

# CONCEPT OF PROGRAMMING LANGUAGES

# Outline of Presentation:

- Computer Programming Concepts (Basic)
- What is SDLC
- Pros & Cons of SDLC
- Importance of SDLC
- SDLC Phase
- SDLC Models
- Persons Involved In SDLC

# About Me:

Microsoft Certified: Azure AI Engineer Associate (Exam Code: AI-102)

Microsoft Certified: Azure AI Fundamentals (Exam Code: AI-900)


Winner of Microsoft Ambassadors' Led Event in Southeast Asia Region




**Emad Adnan**  
has successfully passed all requirements for  
**Microsoft Certified: Azure AI Engineer Associate**

Credential ID: D97706313E975AE5  
Certification number: 964E28-0E41EE  
Earned on: June 23, 2024  
Expires on: June 24, 2025

✓ Online Verifiable



Satya Narayana Nadella



**Status** Active ✓ Online Verifiable

Credential ID: D97706313E975AE5

Certification number: 964E28-0E41EE

Earned on: June 23, 2024

Expires on: June 24, 2025 at 4:59 AM (UTC +05:00)

[View certification page](#)

# Enhancing Presentation Skills



## Certificate of Completion

**Emad Adnan**

has successfully completed the HP LIFE online course

### Effective Presentations

By completing this course, the above-named student has learned new skills including how to tailor information for their audience, how to deliver a persuasive presentation, and how to create an effective and well-designed presentation.

Presented 9/13/2024

A handwritten signature in black ink, appearing to read "Stephanie Bormann".

---

Stephanie Bormann  
Deputy Director, HP Foundation

Certificate serial number: 192d7711-2afe-49fb-a04f-3f6ffe42fc0e



**LET'S GET STARTED!**



# COMPUTER PROGRAM

A computer program is a set of instructions (statements) written in a programming language to solve a particular problem and achieving specific results.

Any task performed by a computer is controlled by a set of instructions that are executed by the microprocessor.

# PROGRAMMING

Computer programming is the process of writing a computer program in computer language to solve a particular problem.

Computer program controls the operation of a computer, and it is developed in a computer language to perform a task.

# PROGRAMMING LANGUAGES IN GENERAL

- Programming languages are tools for communicating instructions to computers using symbols and rules.
- They have specific functions and limited vocabularies compared to human languages.
- Unlike human languages, which are rich and context-dependent, programming languages are designed specifically for constructing computer programs.
- Examples include C, C++, Visual BASIC, and Java.





# CHARACTERISTICS OF PROGRAMMING LANGUAGES

**Input/Output Instructions:** These instructions direct the computer to "read from" or "write to" a peripheral device

**For example,** disk drive or a printer

**Computation Instructions:** These instructions direct the computer to perform arithmetic operations (add, subtract, multiply, divide, and raise a number to a power).

**For example,**  $\text{PAY} = \text{HOURS} \times \text{RATE}$  computes gross earnings for hourly

**Control Instructions:** These instructions can alter the sequence of the program's execution or terminate execution.

**For example,** in an accounts receivable program, different sequences of instructions are executed for customers who pay on time and for those who pay after the due date.

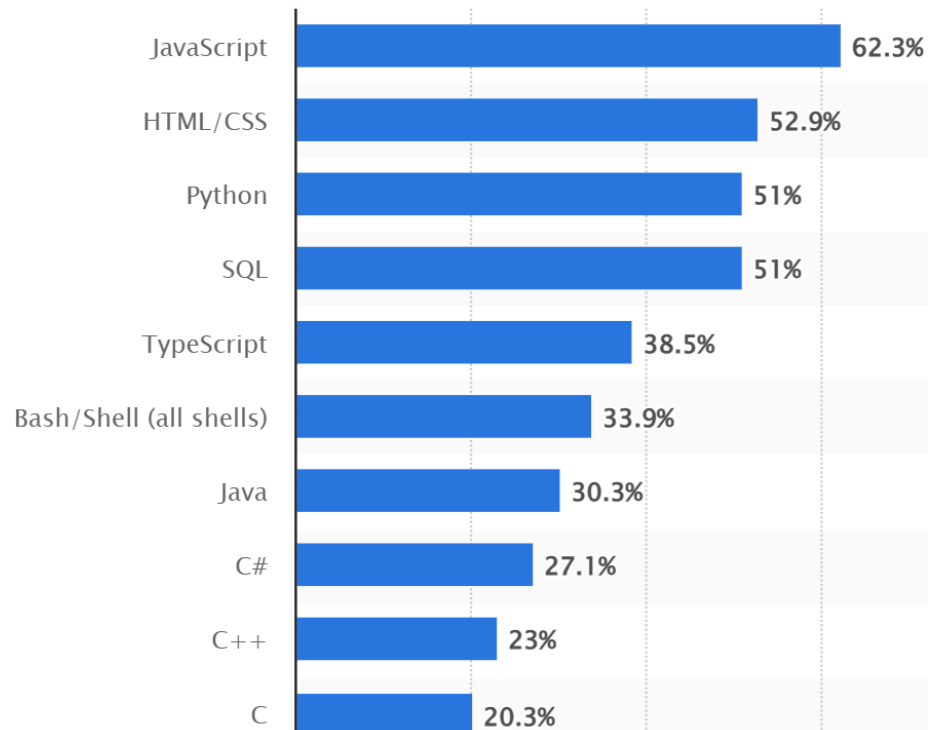
**Assignment Instructions:** These instructions transfer data internally from one RAM location to another.

**For example,** a data item may be copied from one location to another, the result of a computation may be stored at a specified location, or an intermediate result may be retrieved from a given location.

# Popular High Level Languages

Technology & Telecommunications › Software

## Most used programming languages among developers worldwide as of 2024



# C/C++

01

C LANGUAGE WAS DEVELOPED IN EARLY 1970S BY DENNIS RITCHIE AT BELL LABORATORIES.

02

IT IS A HIGHLY STRUCTURED PROGRAMMING LANGUAGE THAT IS EASY TO UNDERSTAND AND USE.

03

IN THE PAST, IT WAS MAINLY USED FOR WRITING SYSTEM PROGRAMS SUCH AS OPERATING SYSTEMS, COMPILERS, ASSEMBLERS, ETC.

04

OBJECT ORIENTED PROGRAMMING WAS INTRODUCED BY C++

# PYTHON

01

IT IS AN  
INTERPRETED  
LANGUAGE  
MEANING IT'S  
EXECUTED LINE BY  
LINE, NOT  
COMPILED.

02

SIMPLE SYNTAX  
USING ENGLISH  
KEYWORDS.

03

VERSATILE  
LANGUAGE USED IN  
WEB DEV, DATA  
SCIENCE, AI, AND  
MORE.

04

FREE TO USE,  
DISTRIBUTE, AND  
MODIFY, WITH A  
LARGE COMMUNITY.

# VISUAL BASIC (VB)

01

Visual basic (VB) is a high-level language which evolved from the earlier version called BASIC.

02

BASIC stands for beginner's All-purpose symbolic instruction code.

03

It provides a graphical development environment to programmers to develop powerful Windows and Web application.



# VISUAL PROGRAMMING

Visual programming is a way to create programs by drawing, pointing, and clicking on diagrams and icons.

It builds on Object-Oriented Programming (OOP) but makes it easier by using visual tools.

The goal is to make programming simpler for both programmers and non-programmers.

Visual programming focuses on solving problems rather than learning complex syntax.

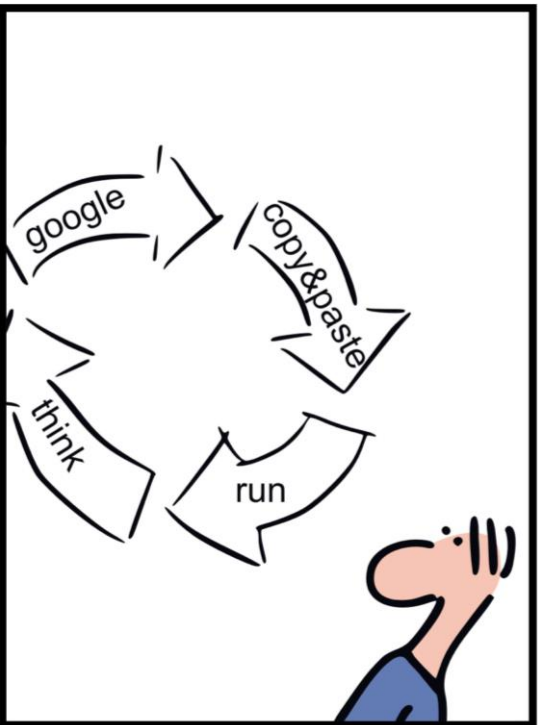
Visual BASIC, created by Microsoft in the early 1990s, is a popular visual programming language.

