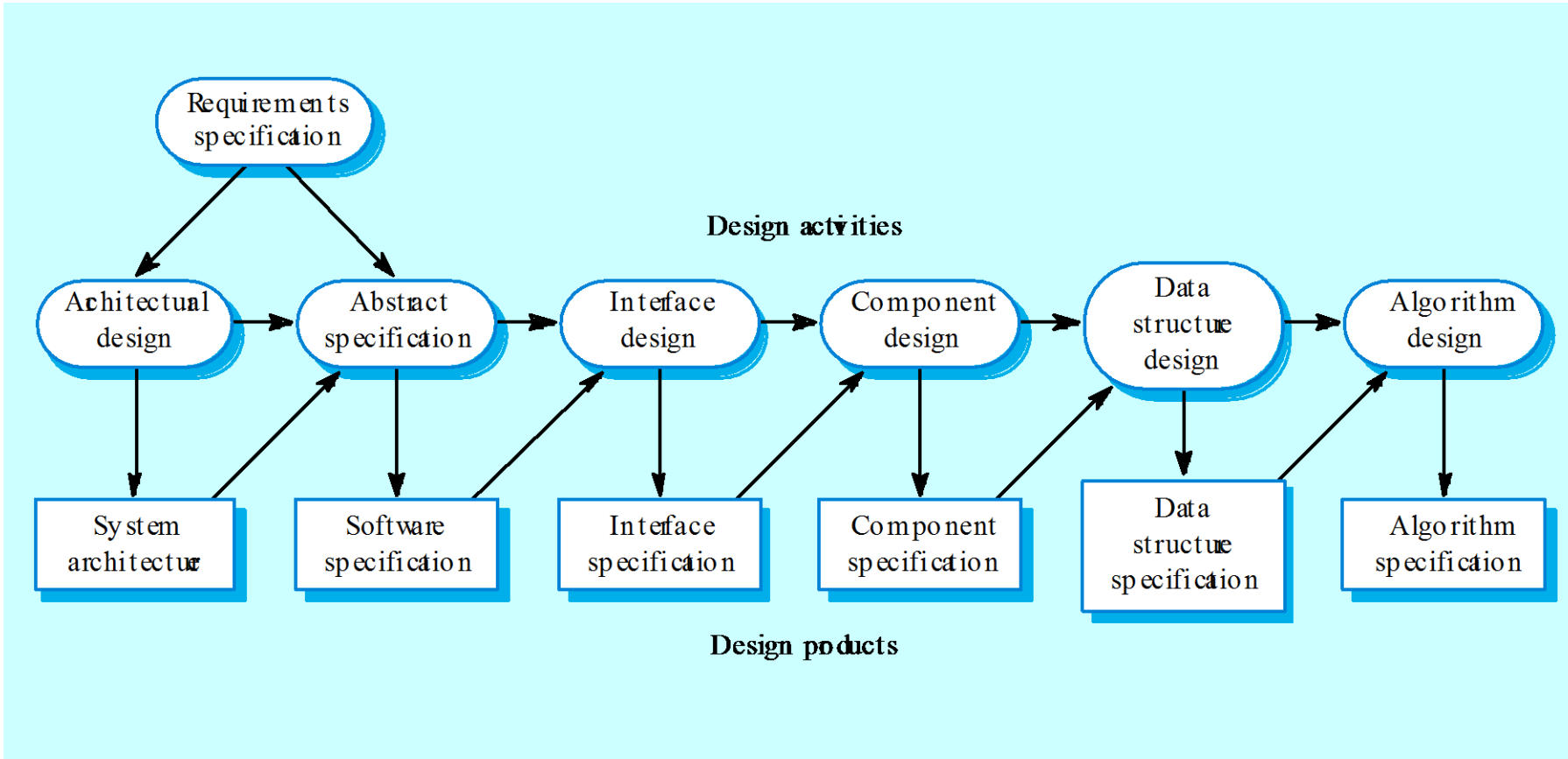
Software design and implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

Design process activities

- Architectural design
- Abstract specification
- Interface design
- Component design
- Data structure design
- Algorithm design

The software design process



Structured methods

- Systematic approaches to developing a software design.
- The design is usually documented as a set of graphical models.
- Possible models
  - Object model;
  - Sequence model;
  - State transition model;
  - Structural model;
  - Data-flow model.

