SOFTWARE ENGINEERING
CSEN1131

MODULE-1 SYLLABUS

UNIT 1 Introduction to Software Engineering

The story of Software development and issues faced, Need for Systematic process for addressing issues, Products, custom solutions, services, domains, Technologies, Software life cycle, software development lifecycle, software release process, source control, versioning, maintenance of software. DevOps.

Software Development Processes: Waterfall, Iterative, Spiral.

WHAT IS SOFTWARE ENGINEERING?

SOFTWARE DEFINITION

Software can defined as:

- Collection of programs & related documents that provide desire features, function & performance.
- Software products developed for a particular customer can be either Generic/ customised
- **Generic -** developed to be sold to a range of different customers/group

  e.g. PC software such as Excel or Word.
- (custom) - developed for a single customer according to their specification

SOFTWARE ENGINEERING DEFINITION

- **It is Defined as a** Discipline in which theories, method & tools are applied to develop quality software Based on problem constraint tools & technologies are applied to develop quality software
- Software engineering is the most important technology with rapid development
  - National economy is dependent
  - More systems are controlled by software

LEGACY SOFTWARE

- They are old and larger computer based systems which may use "obsolete technology"
- These systems may include older hardware, older software old process, old procedures
- Legacy systems are used for long period because it is too risky to replace them
- Air traffic control system
- **Legacy software has important characteristics**
- Poor quality
- Poor documentation
- Poor managed history
- Poor code

Examples of Software Development

- Web Development
- Mobile Development
- Software Tool Development
- Application Development
- Data Management Software Development
- Security Software Development
- Embedded Systems Development
- Cloud Computing Software Development

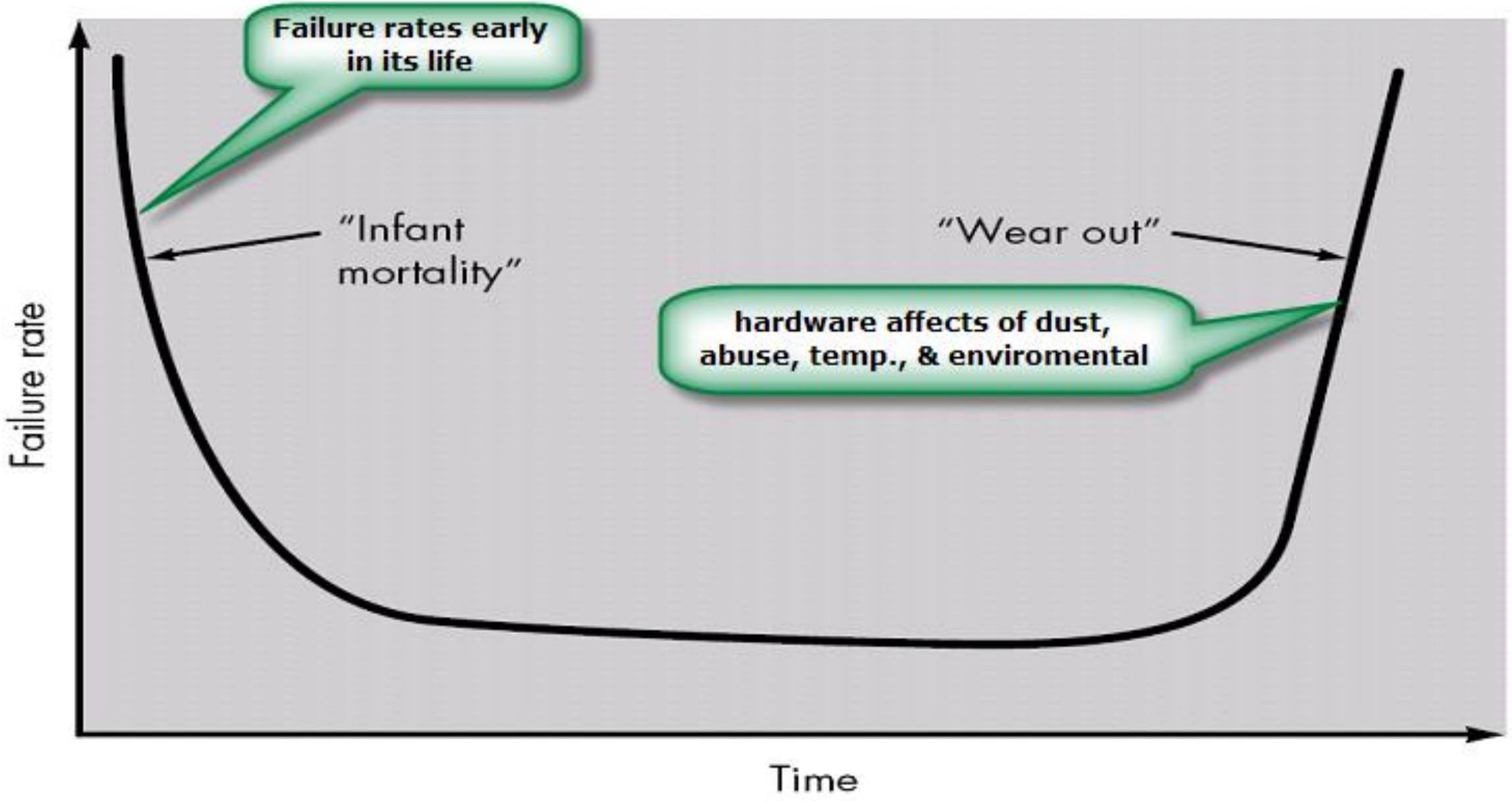
WORK PRODUCT OF SDLC

Requirement gathering	SRS
Design	ER,DFD,Design document
Coding	Source code
Testing	Test plan
Maintanence	User manual

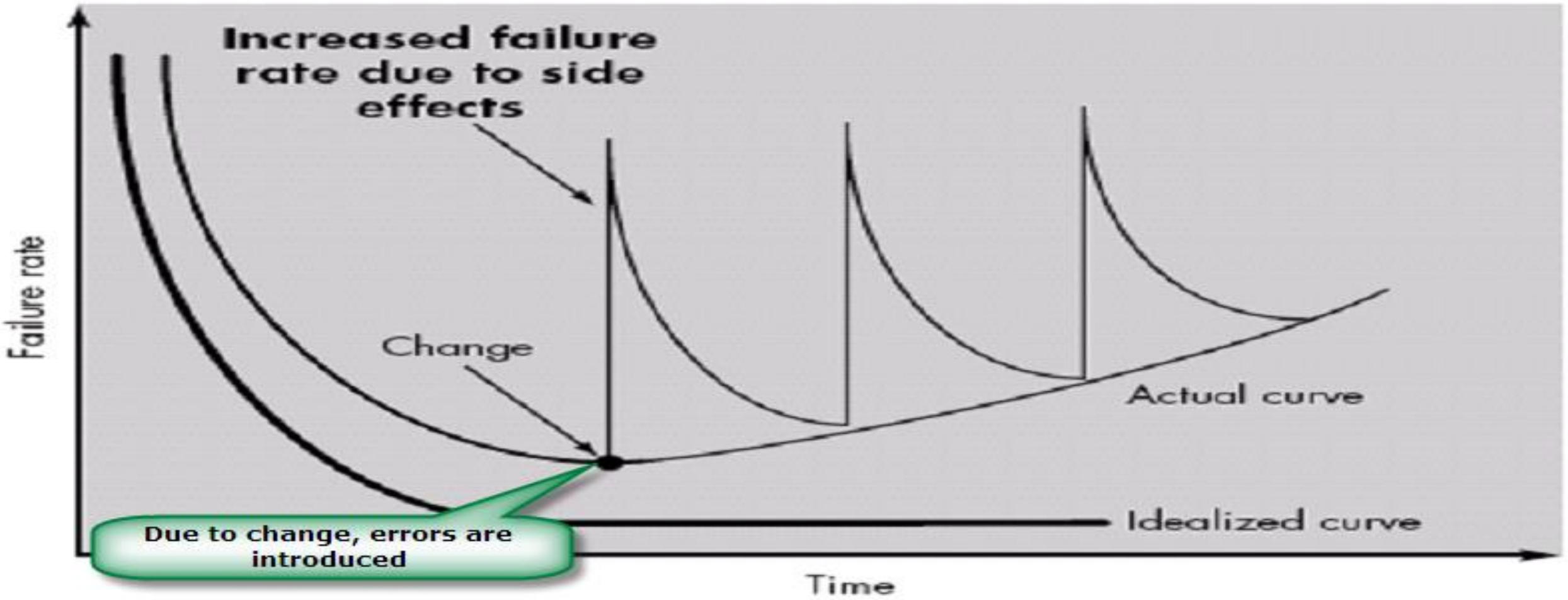
SOFTWARE CHARACTERISTICS

Hardware curve	Software curve
In early stages of hardware development process failure rate is high because of manufacturing defects. After correcting failure rate decreases Failure rate remains constant for period of time Again increases because of environment. Hardware components wears out can be replaced	It doesn't effected with environment. Hence it is a idealized curve. Due to undiscovered error rate is high. During life of software as defects get introduced failure rate is high Curve looks like spike Frequent changes in software causes software to detoriate.

FAILURE CURVE FOR HARDWARE



FAILURE CURVE FOR SOFTWARE



When a hardware component wears out, it is replaced by a spare part.

THE STORY OF SOFTWARE DEVELOPMENT: EVOLUTION OF SOFTWARE DEVELOPMENT

"In today's digital age, software development is at the heart of technological advancements, influencing how we live, work, and communicate.“

Evolution of Software Development

- 1950s - Assembly Language
- 1970s - High-level Programming languages like C and Pascal
- 1990s - Object-oriented Programming (OOP) brought modularity and code reusability.
- 2000s - Agile and Iterative Development emphasizing collaboration, flexibility, and iterative development.