It provides a visual environment where users can build applications using drag-and-drop tools, buttons, scroll bars, and menus.

Scratch & Visual BASIC works with Microsoft Windows to create other Windows-compatible applications.



# Software Development Life Cycle (SDLC)

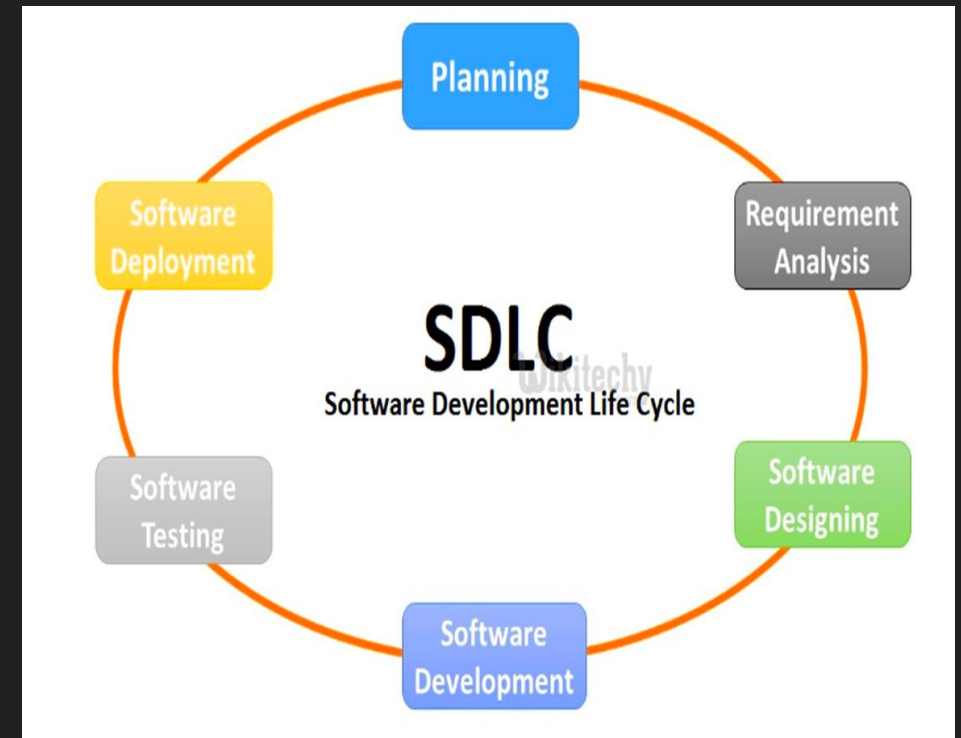


DEVELOPMENT CYCLE

# Software Development Life Cycle (SDLC)

SDLC is a **step-by-step process** that software developers follow to create a software product, from *idea* to *launch*. It's like a **roadmap** that guides the development team through all the stages of creating a software application.

A **structured approach** to planning, designing, developing, testing, and delivering software applications, from initial concept to final deployment, with the goal of ensuring quality, reliability, and customer satisfaction.



# Explanation of what SDLC is:



The process of creating a new software or system. These are the **models and methodologies** that experts use to develop the system.



In software engineering the SDLC concept refers to many kinds of **software development techniques**. These techniques process the framework for planning and controlling the creation of software.



The software development life cycle (SDLC) is the process of planning, writing, modifying, and maintaining software. Developers use the methodology as they design and write modern software for computers, cloud deployment, mobile phones, video games, etc.



In IT, the term "life cycle" was first used in the 1950s to describe the stages involved in developing a new computer system, but it is now commonly used to refer to all the stages in the production of any type of software.

# Stakeholders of SDLC:

Stakeholders of SDLC are those **entities or groups** which are either within the organization or outside of the organization that sponsor, plan, develop or use a project. Stakeholders may be users, managers and developers.

It is the duty of the project management team to **identify the stakeholders**, determine their requirements, expectations and manage their influence in relation to the requirements to ensure a successful project.

## Stakeholders



User  
Secondary



PM  
Primary



Developer  
Primary



Competitor  
?



Customer  
Primary

**Q: Identify the Stakeholders in the College?**



# Persons Involved In SDLC

Management Personnel

Project Manager

System Analyst

Programmer

Software Tester

Customer or End user

# Management Personnel/Team:

A strong management team can satisfy the customers and acquirers of the software system. Also, a proposed project or a product will only meet its objectives if managed properly, otherwise, will result in failure.

## Key Roles:

- i. provide consistency of success of the software regarding time, cost, and quality objectives.
- ii. ensure that customer expectations are met.
- iii. collect historical information and data for future use.
- iv. provide a method of thought for ensuring all requirements are addressed through a comprehensive work definition process.
- v. reduce risks associated with the project.



# Project Manager:

A project manager is a professional responsible for **planning, execution, and closing** of any project. Apart from management skills, a software project manager will typically have an **extensive background** in software development.

He/She is also expected to be familiar with the whole SDLC process.

The key roles of a project manager are:

- i. Developing the project plan
- ii. Managing the project budget
- iii. Managing the project stakeholders
- iv. Managing the project team
- v. Managing the project risk
- vi. Managing the project schedule
- vii. Managing the project conflicts



# Programmer:

A programmer is a **technical person** that writes computer programs in computer programming languages to develop software. A programmer **writes, tests, debugs, and maintains** the detailed instructions that are executed by the computer to perform their functions.

The responsibilities of a programmer includes:

- i. Writing, testing, and maintaining the instructions of computer programs.
- ii. Updating, modifying and expanding existing programs.
- iii. Testing the code by running to ensure its correctness.
- iv. Preparing graphs, tables and analytical data displays which show the progress of a computer program.



## Software Tester:

A software tester is a computer programmer having specialty in testing the computer programs using different testing techniques. Software tester is responsible for understanding requirements, creating test scenarios, test scripts, preparing test data, executing test scripts and reporting defects and reporting results.

## Customer or End User:

A Customer is an individual or a company which buys and uses the system. Customers usually purchase software from software manufacturer companies (software houses), users' groups and individuals. Customers are also called clients but the only difference between the two is that the customers purchase the software products, and the clients purchase services. Customers are the real evaluators of a software product by using it and identifying its merits and demerits.



# Importance of SDLC in Software Development

## **Improved Quality:**

Ensures software meets requirements and is reliable.

## **Increased Productivity:**

Streamlines development process, reducing time and costs.

## **Enhanced Customer Satisfaction:**

Meets customer needs and expectations.

## **Reduced Costs:**

Minimizes rework, delays, and project failures.

## **Better Risk**

**Management:** Identifies and mitigates potential risks.

## **Improved Collaboration:**

Fosters communication and teamwork among stakeholders.

## **Increased Efficiency:**

Optimizes resource allocation and utilization.

# Pros & Cons of SDLC:

## Pros

- Thorough analysis and design
- Well-defined phases to track progress
- Tangible outputs at the end of each phase
- Works well if requirements are well understood
- Only reasonable method in large complex systems.

## Cons

- Lengthy process
- Requirements are frozen and difficult to change
- Requires significant resources in time/people
- Increase cost of project

# Real Life Case Study

The **Sonos app disaster** is the problematic launch of Sonos redesigned mobile app in May 2024. The new app was intended to modernize the user experience but ended up causing **significant issues** due to bugs, missing features, and outdated code. Users reported a loss of functionality, and the app's instability led to **widespread dissatisfaction**.

Sonos had to delay new product releases and faced financial setbacks, including a projected revenue shortfall of **\$200 million**. The company is now working to fix the app and regain customer trust.

08-26-2024 | TECH

## How to avoid the new Sonos app, which everyone hates

The new app has been a disaster for Sonos, which recently laid off staff and cut sales projections for the next quarter.



[Source Photo: [Tim Foster](#)/Unsplash]

BY **JARED NEWMAN** 2 MINUTE READ

# Real Life Case Study

CrowdStrike Outage refers to a major incident in **July 2024** where a faulty software update caused widespread **disruptions for millions** of Windows users.