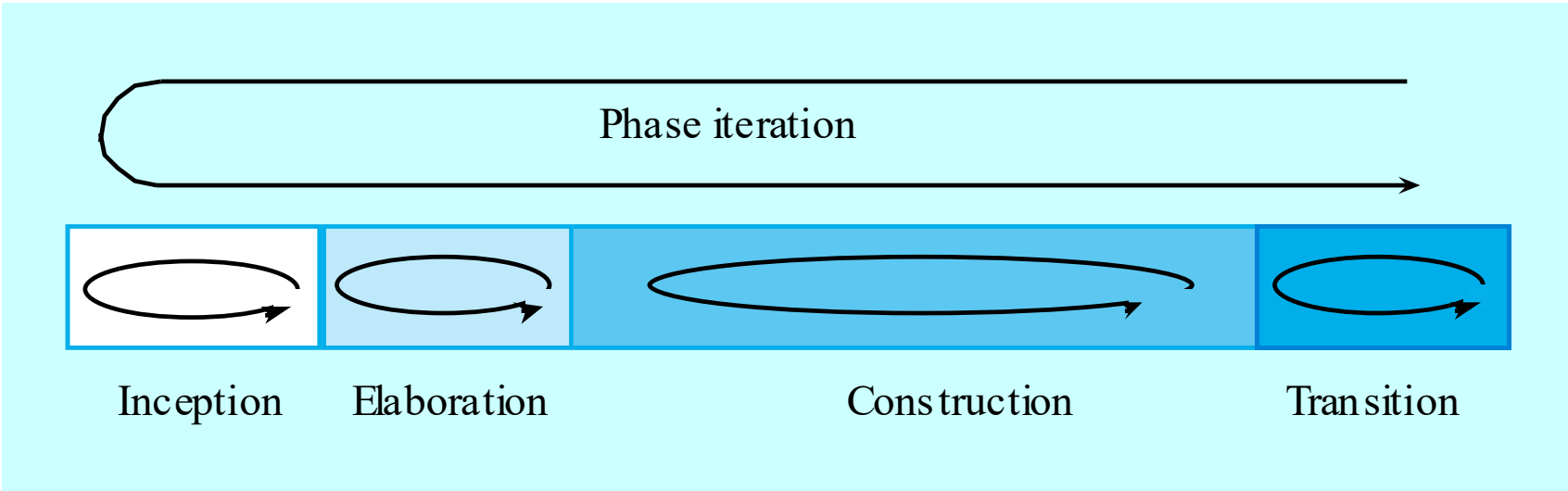
Software validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.

The Rational Unified Process

- A modern process model derived from the work on the UML and associated process.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practive perspective that suggests good practice.

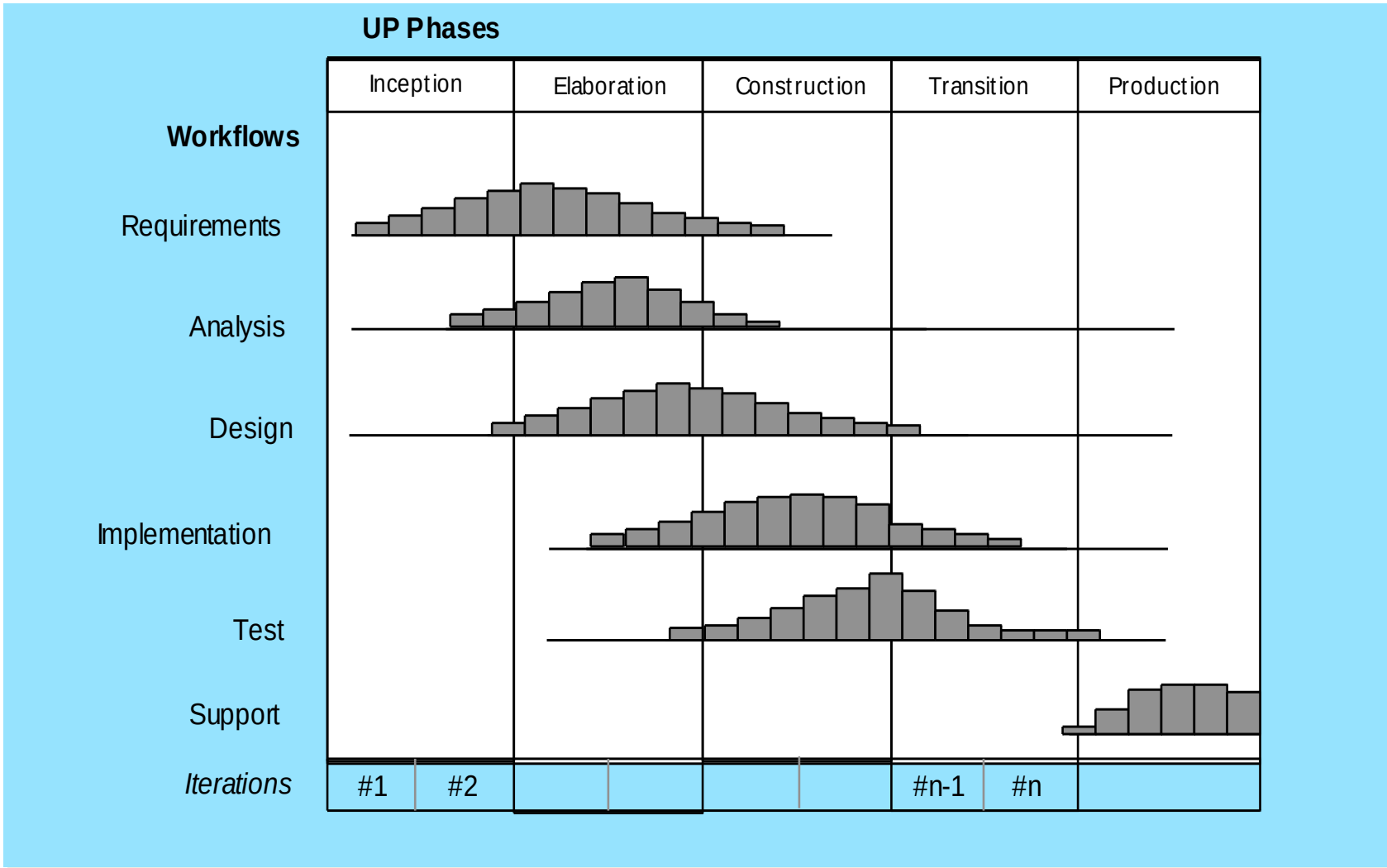
RUP phase model



RUP phases

- Inception
  - Establish the business case for the system.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture.
- Construction
  - System design, programming and testing.
- Transition
  - Deploy the system in its operating environment.