CHANGES IN SOFTWARE OVER A SPAN OF 50 YEARS

Era of computing	Software
Early years	Batch processing, customized software
Second era	Multi users, real time systems
Third era	Distributed systems
Fourth era	Object oriented systems, expert systems,parallel computing
Fifth era	Mobile computing

The Story Of Software Development: Software Lifecycle

- It is a broader concept
- It include its development, usage and retirement.

Phases:

- **Conception:** Identifying the need for software and defining its initial purpose.
- **Development:** The phase covered by SDLC, including design, coding, testing, and deployment.

The Story Of Software Development: Software Lifecycle

Utilization: The period during which the software is actively used by end-users.

Maintenance: Ongoing support, updates, and modifications during the software's active use.

Retirement/Disposal: The phase when the software is phased out, replaced, or discontinued.

The Story Of Software Development: Software Development Lifecycle

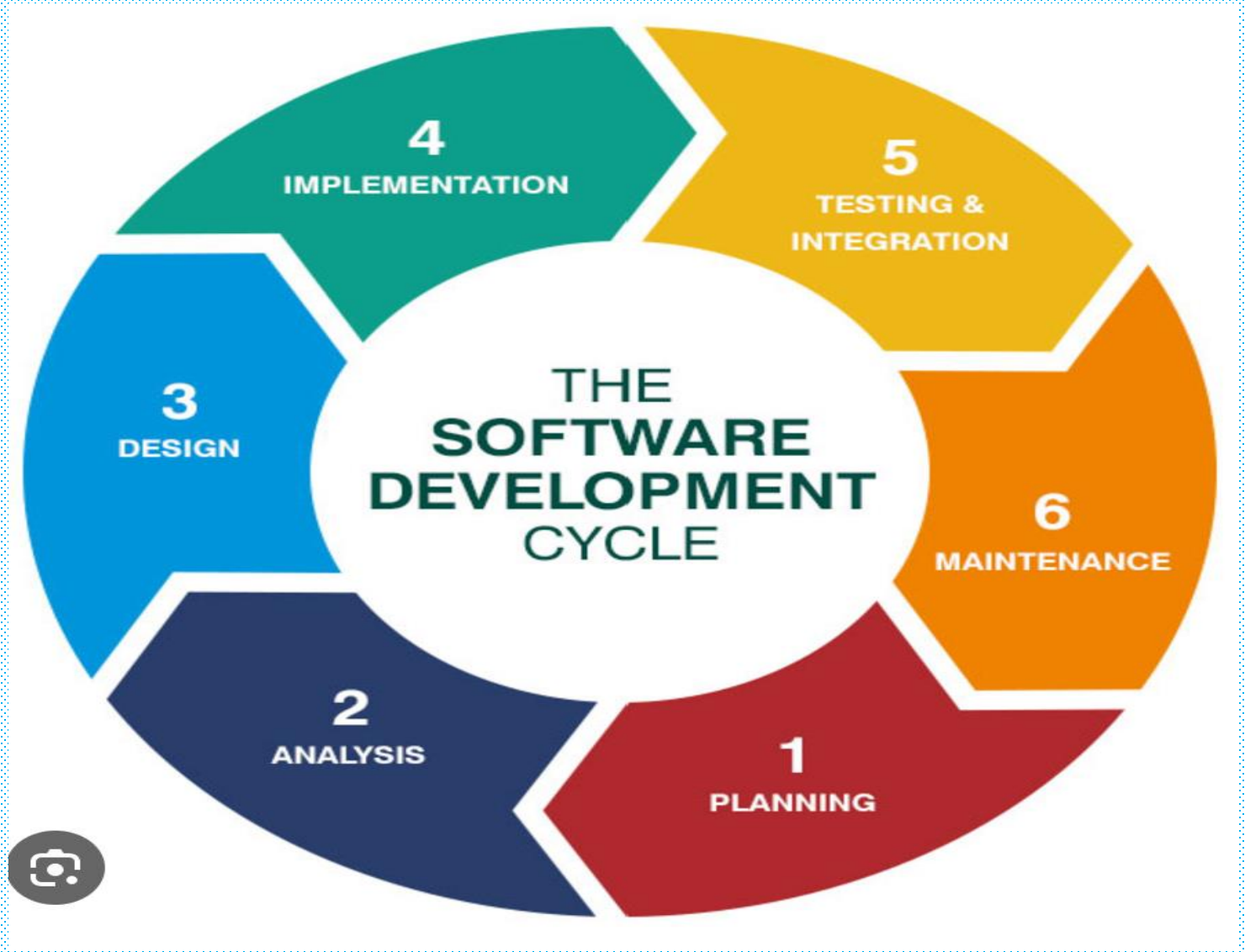
- SDLC is a systematic process used by software developers to design, develop, test, and deploy high-quality software.\

Key Stages:

1. **Requirements:** Gathering and defining what the software is expected to do.
2. **Planning** how the software will meet the requirements.
3. **Design:** Structuring the system architecture.
4. **Implementation:** Writing code that fulfills the design.
5. **Testing:** Verifying that the software works as intended. Ensuring the software meets quality standards
6. **Deployment:** Releasing the software for use.
7. **Maintenance:** Ongoing support, bug fixes, and updates. Updating and refining software post-deployment.

The Story Of Software Development:

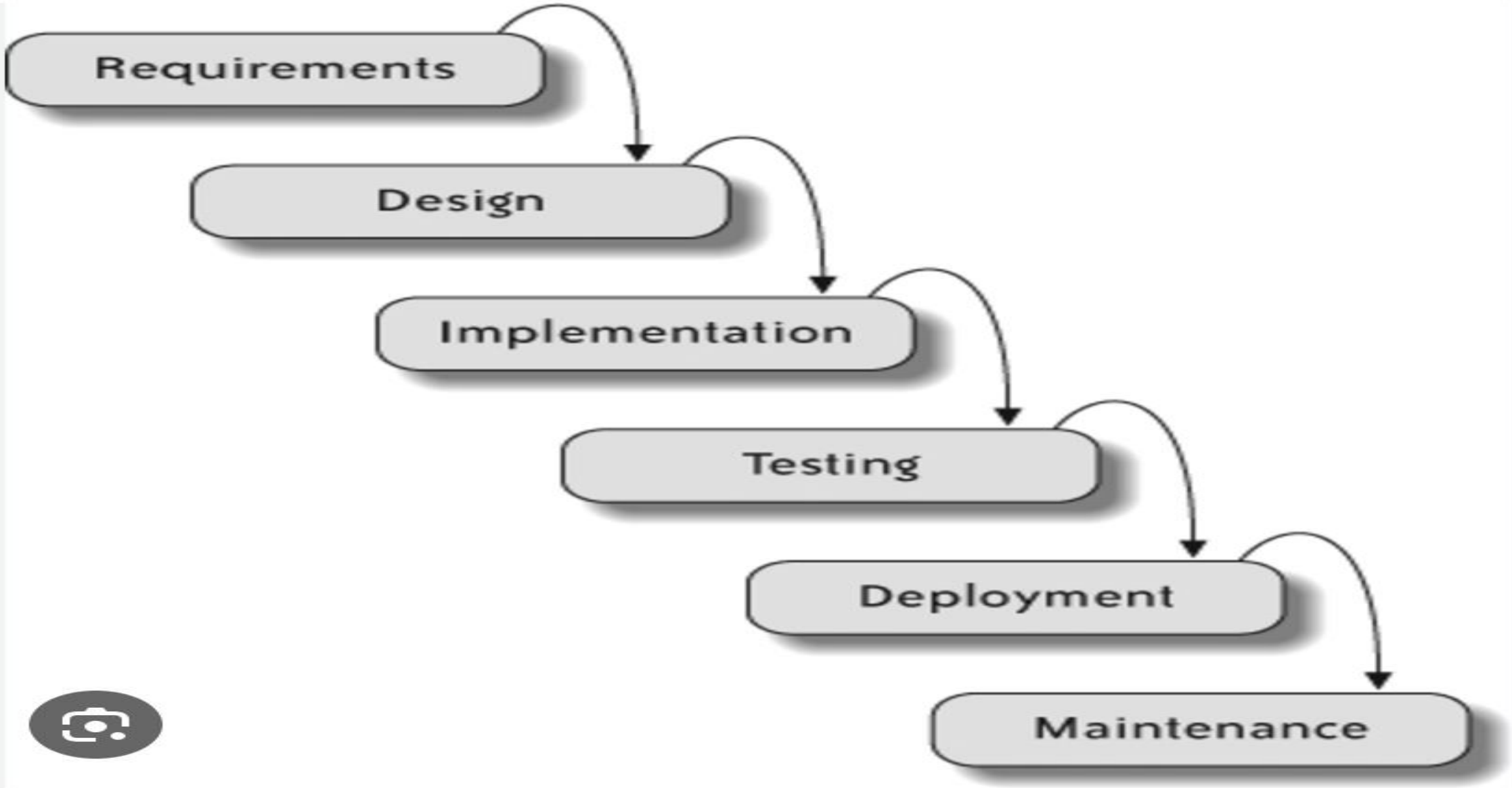
Software Development Lifecycle

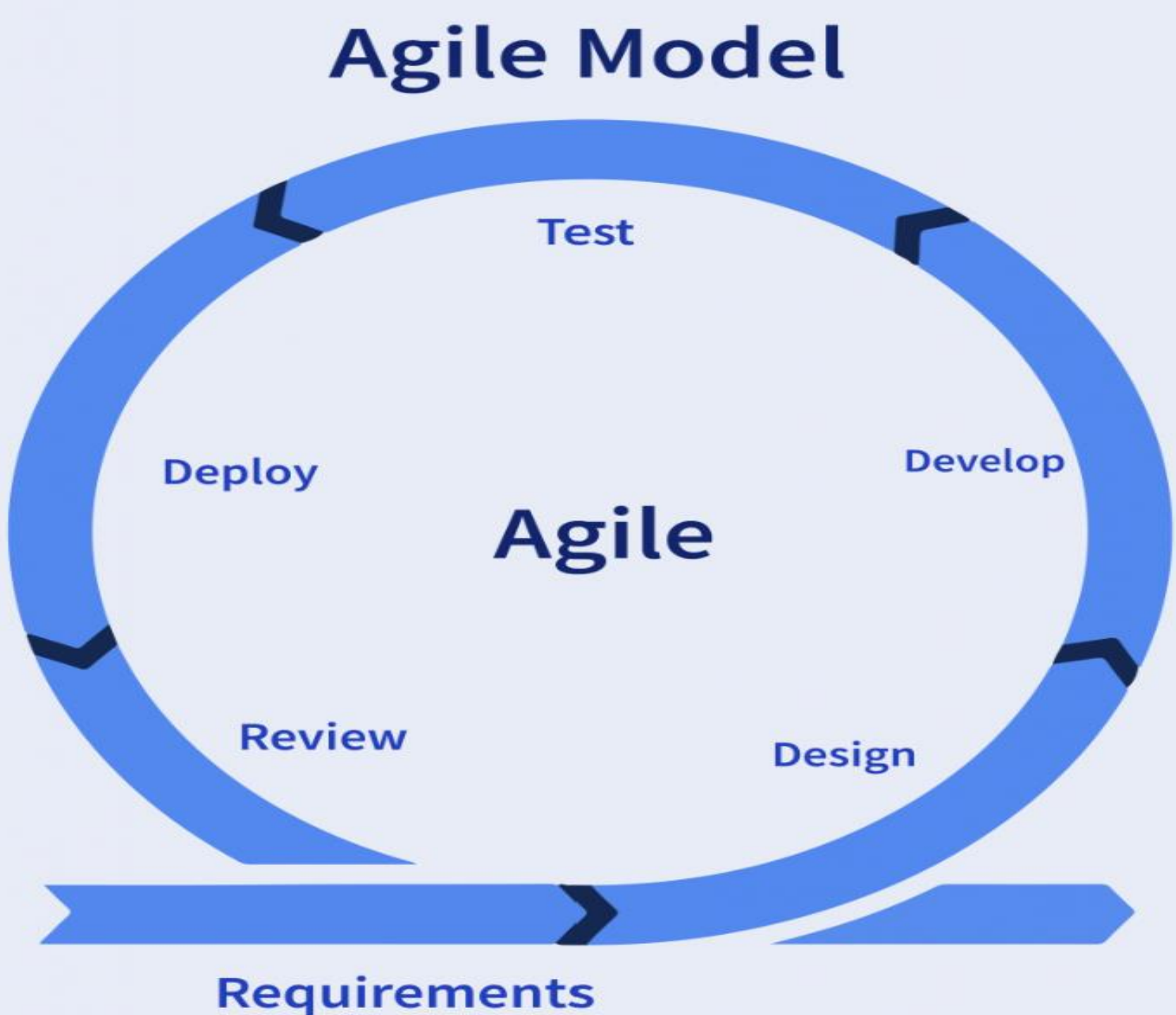


The Story Of Software Development: Approaches to Development

- **Waterfall Model**: Sequential development with well-defined stages.
- **Agile Methods**: Iterative and incremental development with a focus on adaptability.
- **DevOps**: Integration of development and operations for continuous delivery

WATERFALL MODEL




Agile Model
Test
Develop
Design
Requirements
Review
Deploy
Agile

DEVOPS FLOWCHART

CHALLENGES IN SOFTWARE ENGINEERING

- **Collaborating** with other team members, such as project managers, designers, and other developers
- **"Complexity of Projects - Managing intricate software projects with numerous components and dependencies."**
- **"Changing Requirements - Adapting to evolving client needs throughout the development process.“**
- **Meeting Deadlines** to ensure the quality of the final product.
- **Debugging** software can be time-consuming and complex, particularly for large and complex systems.

COMMON CHALLENGES FACED IN SOFTWARE DEVELOPMENT

- **Security concerns**
- **Quality assurance**
- **Keeping up with new technologies**
- **Managing changing requirements**
- **Collaboration issues**
- **Time Delays**
- **Dealing with legacy code:** Developers often have to work with legacy code.
- **Balancing short-term and long-term goals:** we need to ensure that the software is maintainable and scalable in the long-term.

Common issues in software development

- "Budget Overruns - Overspending due to unforeseen challenges or scope changes."
- "Code Complexity - Difficulty in managing codebases, leading to potential errors.“

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: MANAGING COMPLEXITY

Problem: Software systems are inherently complex due to their size, functionality, and integration with other systems.

Systematic Process:

- Divide the system into smaller, manageable components (modular design).
- Use structured methodologies like the Waterfall or Spiral model to approach development in stages.

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: HANDLING CHANGING REQUIREMENTS

Problem: Requirements often evolve, leading to scope and impacting timelines and budgets.

Systematic Process:

- Implement Requirements Engineering to gather, analyze, and document needs systematically.
- Use iterative models (e.g., Agile) that accommodate changes through incremental updates.

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: ENSURING QUALITY ASSURANCE

Problem: Without a structured testing process, defects can go unnoticed until deployment.

Systematic Process:

- Incorporate Verification and Validation (V&V) techniques to catch defects early.
- Use systematic testing strategies, including:

unit testing

integration testing

user acceptance testing.

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: RISK MANAGEMENT

Problem: Unanticipated risks like technology failures or resource unavailability can derail projects.

Systematic Process:

- Identify and assess risks during project planning.
- Use risk mitigation frameworks to develop contingency plans.

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: BETTER RESOURCE MANAGEMENT

Problem: Projects often face resource constraints, including time, budget, and skilled personnel.

Systematic Process:

- Use tools like Gantt charts and resource allocation matrices for effective planning.
- Employ methodologies like Critical Path Analysis (CPA) to identify and focus on key activities.

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: COLLABORATION AND COMMUNICATION

Problem: Miscommunication between stakeholders and developers leads to mismatched expectations.

Systematic Process:

Use clear documentation standards for requirement specifications, design models, and test cases.

Conduct regular stakeholder meetings and progress reviews.

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES:

LONG-TERM MAINTENANCE AND SUSTAINABILITY

Problem: Software often needs updates, bug fixes, and integration with newer systems post-deployment.

Systematic Process:

Follow coding and design standards that prioritize maintainability.

Use Configuration Management tools to track changes and version history

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: SECURITY AND RELIABILITY

Problem: Increasing cybersecurity threats demand robust and secure software systems.

Systematic Process:

- Embed security requirements during the design phase.
- Conduct systematic penetration testing and threat modeling to identify vulnerabilities.

NEED FOR SYSTEMATIC PROCESS FOR ADDRESSING ISSUES: MEETING DEADLINES AND BUDGETS

Problem: Without structured planning, projects often exceed time and cost estimates.

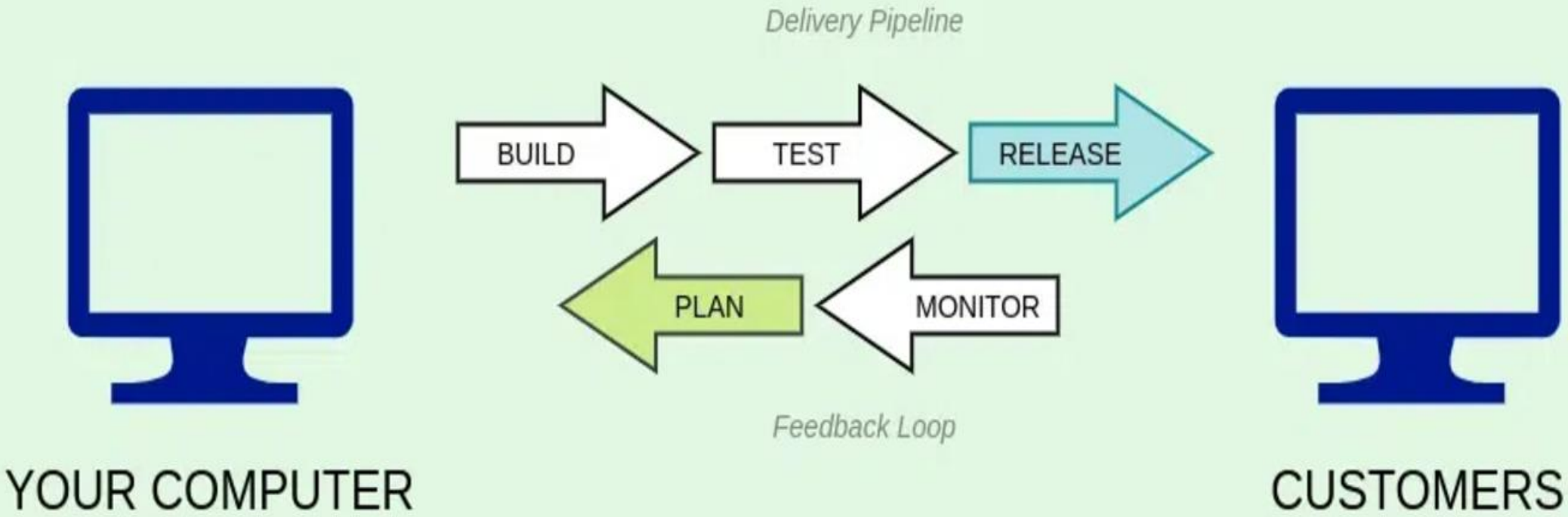
Systematic Process:

- Use project management tools and frameworks (e.g., SCRUM in Agile).
- Set realistic milestones and monitor progress regularly.

DEVOPS ("DEVELOPMENT" AND "OPERATIONS")

- It is a software development approach
- DevOps Engineer combines software development and IT operations to improve how software is built and deployed
- It is designed to increase an organization's ability to deliver applications and services faster than traditional software development processes.
- It aims to shorten the software development lifecycle (SDLC)
- It emphasizes collaboration and communication between development (Dev) and operations (Ops) teams.