The update, deployed by the cybersecurity firm CrowdStrike, resulted in systems **crashing and becoming unresponsive**. The outage had a significant impact on businesses and individuals worldwide, leading to widespread frustration and operational challenges.

While CrowdStrike quickly acknowledged the issue and deployed a fix, the incident highlighted the potential risks associated with software updates and the critical role of cybersecurity in modern IT environments.

TOTAL LOSS: \$10bn+



The screenshot shows a web page from The Verge. At the top, the site's logo 'The Verge' is displayed in a large, light blue font. Below it, a navigation bar contains links for 'Tech', 'Reviews', 'Science', 'Entertainment', 'AI', and 'More +'. The article's category is 'MICROSOFT / TECH / SECURITY'. The main headline reads 'CrowdStrike blames test software for taking down 8.5 million Windows machines'. Below the headline is a large image of the CrowdStrike logo, which consists of a stylized 'C' made of three diagonal lines and the word 'CROWDSTRIKE' in bold, black, uppercase letters. To the right of the image, a sub-headline states: '/ CrowdStrike is making improvements to error handling and software rollouts.' At the bottom right, the author information reads: 'By Tom Warren, a senior editor and author of Notepad, who has been covering all things Microsoft, PC, and tech for over 20 years. Jul 24, 2024, 2:33 PM GMT+5'. At the bottom left, there is a small caption: 'Image: The Verge'. At the bottom right, there is a link: '65 Comments (65 New)'.



# 2024 CrowdStrike-related IT outages

🌐

31 languages

▼

Contents

hide

(Top)

Background

- > 

Outage
- > 

Impact
- > 

Response
- > 

Analysis

See also

References

Article

Talk

Read

Edit

View history

Tools

▼

From Wikipedia, the free encyclopedia

On 19 July 2024, American cybersecurity company CrowdStrike distributed a faulty update to its Falcon Sensor security software that caused widespread problems with Microsoft Windows computers running the software. As a result, roughly 8.5 million systems crashed and were unable to properly restart<sup>[1]</sup> in what has been called the largest outage in the history of information technology<sup>[2]</sup> and "historic in scale".<sup>[3]</sup>

The outage disrupted daily life, businesses, and governments around the world. Many industries were affected—airlines, airports, banks, hotels, hospitals, manufacturing, stock markets, broadcasting, gas stations, retail stores, and more—as were governmental services, such as emergency services and websites.<sup>[4][5]</sup> The worldwide financial damage has been estimated to be at least US\$10 billion.<sup>[6]</sup>

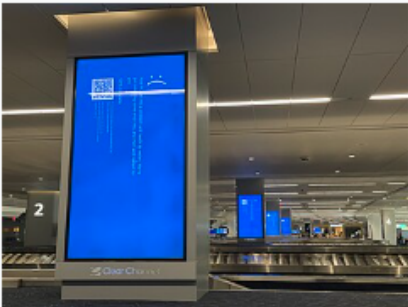
Within hours, the error was discovered and a fix was released,<sup>[7]</sup> but because many affected computers had to be fixed manually,<sup>[8]</sup> outages continued to linger on many services.<sup>[9][10]</sup>

## Background [[edit](#)]

CrowdStrike produces a suite of security software products for businesses, designed to protect computers from cyberattacks. Falcon, CrowdStrike's Endpoint detection and response agent, works at the operating system kernel level on individual computers to detect and prevent threats.<sup>[11]</sup> Patches are routinely distributed by CrowdStrike to its clients to enable their computers to address new threats.<sup>[12]</sup>

CrowdStrike's own post-incident investigation identified several errors that led to the release of a fault update to the "Crowdstrike Sensor Detection Engine":<sup>[13]</sup><sup>*[non-primary source needed]*</sup>

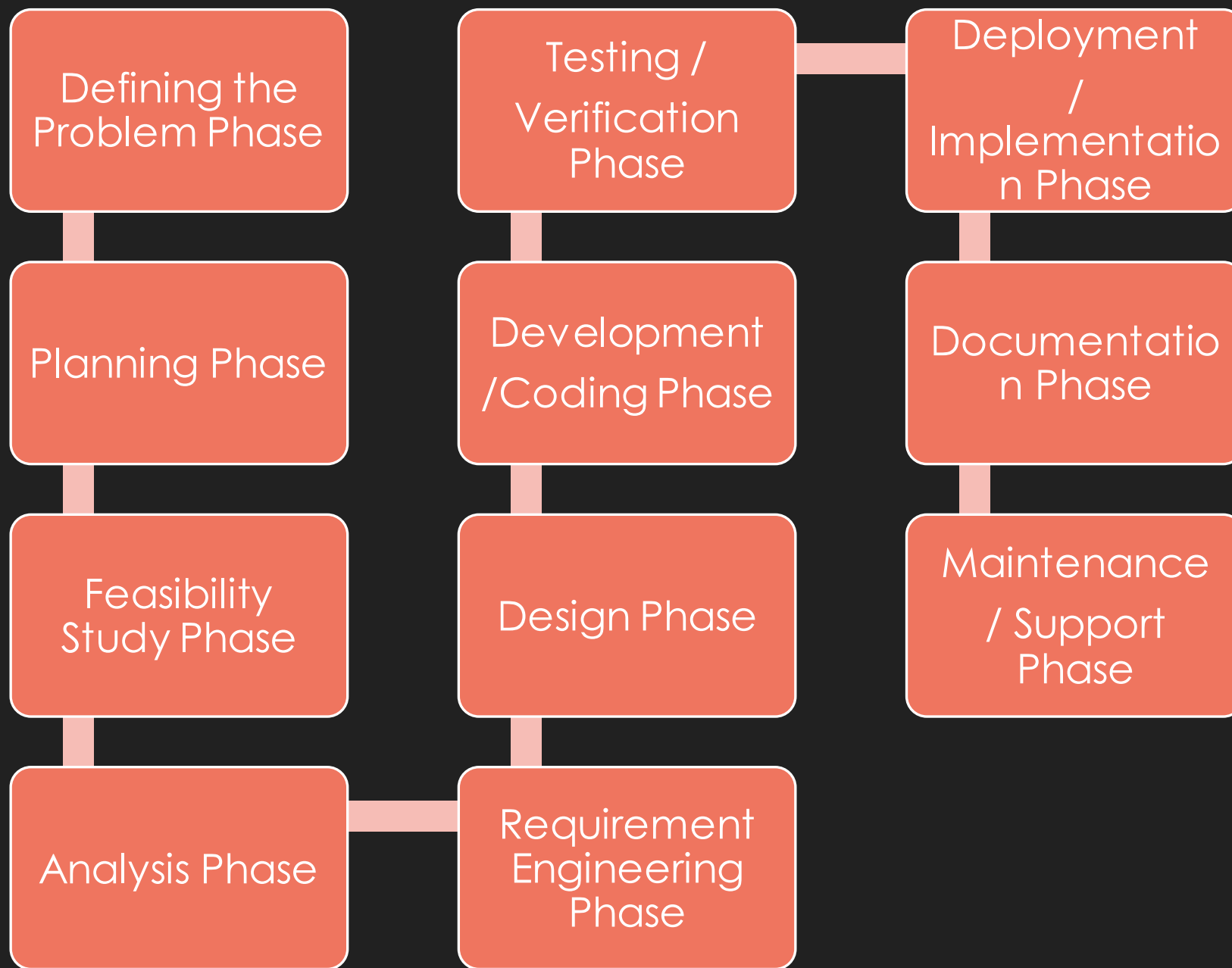
## 2024 CrowdStrike-related IT outages



Multiple blue screens of death caused by a faulty software update on baggage carousels at LaGuardia Airport, New York City