RUP Phases


UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n

RUP Work Products

Inception phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment.
- Project plan,
  - phases and iterations.
- Business model,
  - if necessary.
- One or more prototypes

Elaboration phase

- Use-case model
- Supplementary requirements
  - including non-functional
- Analysis model
- Software architecture
  - Description.
- Executable architectural
  - prototype.
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

Construction phase

- Design model
- Software components
- Integrated software
  - increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current
    - increment

Transition phase

- Delivered software increment
- Beta test reports
- General user feedback

RUP good practice

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

Static workflows

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 29).
Project management	This supporting workflow manages the system development (see Chapter 5).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Agile Model

- **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.

Agile Methodology

- The Agile model helps teams identify and address small issues on projects before they evolve into more significant problems, and it engages business stakeholders to give feedback throughout the development process.

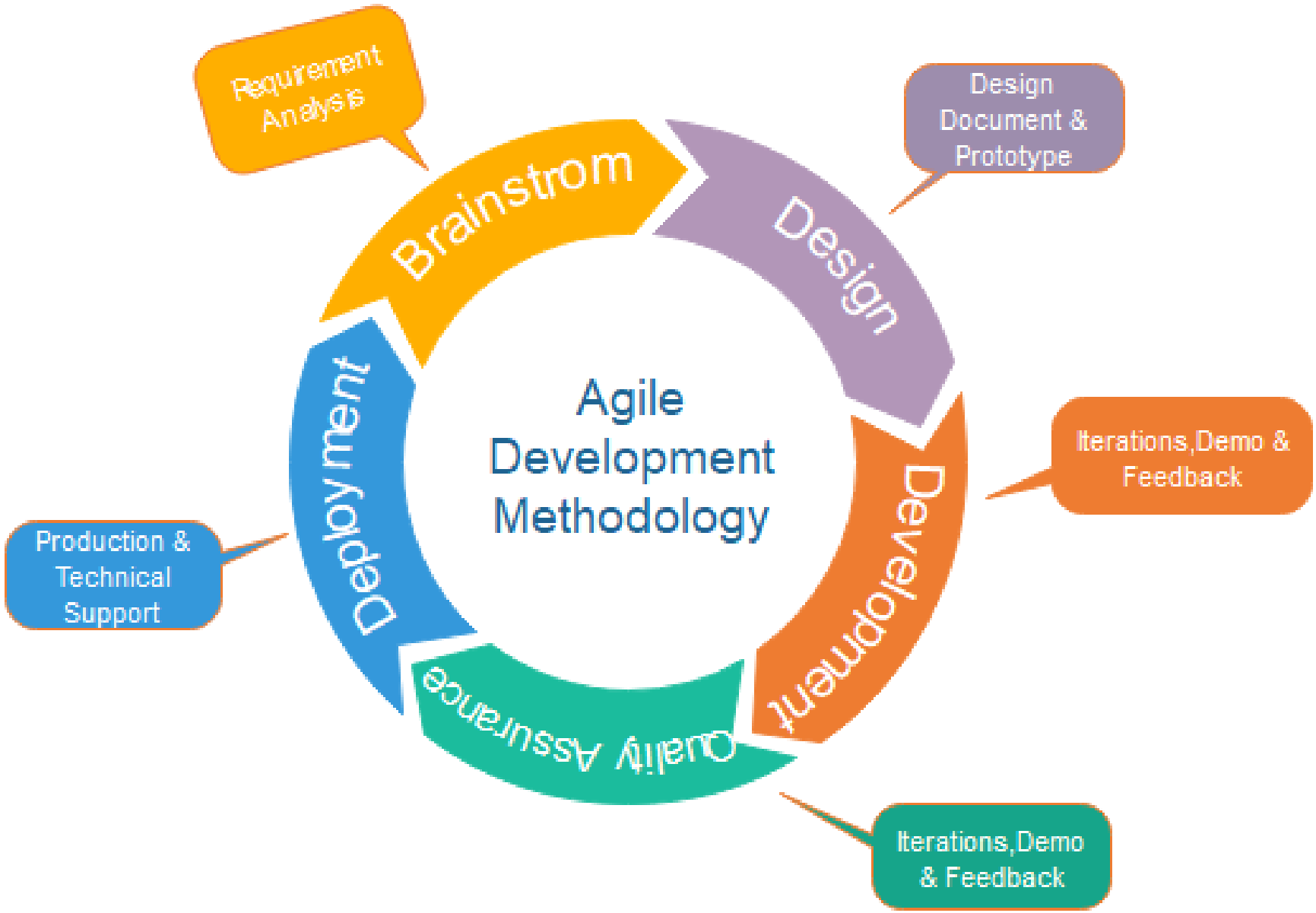


Fig. Agile Model

Phases of Agile Model

- Requirements gathering
- Design the requirements
- Construction/ iteration
- Testing/ Quality assurance
- Deployment
- Feedback

Agile Model

- **Iterative:** Break the project into small cycles and deliver working software early and often.
- **Collaborative:** Work closely with users and stakeholders to understand their needs and get feedback.
- **Adaptive:** Be open to changes and adapt the plan accordingly.
- **Continuous Feedback:** Get feedback from users and stakeholders throughout the development process.