DevOps Model



DEVOPS: DELIVERY PIPELINE

- The pipeline represents the different stages that software goes through before it is released to production.

These stages might typically include:

- **Build:** software code is compiled and packaged into a deployable unit.
- **Test:** software is rigorously tested to ensure it functions as expected and identifies any bugs.
- **Release:** software is deployed to production for end users.


Continuous Development
Continuous Integration
Continuous Testing
Continuous Deployment
Continuous Monitoring
Continuous Feedback
Continuous Operations
DevOps Lifecycle

C's of DevOps

1. Continuous Development
2. Continuous Integration
3. Continuous Testing
4. Continuous Deployment/Continuous Delivery
5. Continuous Monitoring
6. Continuous Feedback
7. Continuous Operations

BENEFITS OF DEVOPS

1. **Faster Delivery:** DevOps enables organizations to release new products and updates faster and more frequently, which can lead to a competitive advantage.

2. **Improved Collaboration:** DevOps promotes collaboration between development and operations teams, resulting in better communication, increased efficiency, and reduced friction.

3. **Improved Quality:** DevOps emphasizes automated testing and continuous integration, which helps to catch bugs early in the development process and improve the overall quality of software.

4. **Increased Automation:** DevOps enables organizations to automate many manual processes, freeing up time for more strategic work and reducing the risk of human error.

BENEFITS OF DEVOPS

6. **Increased Customer Satisfaction:** DevOps helps organizations to deliver new features and updates more quickly, which can result in increased customer satisfaction and loyalty.

7. **Improved Security:** DevOps promotes security best practices, such as continuous testing and monitoring, which can help to reduce the risk of security breaches and improve the overall security of an organization's systems.

8. **Better Resource Utilization:** DevOps enables organizations to optimize their use of resources, including hardware, software, and personnel, which can result in cost savings and improved efficiency.

SOURCE CONTROL (OR VERSION CONTROL)

- The practice of tracking and managing changes to code.
- Source control management (SCM) systems provide a running history of code development
- It help to resolve conflicts when merging contributions from multiple sources.

VERSION CONTROL

- They are category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.
- They keeps track on changes made on a particular software and take a snapshot of every modification.
- Helps in recovery in case of any disaster or contingent situation,
- Informs us about Who, What, When, Why changes have been made.
- Reduce possibilities of errors and conflicts

TYPES OF VERSION CONTROL SYSTEMS

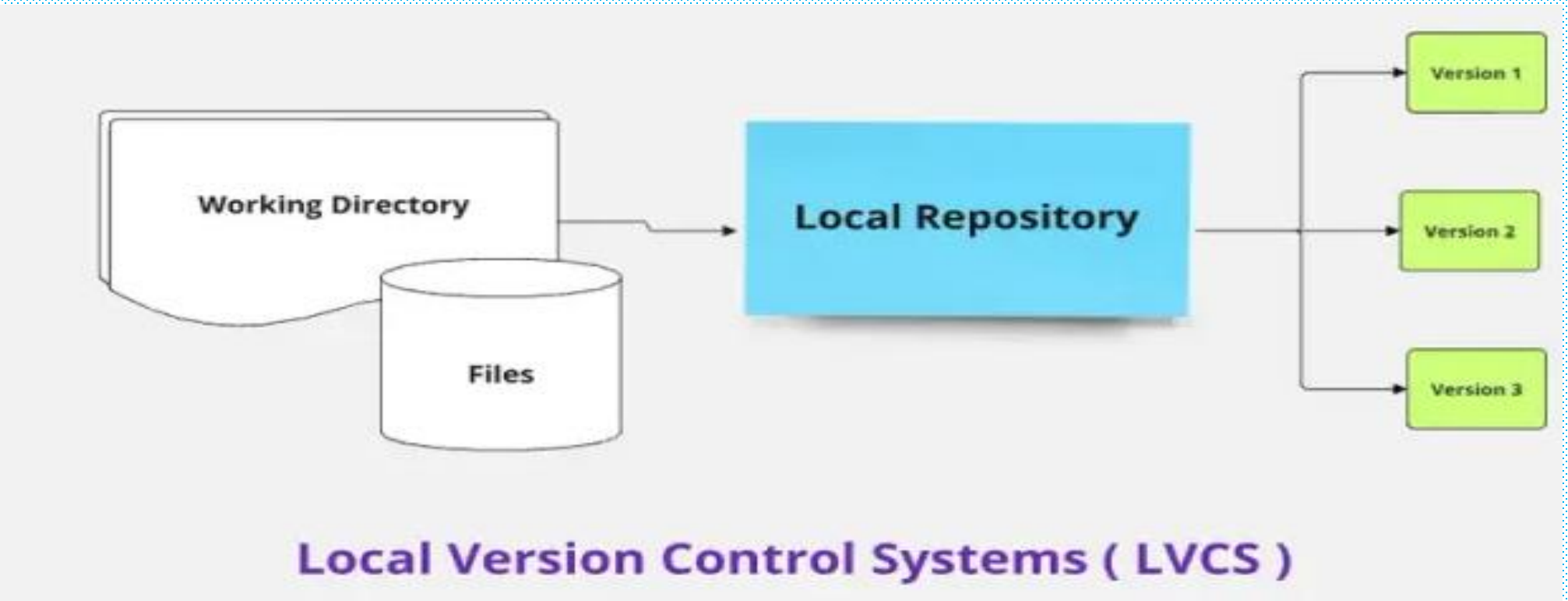
- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems

Two things are required to make your changes visible to others

- You commit
- They update

LOCAL VERSION CONTROL SYSTEMS

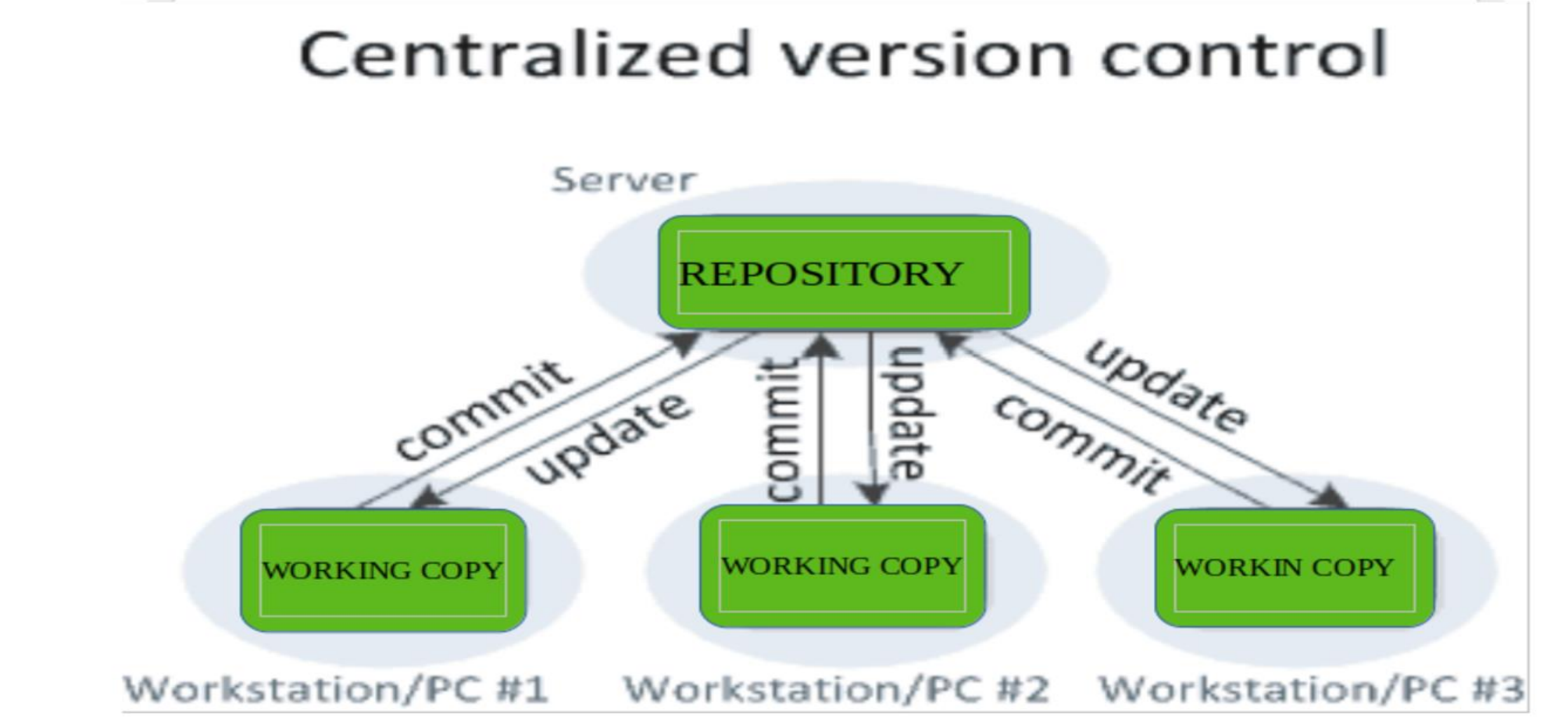
- It is one of the simplest forms
- It has a database
- It keeps all the changes to files under revision control