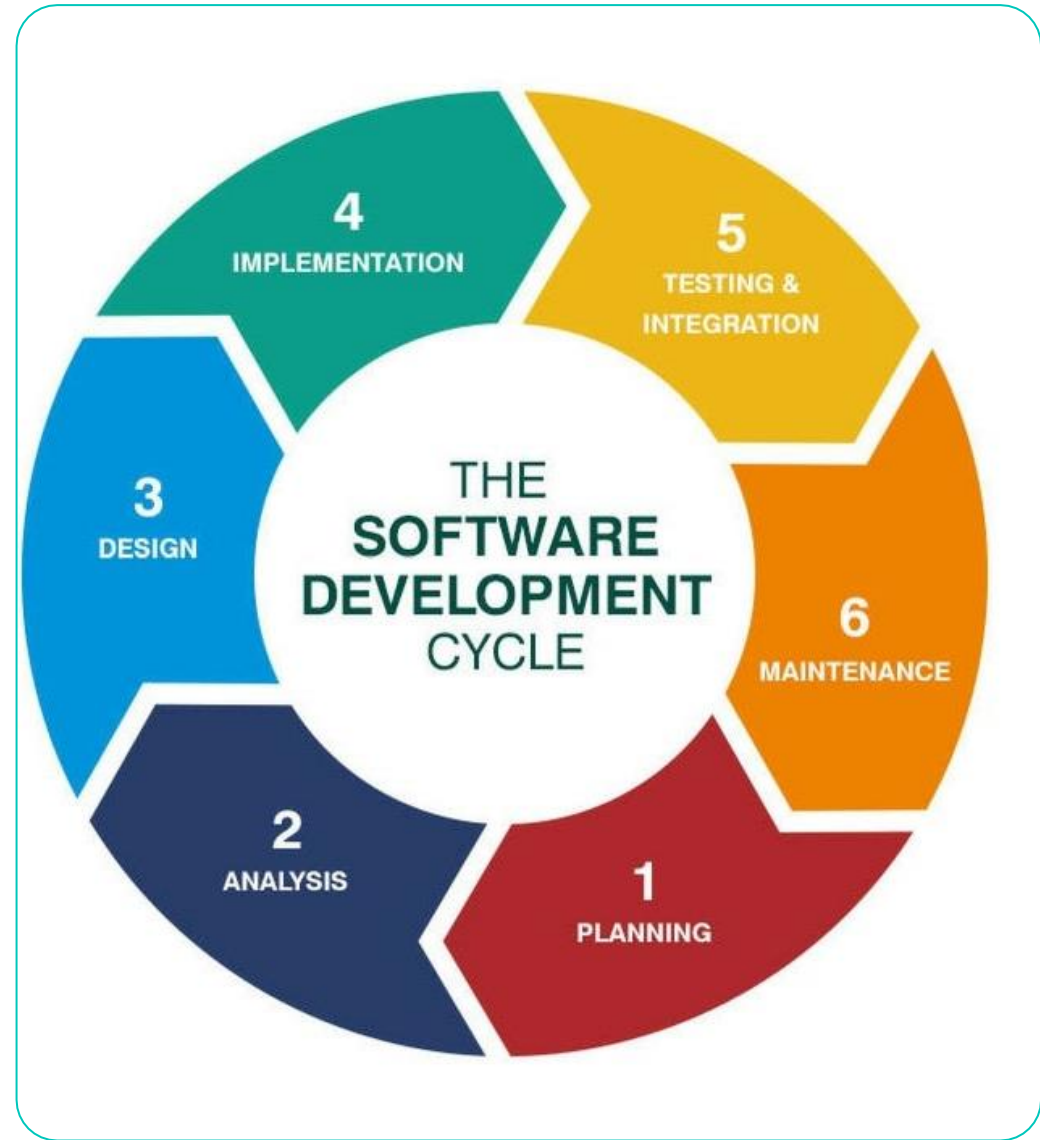
<b>Date</b>	19 July 2024
<b>Location</b>	Worldwide
<b>Also known as</b>	Y2K24
<b>Type</b>	IT outage, computer crash
<b>Cause</b>	Faulty CrowdStrike software update
<b>Outcome</b>	~8.5 million Microsoft Windows operating systems crash worldwide, causing global disruption of critical services





Phases of SDLC

# Phases Of SDLC



# Phase One: DEFINING THE PROBLEM PHASE

- ❑ **Purpose:** Clearly define the problem or system to be developed.
- ❑ **Decision:** Document and approve all requirements from the customer or company.
- ❑ **Evaluation:** Ensure all product requirements are designed and developed during the development life cycle.
- ❑ **Input:** Customer requirements, existing system documentation.
- ❑ **Output:** Approved requirements document.

## Case Study

The library of XYZ Elementary School faces challenges in efficiently managing its **book inventory, tracking borrowed books, and providing an organized system** for library staff. There is a need for a comprehensive Library Management System (LMS) to address these **issues and enhance** the overall library experience.

## Phase Two: Planning Phase

- **Purpose:** Identify the project's goal and assess necessary requirements for product development.
- **Decision:** Evaluate resources, including personnel and costs, and conceptualize the new product.
- **Evaluation:** Analyze gathered information to explore potential alternative solutions.
- **Input:** Project goals, resource availability, cost estimates.
- **Output:** Comprehensive project plan presented to management for approval.

### Case Study

A project team is formed, including representatives from the school administration, librarians, and IT professionals. The team establishes **project goals, scope, budget** and a timeline for the development of the LMS. Key **milestones and deliverables** are identified.

# Phase Three: Feasibility Study Phase

- **Purpose:** Evaluate the viability and practicality of developing the proposed system.
- **Decision:** Determine if the system is technically, financially, legally, and operationally feasible.
- **Evaluation:** Analyze and evaluate the proposed system within the estimated cost and time.
- **Input:** Technical, financial, legal, and operational data.
- **Output:** Feasibility study report.

## Case Study

A feasibility study is conducted to **assess the technical, operational, and economic viability** of implementing the LMS. The study concludes that the benefits of the system, including **improved efficiency and user satisfaction**, outweigh the costs.

# Phase Three: Different feasibility Considerations

**Technical Feasibility:** Technical feasibility assesses the practicality of implementing a proposed project from a technological standpoint. It involves evaluating whether the necessary technology (hardware , software), tools and resources are available or can be developed to support the system.

Example: Consider the Students' Examination System Development project

**Technical Feasibility Considerations:** The developing company will evaluate whether the existing infrastructure (hardware/software) can support the proposed system.

**Economic Feasibility:** Economic feasibility evaluates the financial viability of a proposed system by comparing its costs and benefits.

**Operational Feasibility:** Operational feasibility assesses the extent to which a proposed system aligns with the organization's operational processes and goals.

**Legal Feasibility:** Legal feasibility evaluates whether a proposed system complies with applicable laws, regulations, and standards.

**Schedule Feasibility:** Schedule feasibility assesses whether a system can be completed within a specified timeframe.



## Phase Four: Analysis Phase

- ❑ **Purpose:** Determine end-user requirements and analyze the existing system.
- ❑ **Decision:** Decide whether the project should proceed with available resources.
- ❑ **Evaluation:** Assess if the proposed system can be developed with the available resources and budget.
- ❑ **Input:** End-user requirements, existing system analysis.
- ❑ **Output:** Analysis report with suggested improvements.

### Case Study

Detailed analysis is performed to understand the current library processes, identify user needs, and document system requirements. The team conducts interviews with librarians and observes day-to-day operations to gather essential information.

# Software Requirements Specification (SRS)

The SRS is a document that describes the requirements for a software product, including the software, hardware, functional, and network requirements.

It is one of the first stages of the SDLC and is used throughout the entire process.

## Functional requirements

Define the tasks and actions that the software must perform, including design, graphics, and operating system requirements.

*how the system must work*

## Nonfunctional requirements

Identify the attributes of the system or operational environment, such as usability, security, availability, capacity, reliability, and compliance.

*how the system should perform.*

### Software Requirement Specifications



# Phase Five: Requirement Engineering Phase

- **Purpose:** Gather, analyze, document, and manage requirements for the proposed system.
- **Decision:** Validate and manage requirements throughout the software development life cycle.
- **Evaluation:** Ensure requirements accurately reflect stakeholder needs and expectations.
- **Input:** Stakeholder interviews, surveys, observations, document analysis.
- **Output:** Requirements specification document.

## Case Study

Based on the analysis, a detailed list of functional and non-functional requirements for the LMS are compiled. This includes features such as book cataloguing, user management, check-in/check-out functionality, and reporting capabilities.