Agile Unified Process Life Cycle

- Model
- Implementation
- Test
- Deployment
- Configuration Management
- Project Management
- Environment

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Advantage(Pros) of Agile Method:

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

Agile Testing Models

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- eXtreme Programming(XP)

Lean Model

- The Lean model for software development is inspired by "lean" manufacturing practices and principles. The seven Lean principles (in this order) are: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build in integrity and see the whole.

- The Lean process is about working only on what must be **worked on at the time**, so there's no room for multitasking.
- Project teams are also focused on finding opportunities to cut waste at every turn throughout the SDLC process, from dropping unnecessary meetings to reducing documentation.

DevOps

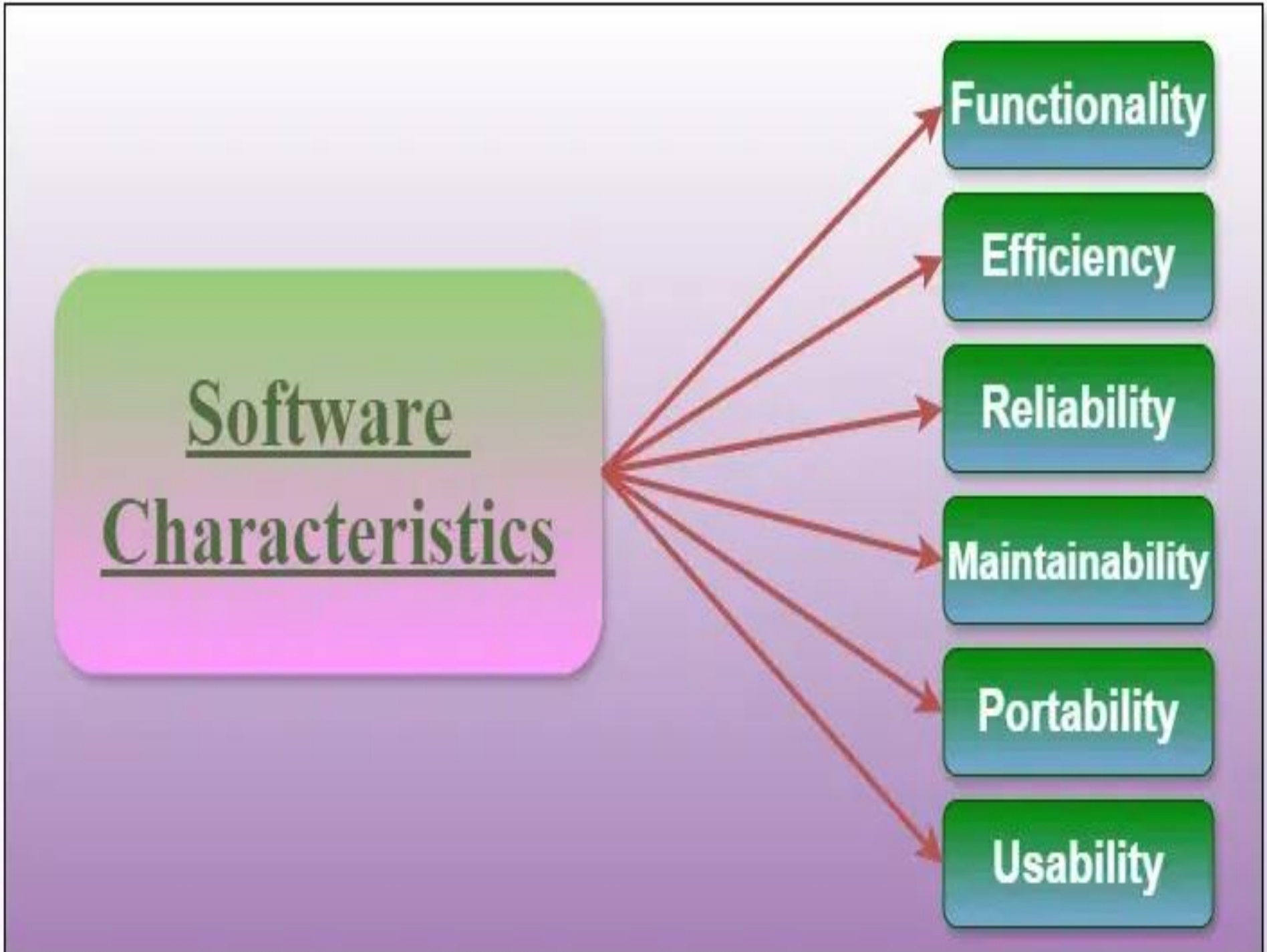
In a DevOps model, Developers and Operations teams work together closely — and sometimes as one team — to accelerate innovation and the deployment of higher-quality and more reliable software products and functionalities.