Centralized Version Control Systems

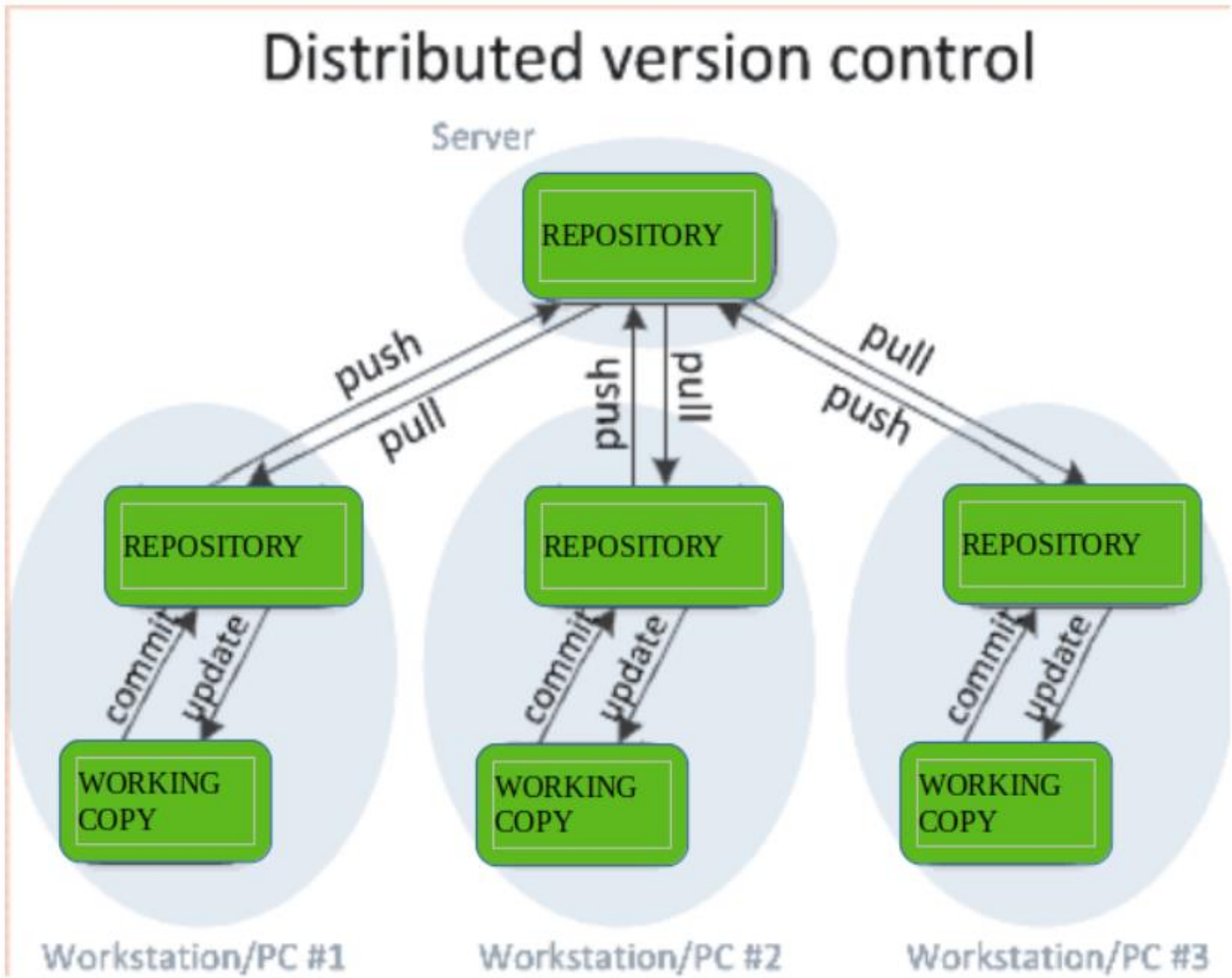
Contain Just One Repository Globally

Every User Need To Commit For Reflecting One's Changes In The Repository



DISTRIBUTED VERSION CONTROL SYSTEMS

- It contain multiple repositories.
- Each user has their own repository and working copy.
- Just committing your changes will not give others access to your changes.
- This is because commit will reflect those changes in your local repository
- you need to push them in order to make them visible on the central repository


Distributed version control
Server
REPOSITORY
push
pull
push
pull
pull
push
REPOSITORY
REPOSITORY
REPOSITORY
commit
update
commit
update
commit
update
WORKING COPY
WORKING COPY
WORKING COPY
Workstation/PC #1
Workstation/PC #2
Workstation/PC #3

WHAT IS THE SOFTWARE RELEASE PROCESS?

- It is also known as Release Management Cycle

It performs a cycle for

Developing

testing

deploying

supporting new versions of software.

PHASES OF A SOFTWARE RELEASE PROCESS

1. Define specific requirements for the release
2. Specify your acceptance criteria
2. Test your software in production
4. Iterate and refine your product
5. Release your product to end-users

How does the software release process typically work?

The software release process is a series of steps that teams follow to get new software ready.

1. **Planning:** First, developers figure out what changes or new features need to be added to the software.

2. **Building:** Next, they write the code to create those features. This is where the actual work of making the software happens.

How does the software release process typically work?

3. **Testing:** Before the software is released, it goes through testing to catch any problems.

4. **Launching:** Once everything is tested and working well, the software is made available to users.

5. **Updating:** After the launch, the team keeps an eye on the software, fixing any issues that come up an making improvements based on user feedback.


Request For Changes
Release Planning & Design
Software build
Review
Test
Deployment
Support
Issue Reporting & Collection
Release Management Process

There are three types of service in the system:

1. **Utility services**
2. **Application services**
3. **Configuration services**

UTILITY SERVICES

- It provide basic application-independent functionality
- It may be used by other services in the system.
- Utility services are usually developed or adapted specifically for this system.

WHAT IS UTILITY SOFTWARE?

- Utility Software is a type of software which is used to analyse and maintain a computer.
- This software is focused on how OS works on that basis it performs tasks to enable the smooth functioning of the computer.

Types of Utility Software

- Antivirus software
- Disk cleaners
- Backup and recovery software
- System optimizers
- Disk defragmenter
- File compression software
- Disk encryption software

Examples of Utility Software

- Antivirus
- File Management
- Disk Management
- Compression Tools

Application services

provide specific applications such as email, conferencing, photo sharing

Access to specific educational content such as scientific

films or historical resources.

They are external services that are either specifically purchased for the system

or are available freely over the Internet.

Application Service Provider

pplication Owners

Application Users

Wallarm

What Are Application Services?

Availability

Mobility

Application services

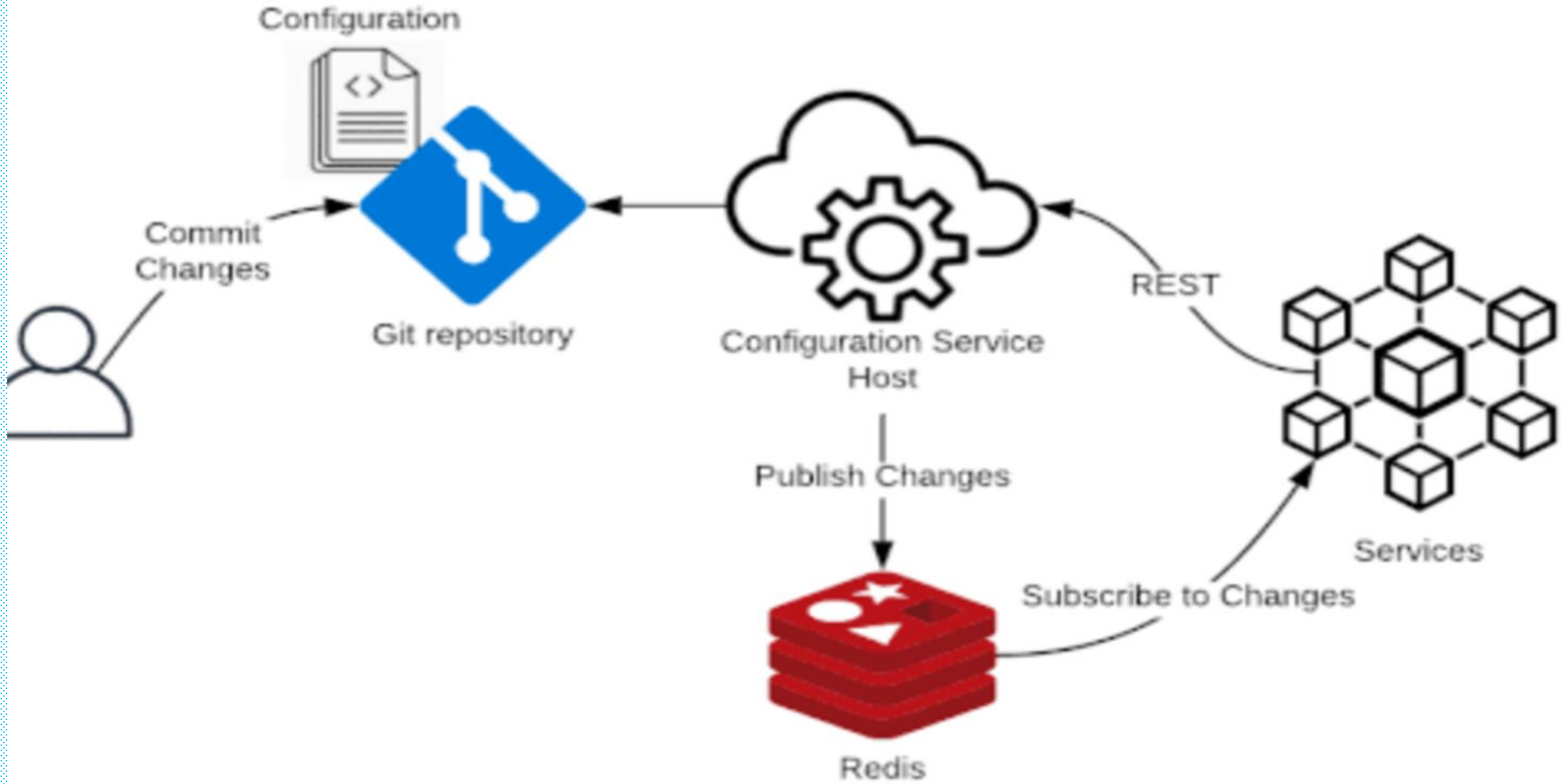
Identity/Access

Security

Performance

CONFIGURATION SERVICES

- used to manage and customize technical systems and products to meet customer preferences:
- It define how services are shared between students, teachers, and their parents.