## Phase Five: Requirement Engineering Steps:

- Requirement gathering
- Requirement validation
- Requirements management

# Phase Five: Requirement Gathering

**Purpose:** Identify and document the needs and expectations of stakeholders.

## Techniques:

1. **Interviews:** Direct conversations with stakeholders to gather information.
  - **Example:** Business analyst conducts interviews with key users and managers for a new CRM system.
2. **Surveys and Questionnaires:** Distributing surveys or questionnaires to collect information from many stakeholders.
  - **Example:** Surveys distributed to students, faculty, and administrators for feedback on a proposed examination system.
3. **Observation:** Observing users in their natural work environment to understand current tasks and identify areas for improvement.
  - **Example:** Observing students, faculty, and administrators during examination times to identify inefficiencies in the existing system.
4. **Document Analysis:** Reviewing existing documentation, reports, and manuals to extract relevant information about the current system or processes.
  - **Example:** Analyzing academic policies, grading criteria, and previous reports for the examination system development project

# User Requirements Specification (URS):

The User Requirements Specification (URS) is created in the requirements gathering phase of the Software Development Life Cycle (SDLC)

A document that outlines the requirements for a product or system from the perspective of the end user

Purpose	Outlines the requirements for a product or system from the end user's perspective
Who writes it	System owner, end users, and quality assurance team
When it's written	Early in the validation process, typically before the system is created
Who uses it	Developers, engineers, and other stakeholders involved in the design, development, and validation of the product or system
What it includes	Functional, operational, and performance requirements, as well as commercial and regulatory requirements



# Phase Five: Requirements Validation

**Purpose:** Ensure that the requirements accurately reflect the needs and expectations of stakeholders.

## **Techniques:**

- **Reviews:** Conducting requirement reviews to check for completeness, consistency, and accuracy.
- **Prototyping:** Creating prototypes to validate requirements with stakeholders.
- **Test Case Generation:** Developing test cases to verify that requirements are testable and meet the specified criteria.
- **Walk-throughs:** Performing walk-throughs to identify any issues or ambiguities in the requirements.

# Phase Five: Requirements Management

**Purpose:** Manage requirements throughout the software development life cycle, ensuring they remain valid and relevant.

## Activities:

- **Tracking Changes:** Monitoring and controlling changes to requirements.
- **Prioritization:** Prioritizing requirements based on their importance and impact on the project.
- **Documentation:** Keeping detailed records of requirements and any changes made.
- **Communication:** Ensuring effective communication with stakeholders to keep them informed about requirement changes and status.

## Phase Six:

# Design Phase

- **Purpose:** Plan system architecture and create detailed specifications based on gathered requirements.
- **Decision:** Use UML and design patterns to create maintainable and scalable software systems.
- **Evaluation:** Ensure design meets requirements and promotes best practices.
- **Input:** Requirements specification document.
- **Output:** System design documents, algorithms, flowcharts.

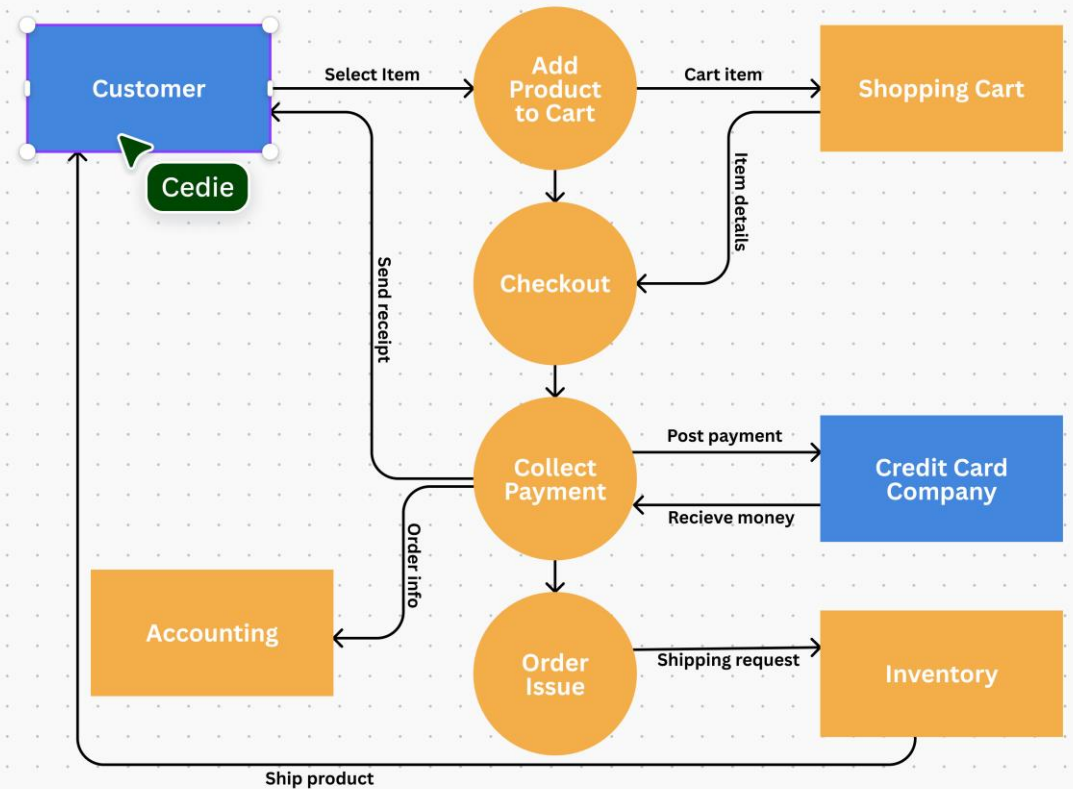
## Case Study

The system architecture and database design are created. User interface prototypes are developed to visualize how the LMS will look and function. The design phase also includes planning for data security, system scalability, and user accessibility.

# Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through a system. It shows the processes, data stores, sources, and sinks within a system. DFDs are used to model the logical flow of data, independent of the physical implementation.

It's a graphical tool that helps project owners conceptualize their projects and think through every important detail.

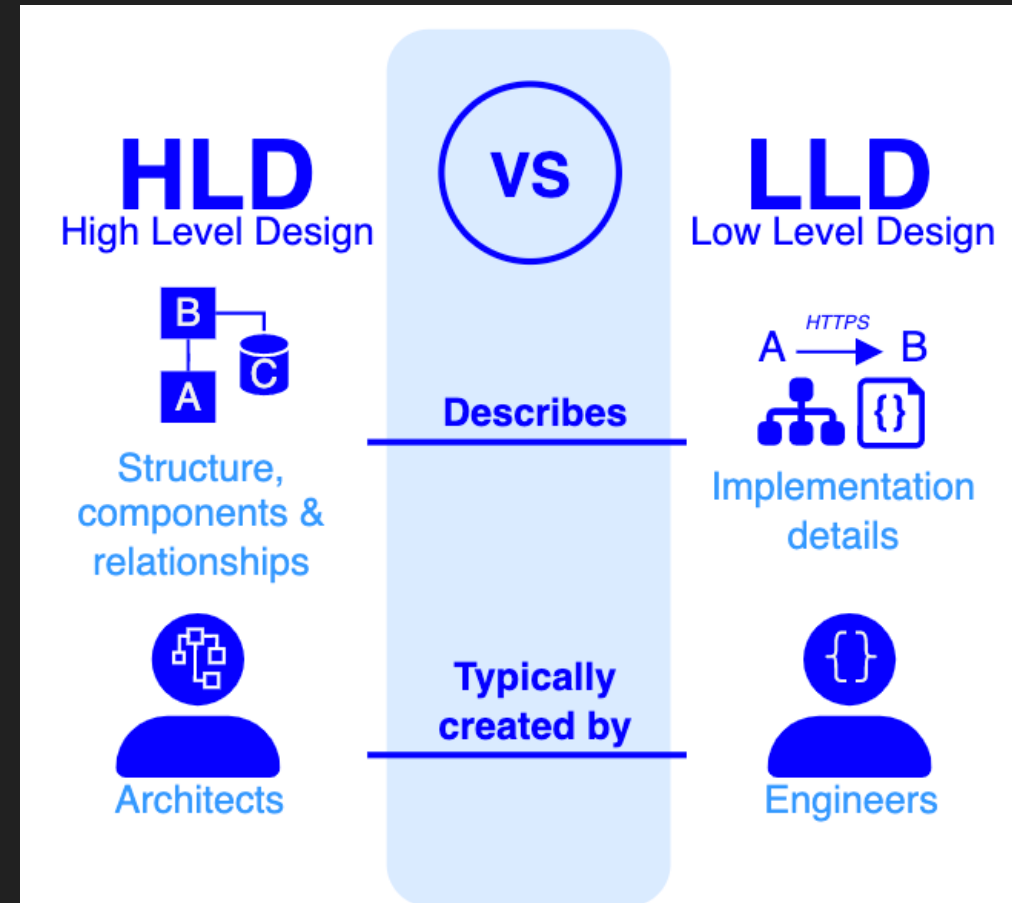


# High-Level Design (HLD)

The first step in the design process and is performed during the early stages of the SDLC. HLD focuses on the system's overall architecture and top-level components. It's also known as **macro level design**.