Updates to products are small but frequent. Discipline, continuous feedback and process improvement, and automation of manual development processes are all hallmarks of the DevOps model.

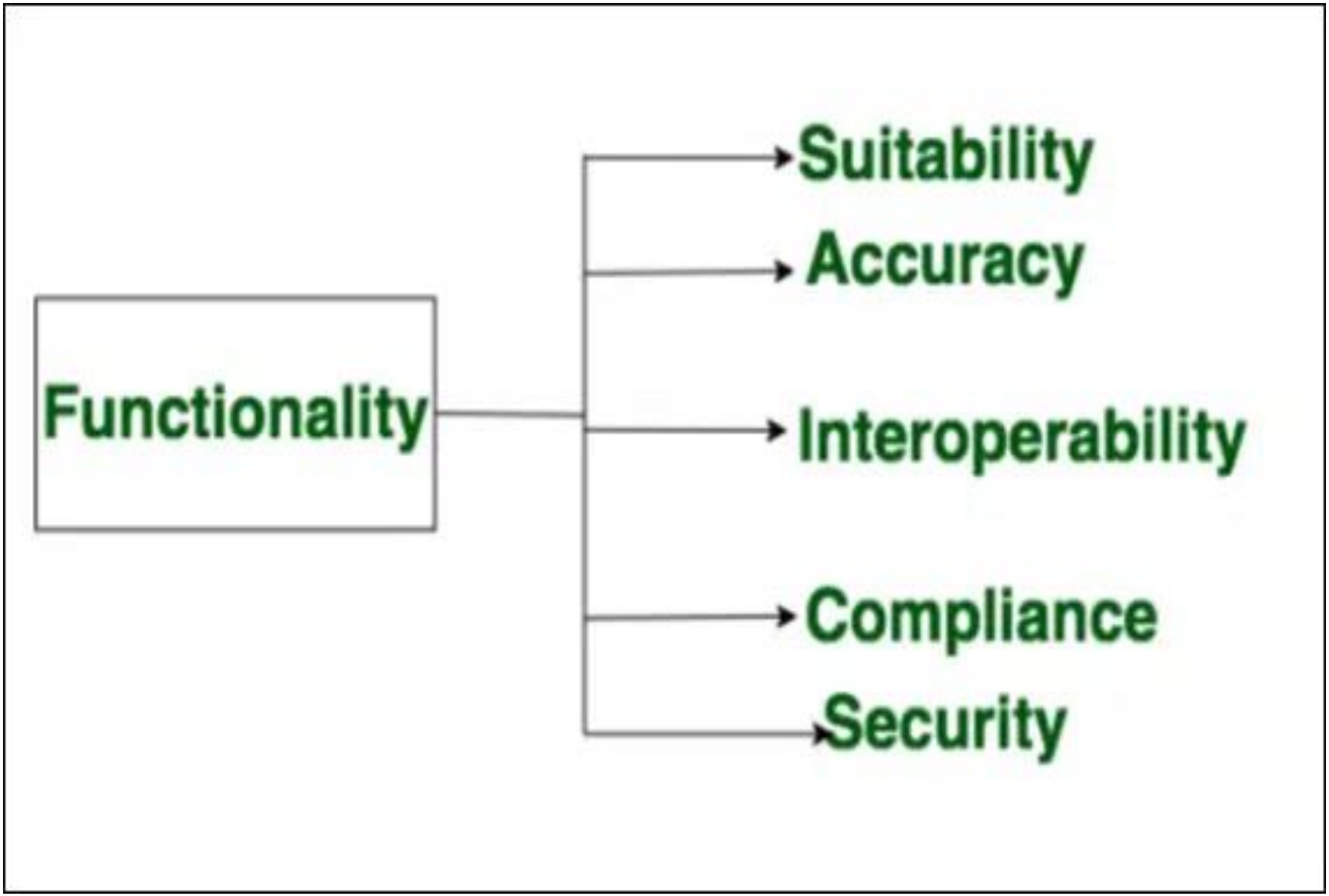
- Amazon Web Services describes DevOps as the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity, evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.
- DevOps is not only an approach to planning and executing work, but also a philosophy that demands a nontraditional mindset in an organization.

Software Quality Standards

- ISO 9126
- Boehm's Software Quality
- Mc Call's Quality Model


Software Characteristics
Functionality
Efficiency
Reliability
Maintainability
Portability
Usability

- Functionality
  - Data storage and retrieval
  - Data processing and manipulation
  - User interface and navigation
  - Communication and networking
  - Security and access control
  - Reporting and visualization
  - Automation and scripting