Configuration
Commit
Changes
Git repository
Configuration Service
Host
REST
Services
Publish Changes
Subscribe to Changes
Redis

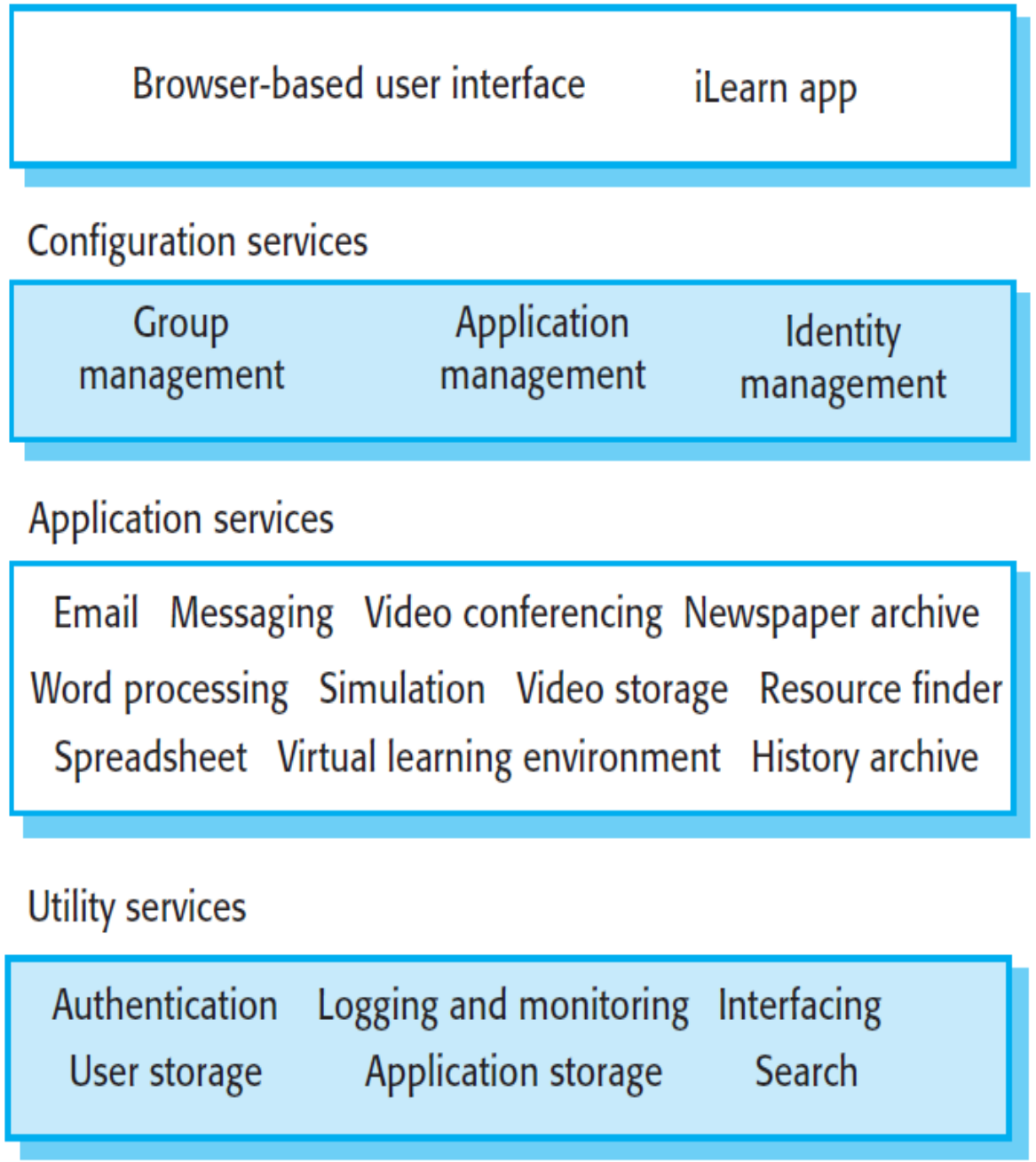


Figure 1.8 The architecture of a digital learning environment (iLearn)

INTEGRATED SERVICES

- They are services that offer an API (application programming interface)
- They can be accessed by other services through that API.
- An authentication service is an example of an integrated service.
- Rather than use their own authentication mechanisms
- Authentication service may be called on by other services to authenticate users.
- If users are already authenticated, then the authentication service may pass authentication information directly to another service.

Integration with Other Google Apps

01 Google Calendar Integration

02 Gmail Integration

03 Google Drive Integration

06 Google Docs Integration

04 Google Keep Integration

05 Google Sheets Integration

INDEPENDENT SERVICES

- ***They*** are services that are simply accessed through a browser
- interface and that operate independently of other services.
- reauthentication may be required for each independent service.

INTRODUCTION TO PROCESS MODEL

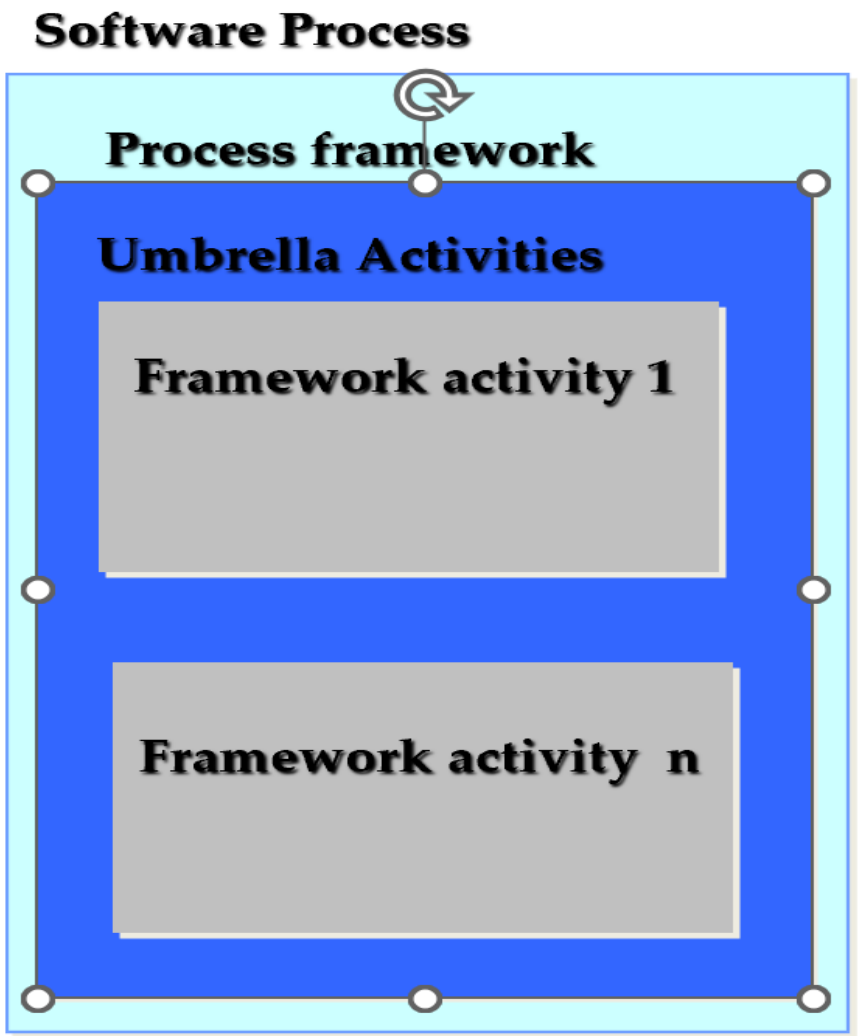
- Process model provides Systematic approach to software development
- Model defines set of { activities, work tasks, work products} to develop quality software
- Process model is an abstract representation of process
- Based on nature of software project process model s chosen

INTRODUCTION TO PROCESS MODEL

Stages in software development process:

- Problem specification
- Requirement gathering
- Analysis
- Design
- Coding
- testing
- maintenance