## Key elements:

- System decomposition into subsystems or modules
- Identification of major components and their responsibilities
- Definition of interfaces between components
- Description of data flow and control flow
- Architectural style or pattern used (e.g. client-server, microservices)

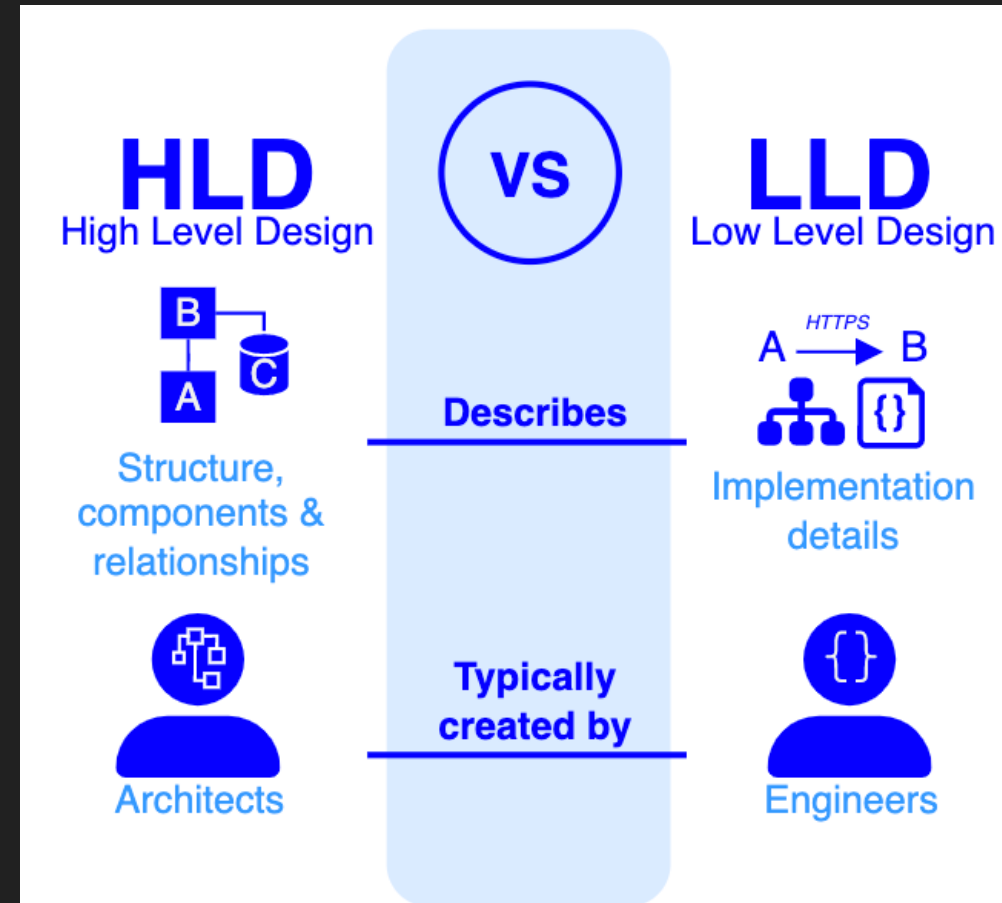


# Low-Level Design (LLD)

This is performed after HLD and focuses on the detailed design of individual components or modules within the system.  
It's also known as **micro level design**.

## Key elements:

- Detailed design of each component's algorithms and data structures
- Specification of classes, methods, and attributes
- Design of data storage and retrieval mechanisms
- Error handling and exception management
- Security considerations





# Phase Seven: Development Phase

- **Purpose:** Translate design plans into actionable code.
- **Decision:** Develop database structures, design codes, and user interface screens.
- **Evaluation:** Enhance precision and efficiency of the written code through iterative processing.
- **Input:** System design documents.
- **Output:** Developed software modules.

## Case Study

The actual coding of the LMS begins, adhering to the design specifications. The development team uses appropriate programming languages and tools to implement the various modules of the system, ensuring that it meets the specified requirements.

# Phase Eight: Testing Phase

- **STLC: Software Testing Life Cycle**
- **Purpose:** Test all aspects of the system for functionality and performance.
- **Decision:** Identify and eliminate bugs through various testing techniques.
- **Evaluation:** Ensure the system aligns with required specifications.
- **Input:** Developed software modules.
- **Output:** Tested and debugged software.

## Case Study

Comprehensive testing is conducted to identify and rectify any bugs or issues. The LMS undergoes unit testing, integration testing, and system testing to ensure its functionality, reliability, and performance meet the desired standards.

# Phase Eight: Types Of Testing

## Black Box Testing:

Black Box Testing is a software testing method where the internal workings or logic of a system are not known to the tester. The focus is on evaluating the system's outputs based on specified inputs without considering its internal code structure.

**Example:** Login Functionality of a Web Application

**Scenario:** Testing the login functionality of a web application.

**Tester's Perspective:** The tester does not have access to the source code or knowledge of the internal implementation details.

### Test Cases:

Input: Valid username and password,

Expected Output: Successful login.

Input: Incorrect password,

Expected Output: Login failure.

Input: Invalid username,

Expected Output: Login failure.

Input: Empty username and password,

Expected Output: Login failure.

# Phase Eight: Types Of Testing

## White Box Testing:

White Box Testing is a testing approach where the tester has knowledge of the internal code, logic, and architecture of the system being tested. It involves evaluating the system's internal structures, code paths, and overall code quality.

**Example:** In the Students' Examination System Development project, the programming module is tested for errors using White box testing.

**Scenario:** Testing a specific function within a software application. Here the students' correct percentage marks and the result is being tested.

**Tester's Perspective:** The tester has access to the source code and understands the internal logic of the function being tested.

### Test Cases:

Test each statement within the program code.

Verify that variables are correctly initialized and updated.

Check for the errors and debug, if needed & evaluate the code's performance under various conditions

# Phase Eight: Types & Methodology Of Testing

## Unit Testing:

Testing individual components or modules of a software application in isolation to ensure they work correctly. This is usually done by developers.

## Integration Testing:

Testing the interaction between integrated modules to ensure they work together as expected. This helps identify issues in the interfaces between modules.

## System Testing:

Testing the complete, integrated system to verify that it meets the specified requirements. This is done in an environment that closely resembles the production environment.

## Acceptance testing:

It is the final and one of the most important levels of testing on successful completion of which the application is released to production. (Alpha/Beta Products).

## End-to-end Tests (E2E Tests):

The process of test the flow of an application to ensure that the entire process of user input and output works smoothly. It involves the entire application and can even extend to testing integrations with external systems.

# Phase Nine: Deployment Phase

- **Purpose:** Make the software/system accessible for use.
- **Decision:** Choose an appropriate deployment method (Direct, Parallel, Phased, Pilot).
- **Evaluation:** Ensure smooth transition from development to utilization.
- **Input:** Tested software, deployment plan.
- **Output:** Installed and activated software system.

## Case Study

The LMS is deployed and integrated into the school's library infrastructure. Training sessions are conducted for library staff to familiarize them with the new system. Data migration from the old system, if applicable, is executed seamlessly.