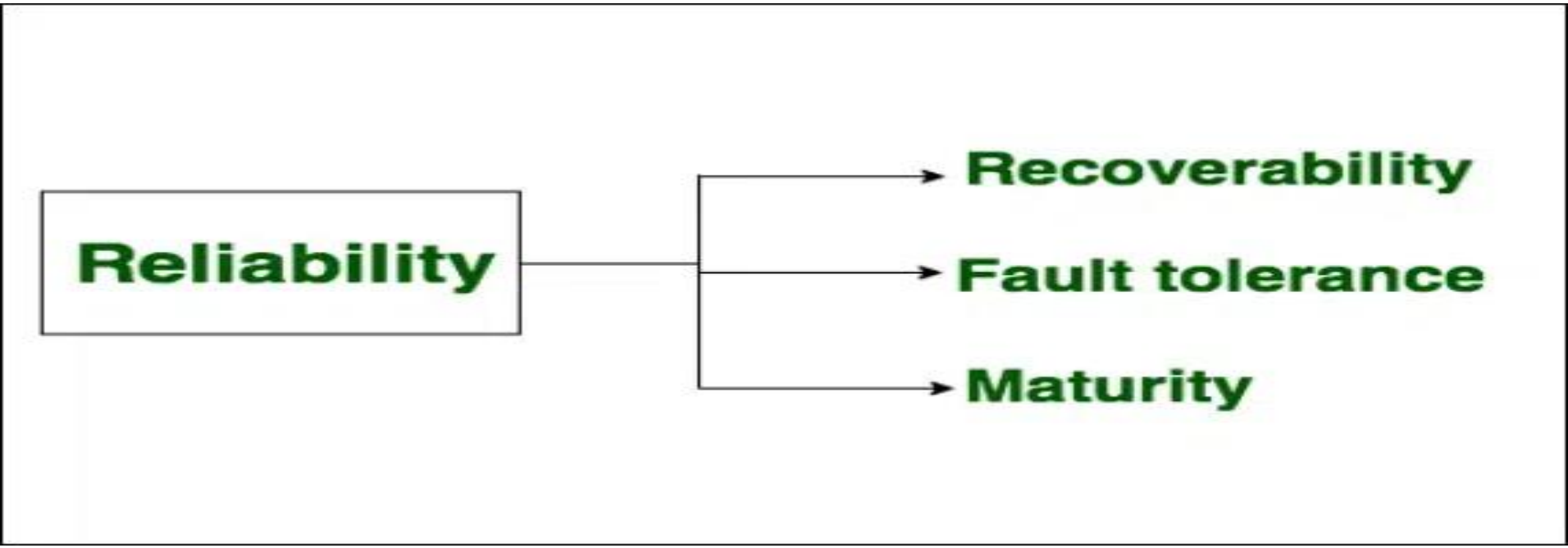

Functionality
Suitability
Accuracy
Interoperability
Compliance
Security

Reliability

Reliability is a characteristic of software that refers to its ability to perform its intended functions correctly and consistently over time.

Factors that can affect the reliability of software include:

- Bugs and errors in the code
- Lack of testing and validation
- Poorly designed algorithms and data structures
- Inadequate error handling and recovery
- Incompatibilities with other software or hardware