Process Framework

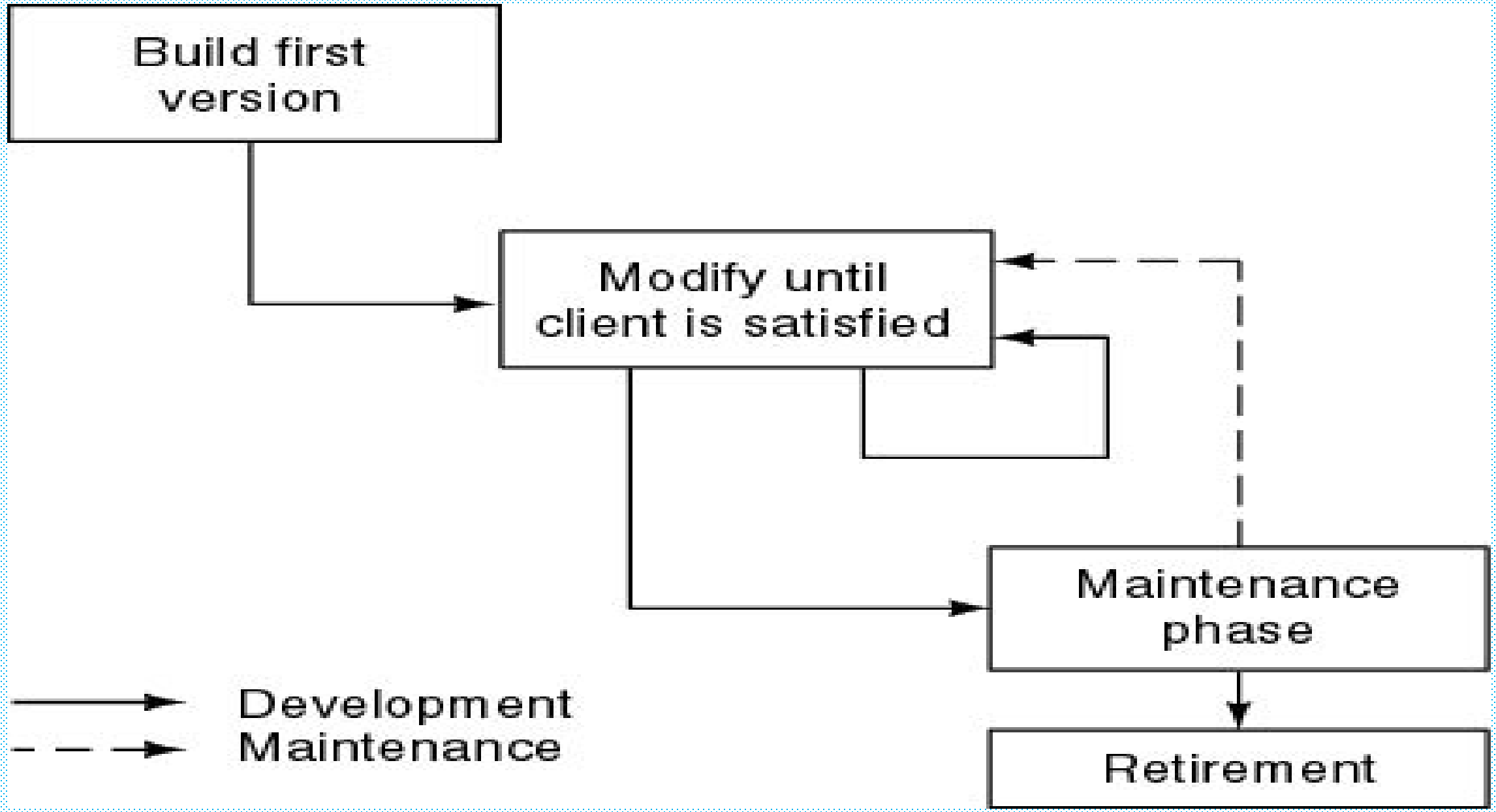


Process Framework

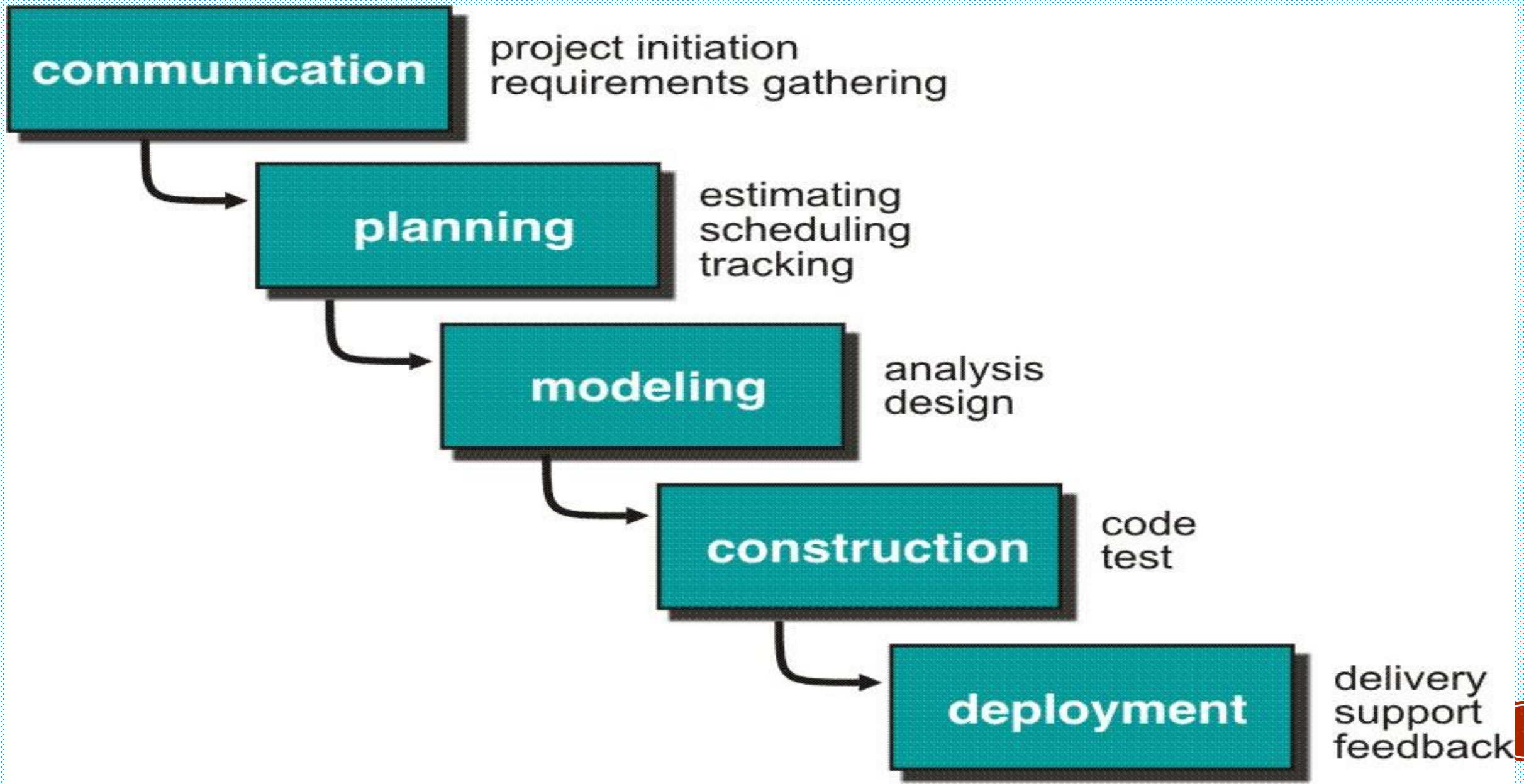
Umbrella Activities

Framework activities
- work tasks
- work products
- milestones & deliverables
- QA checkpoints

BUILD AND FIX MODEL



WATERFALL MODEL



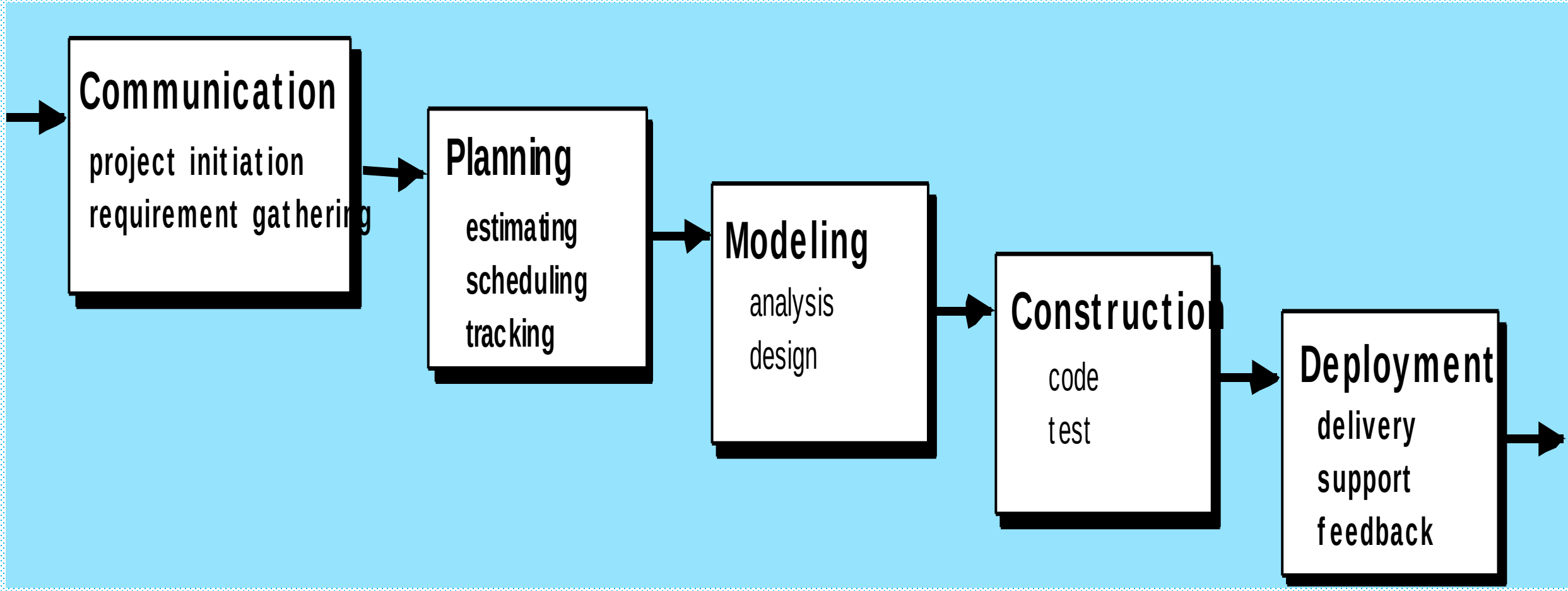
WATERFALL MODEL

- It is linear sequential model
- It follows systematic sequential approach
- It is oldest model
- Requirements has to be well defined

Drawbacks:

- Customer should be patient
- Difficult to follow sequential approach
- Customer not satisfied will be a problem
- Results in blocking states where one has to wait for other.
- Real projects are rarely follow the sequential model.