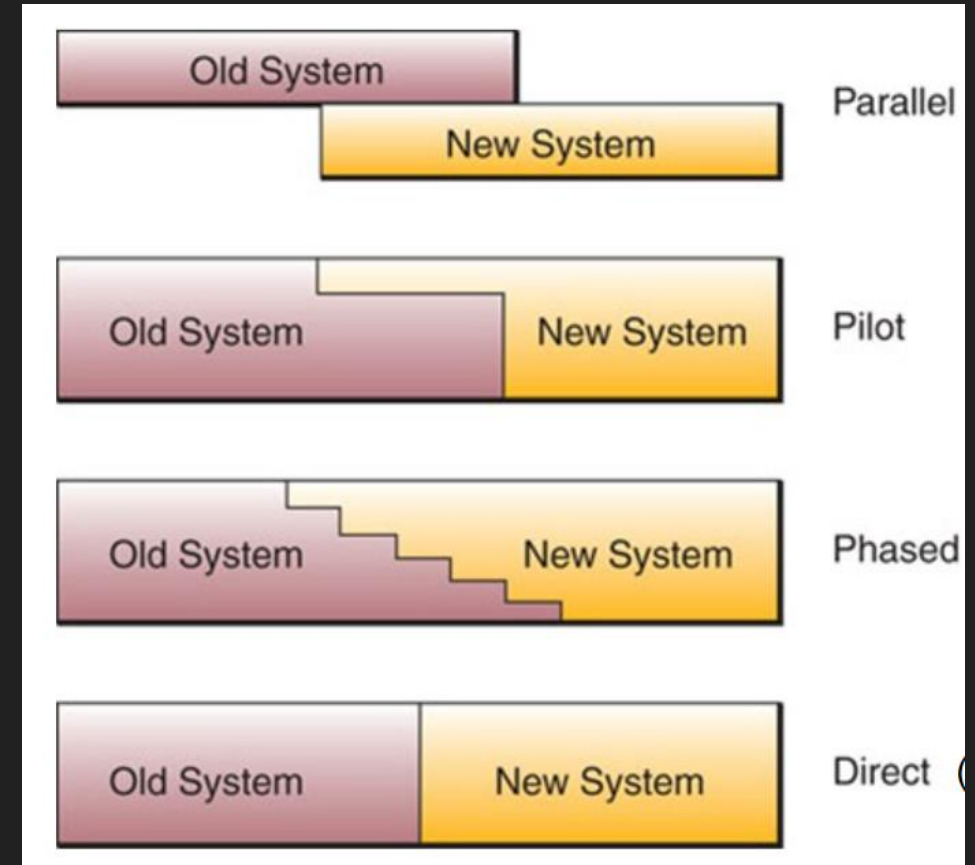
# Phase Nine: Deployment Methodologies

**Parallel:** The parallel method involves running both the old and new systems concurrently for a certain period. This approach allows for the identification and rectification of serious issues with the new system without risking data loss.

**Pilot:** In the Pilot method, the new system is initially deployed for a small user group. These users engage with, assess, and provide feedback on the new system. Once the system is deemed satisfactory, it is then rolled out for use by all users.

**Phased:** The phased implementation method facilitates a gradual transition from the old system to the new one. It involves the incremental introduction of the new system while progressively phasing out the old system.

**Direct:** In this method, the old system is entirely replaced by the new system simultaneously. The transition is abrupt, and once implemented, the old system becomes obsolete.



# Phase Ten: Documentation Phase

- **Purpose:** Develop user documentation for the newly developed system.
- **Decision:** Create detailed records of the system's design and operation.
- **Evaluation:** Ensure documentation aids in diagnosing errors and making changes.
- **Input:** System design and operation details.
- **Output:** User documentation.

## Case Study

Comprehensive documentation, including user manuals, system architecture, and code documentation, is created. This documentation serves as a reference for users and future developers, ensuring the system's maintainability.

# Phase Eleven: Maintenance Phase

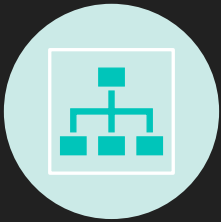
- **Purpose:** Monitor and maintain the system for performance and incorporate needed modifications.
- **Decision:** Identify and implement necessary repairs, modifications, or enhancements.
- **Evaluation:** Ensure the system continues to meet user requirements.
- **Input:** User feedback, performance data.
- **Output:** Updated and maintained system.

## Case Study

Post-implementation support and maintenance are provided to address any issues that may arise. Regular updates and improvements are made to the system based on user feedback and changing requirements, ensuring its continued effectiveness.

# Unified Modeling Language (UML):

A standard modeling language used in software engineering to visualize, specify, construct, and document the artifacts of a software system. It provides a standard way to represent the different aspects of a system, such as its structure, behaviour, and interactions.