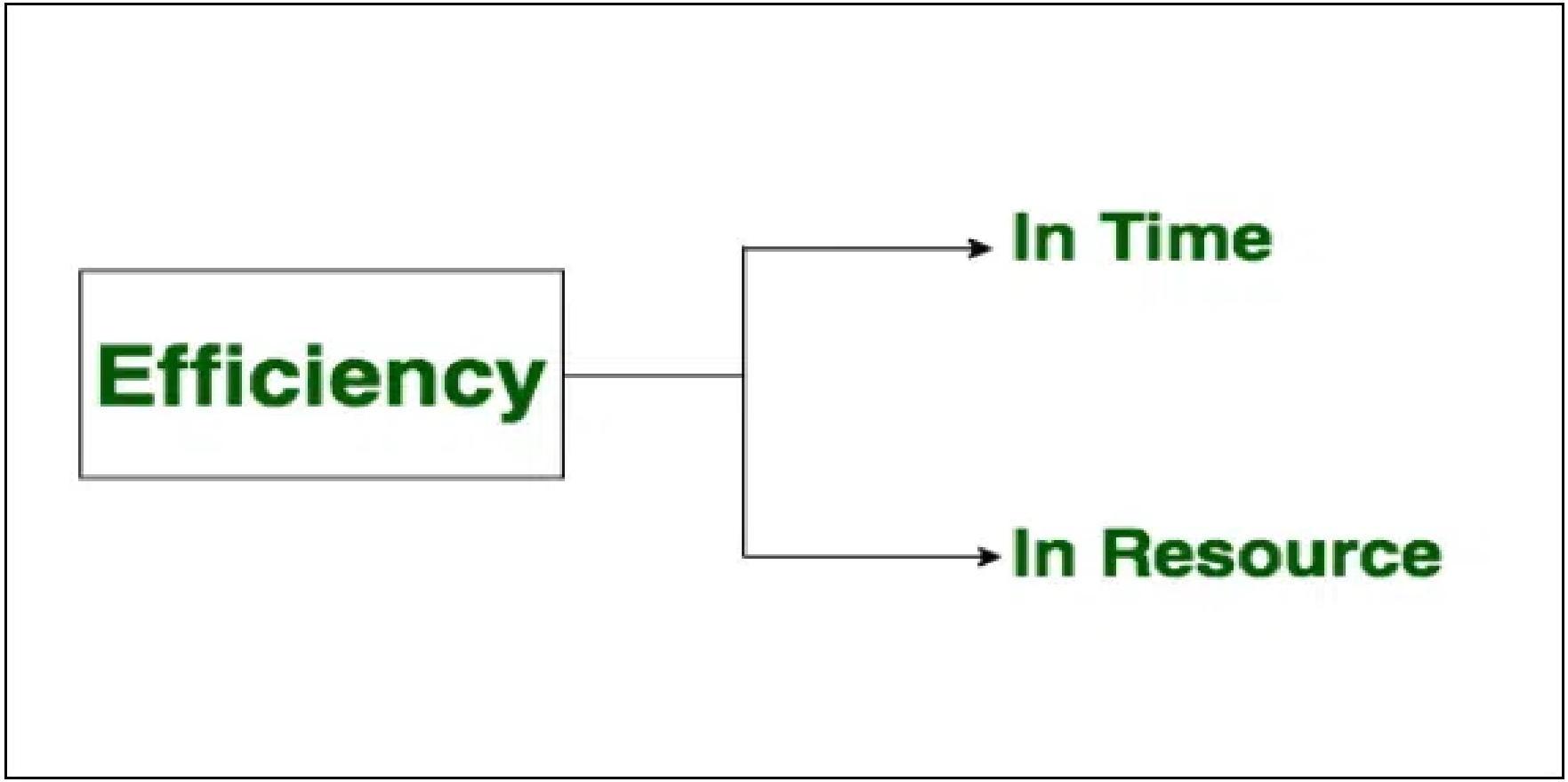

Reliability
Recoverability
Fault tolerance
Maturity

Efficiency

Efficiency is a characteristic of software that refers to its ability to use resources such as memory, processing power, and network bandwidth in an optimal way.

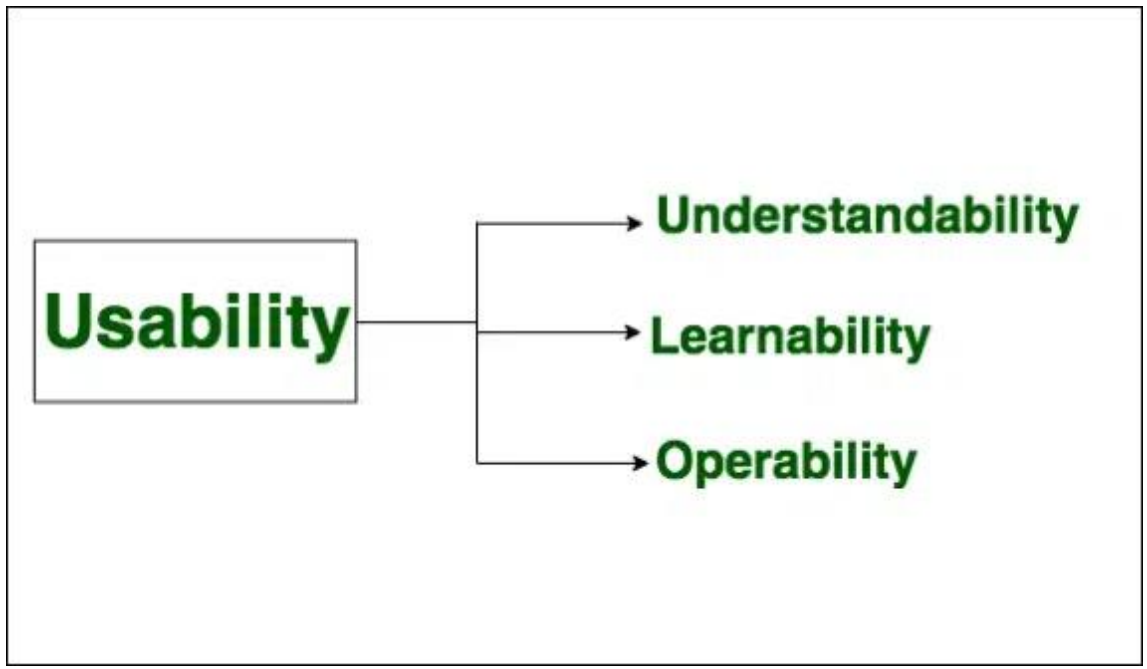
Factors that can affect the efficiency of the software include :

- Poorly designed algorithms and data structures
- Inefficient use of memory and processing power
- High network latency or bandwidth usage
- Unnecessary processing or computation
- Unoptimized code