WATERFALL MODEL



WATERFALL MODEL

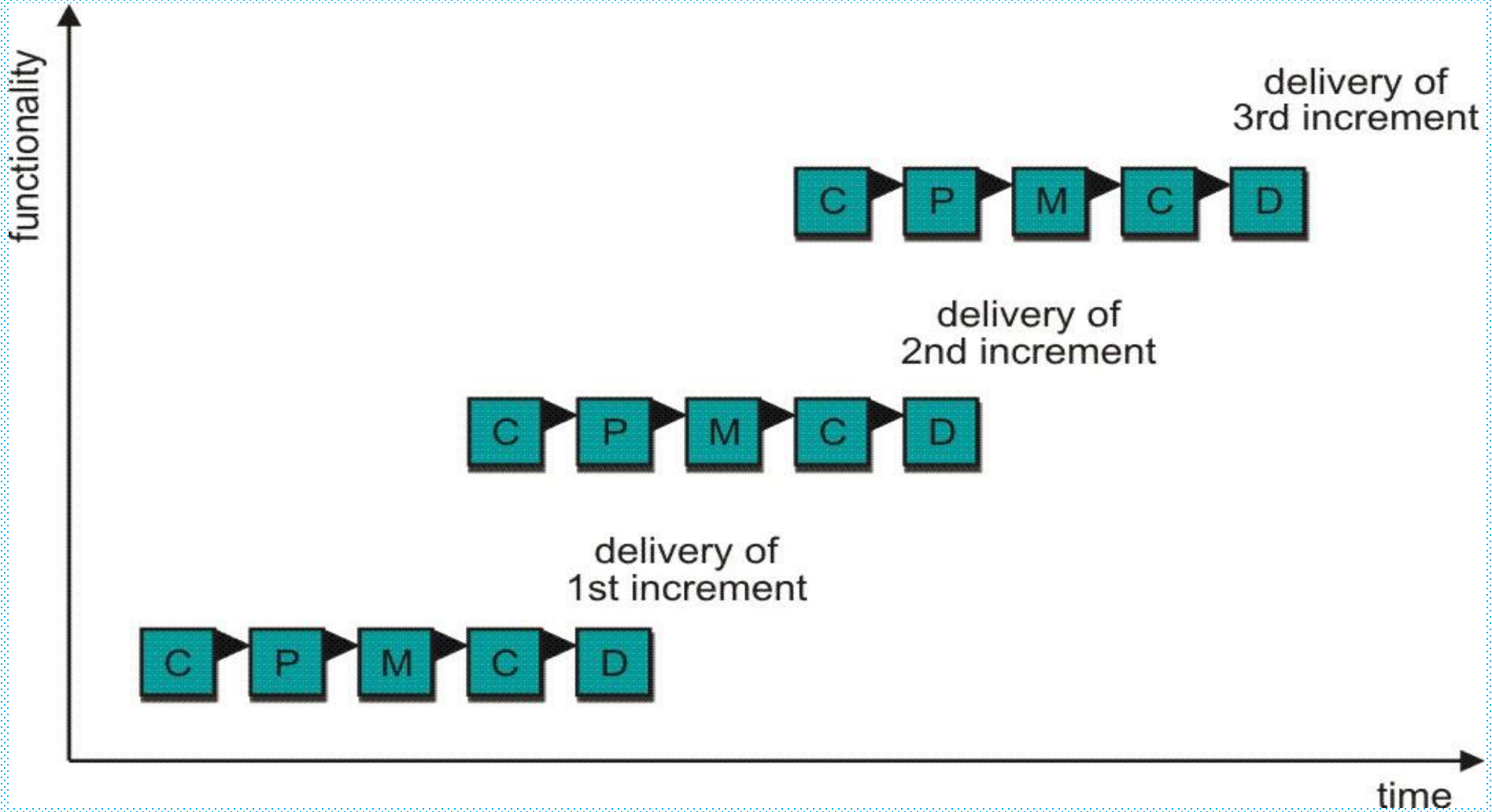
- **Requirement Analysis :** The systems services, constraints and goals are defined by customers with system users.
- **Scheduling tracking** -
  - Assessing progress against the project plan.
  - Require action to maintain schedule.
- **System and Software Design: How** –It establishes and overall system architecture. Software design involves fundamental system abstractions and their relationships.

WATERFALL MODEL

- **Integration and system testing:** The individual program unit or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

- **Operation and Maintenance:** Normally this is the longest phase of the software life cycle. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life-cycle.

INCREMENTAL PROCESS MODEL



C- Communication
P - Planning
M – Modeling
C - Construction
D - Deployment

Delivers software in small but usable pieces, each piece builds on pieces already delivered

THE INCREMENTAL MODEL

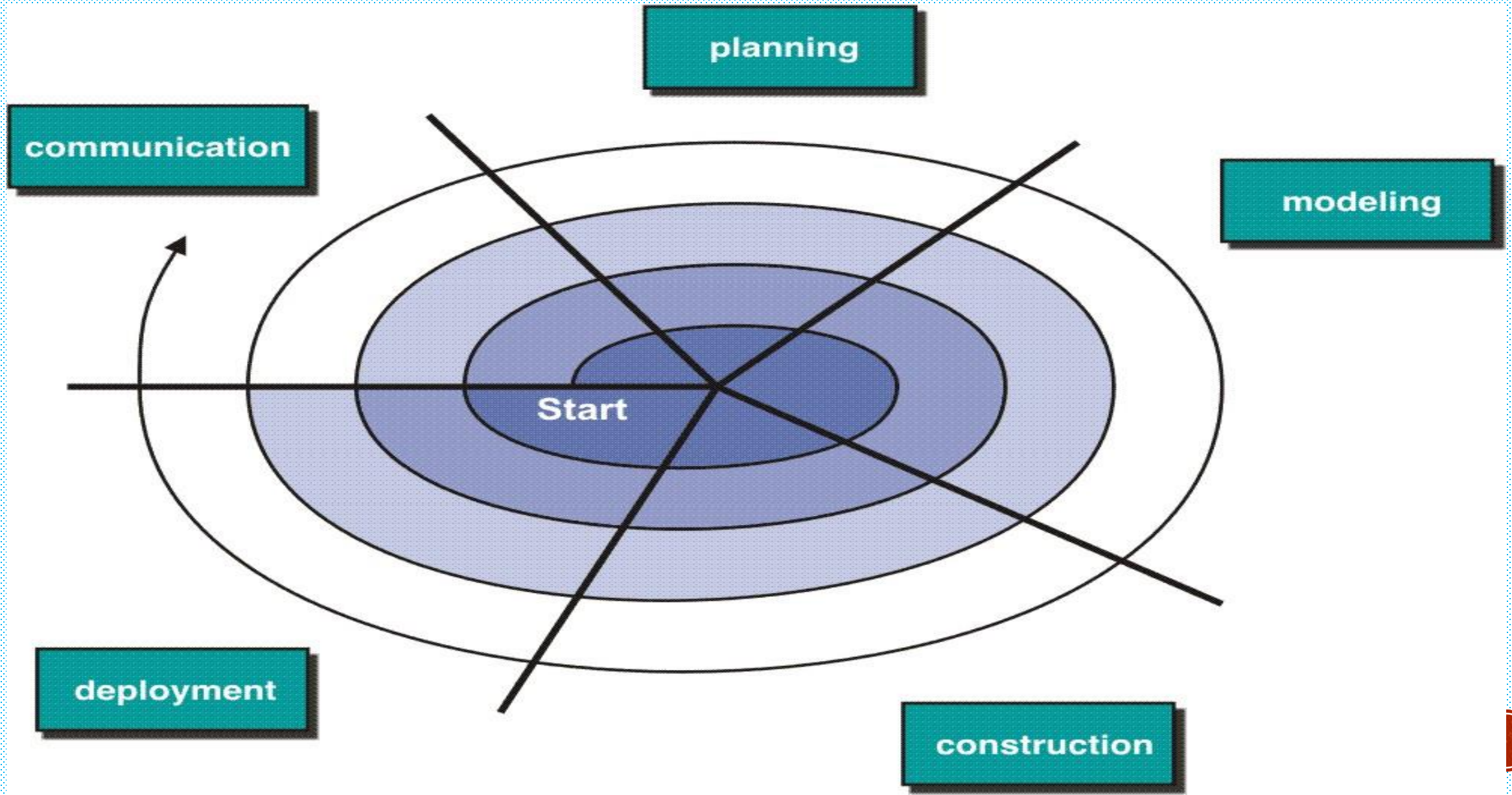
- Linear model in staggered fashion
- It follows iterative approach
- Same phases as in waterfall model
- Increment: Series of releases delivered to customer
- More functionality is added to each release
- First release is basic core product
- New requirements are added to each release
- We choose this model when overall scope is limited & provides limited functionality

THE INCREMENTAL MODEL

Merits:

- Less people are involved
- Risks are managed
- Small span project can be developed
- **Eg:** Word processing package:
- 1st increment: Document processing facilities
- 2nd increment: file processing capabilities are available
- 3rd increment: Spelling facilities are available

EVOLUTIONARY MODEL: SPIRAL MODEL



SPIRAL MODEL

- This has prototype & waterfall model
- Using spiral, software developed in as series of evolutionary release.
  - Early iteration, release might be prototype.
  - Later iteration, more complete version of software.
- Divided into framework activities (C,P,M,C,D). Each activity represent one segment.
- Evolutionary process begins in a clockwise direction, beginning at the center risk.

SPIRAL MODEL

- Early stage is a prototype
- Last stage is a complete software
- Moves in clockwise direction
- 1ST Pass: Product Specification
- 2nd pass: prototype
- 3rd pass: more sophisticated software

SPIRAL MODEL

Customer communication: it is established

Planning: planning is done, resource time ,schedule

Risk analysis: technical & management risks are calculated

Engineering: tasks required to build one /more applications are carried out

Construction & release: tasks required to construct ,test & install the application

Customer evaluation: customer feedback is obtained based on customer evaluation

SPIRAL MODEL

Advantages of spiral model:

- Requirements can be identified at every stage
- Risks can be identified & rectified

Drawbacks: If Communication Is Not Proper Then It Fails