Efficiency
In Time
In Resource

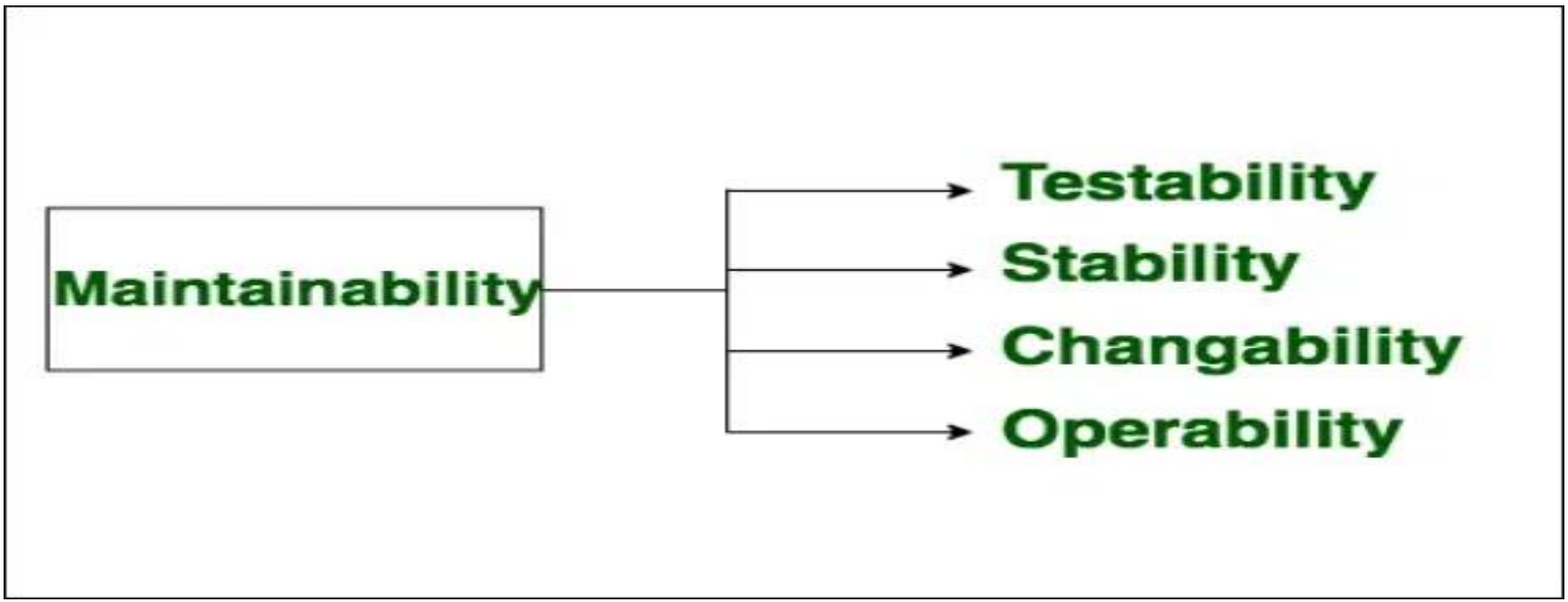
Usability

Software can be used with ease. the amount of effort or time required to learn how to use the software

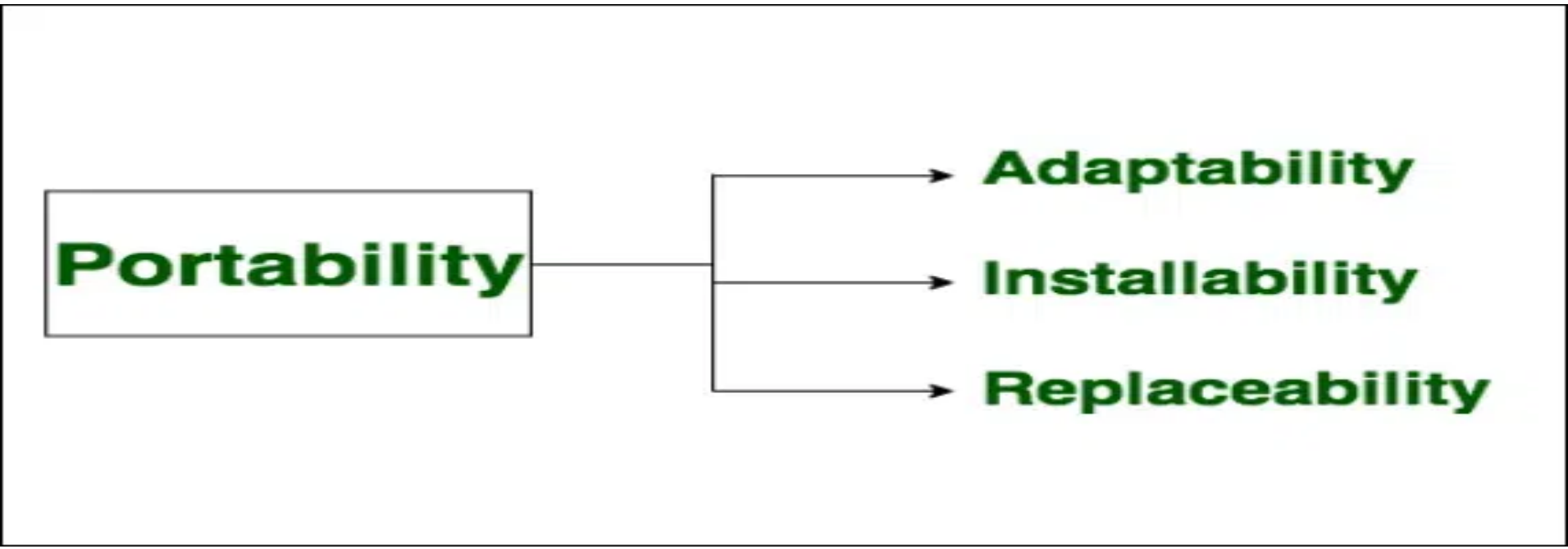


Maintenance

Ease of modifications can be made in a software system to extend its functionality, improve its performance, or correct errors



Portability



Key points

- Software processes are the activities involved in producing and evolving a software system.
- Software process models are abstract representations of these processes.
- General activities are specification, design and implementation, validation and evolution.
- Iterative process models describe the software process as a cycle of activities.
- RUP
- Agile Methodology and AUP
- Software Characteristics and Software Quality Metrics