## **CLASS DIAGRAMS:**

DEFINE THE  
STRUCTURE OF A  
SYSTEM,  
INCLUDING ITS  
CLASSES AND  
RELATIONSHIPS



## **INTERACTION DIAGRAMS:**

SHOW  
INTERACTIONS  
BETWEEN  
OBJECTS IN A  
SYSTEM, AND  
CAN BE USED IN  
TESTING



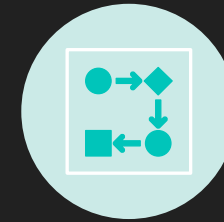
## **USE CASE DIAGRAMS:**

SHOW A  
SYSTEM'S  
ACTORS, USE  
CASES, AND  
INTERACTIONS  
FROM THE USER'S  
PERSPECTIVE



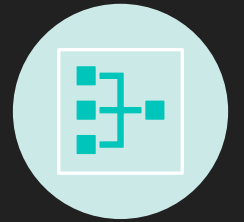
## **SEQUENCE DIAGRAMS:**

SHOW  
MESSAGES OF  
OBJECTS IN A  
SYSTEM OVER  
TIME



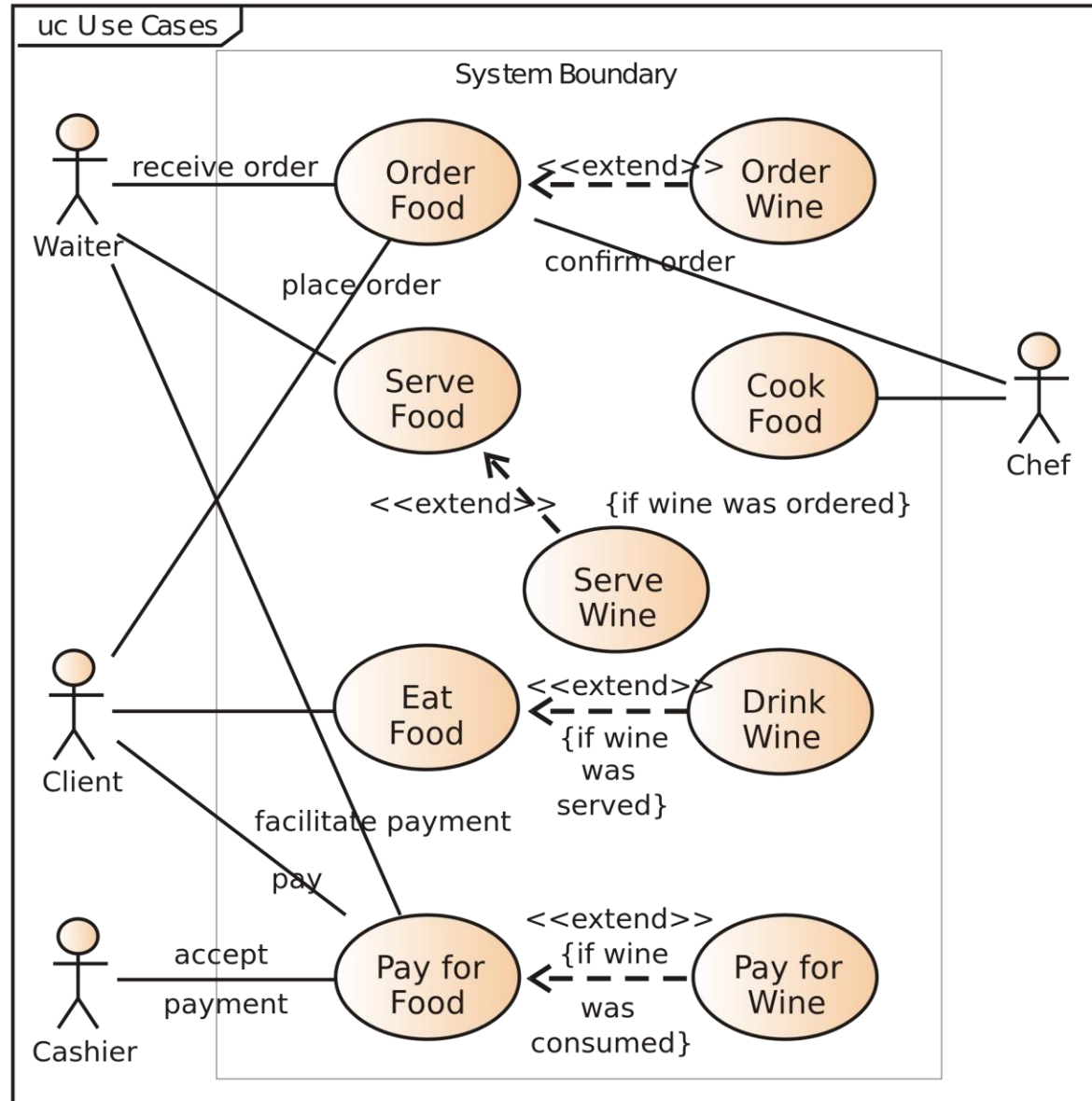
## **ACTIVITY DIAGRAMS:**

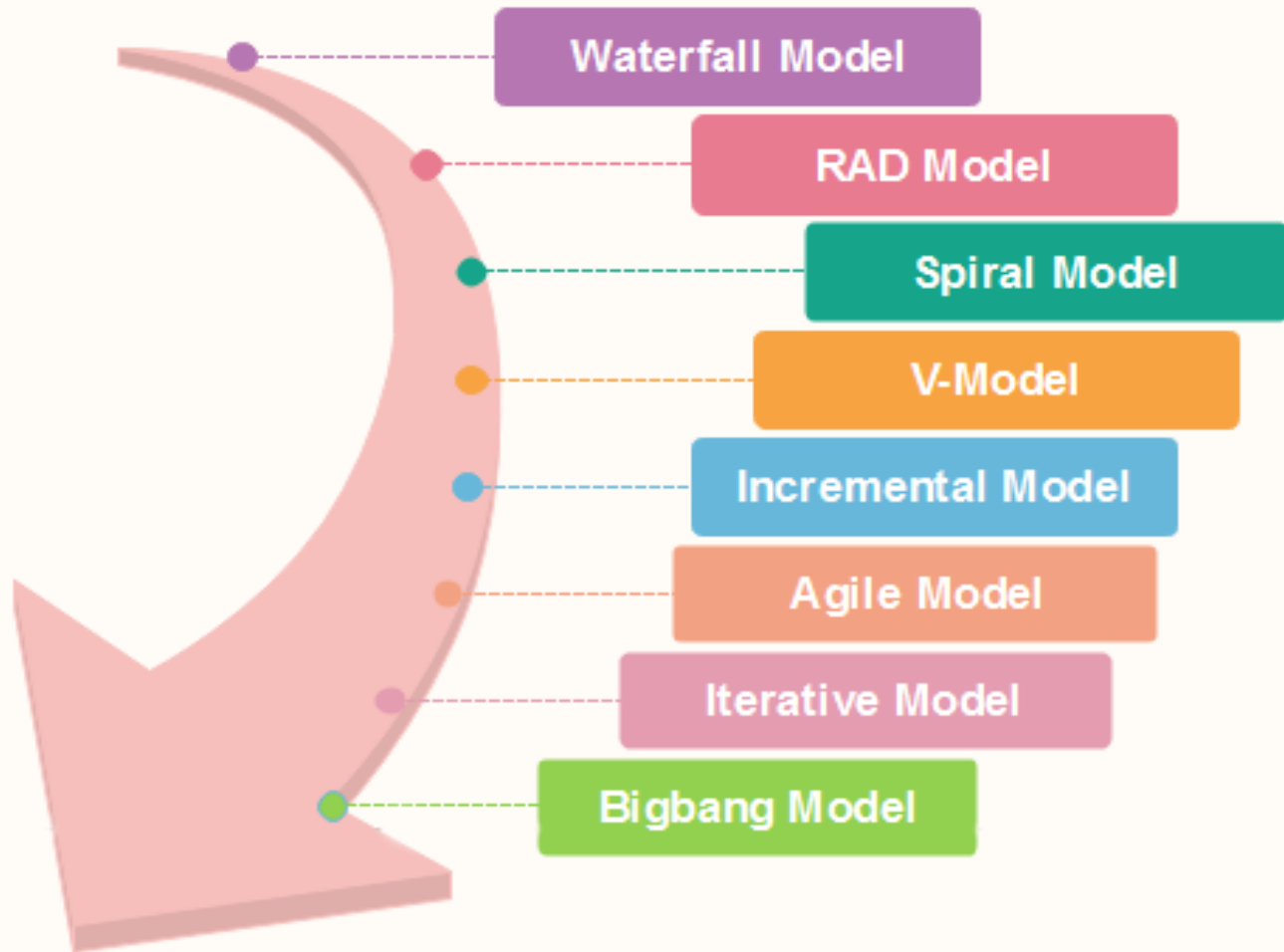
SHOW A  
SYSTEM'S  
ACTIONS,  
FLOWS, AND  
CONDITIONS



## **STATE DIAGRAMS:**

SHOW THE  
STATES,  
TRANSITIONS,  
AND EVENTS OF  
AN OBJECT OR  
SYSTEM





# SDLC MODELS

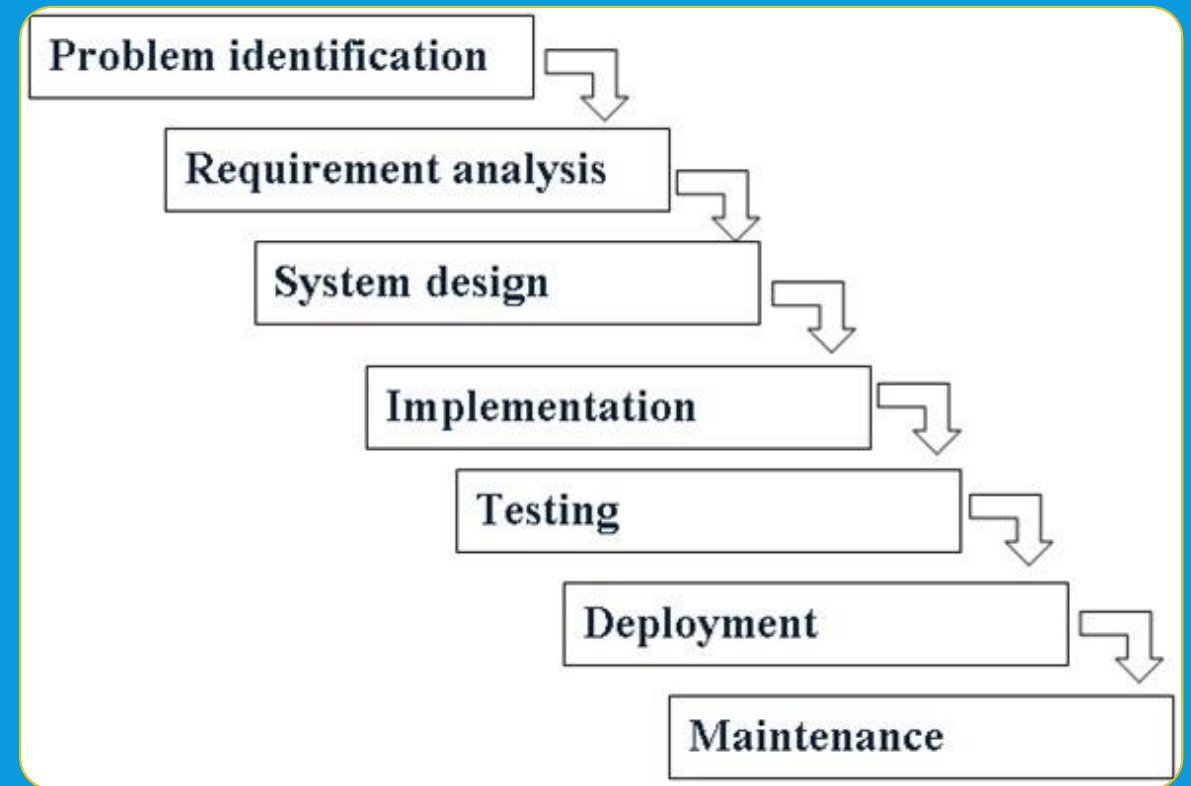
# WATERFALL MODEL

The Waterfall model is a linear and sequential approach to software development in SDLC. It is also known as the "Linear Sequential Model" or "Classic Life Cycle Model".

This SDLC model is the oldest and most forthright. We finish with one phase and then start with the next, with the help of this methodology.

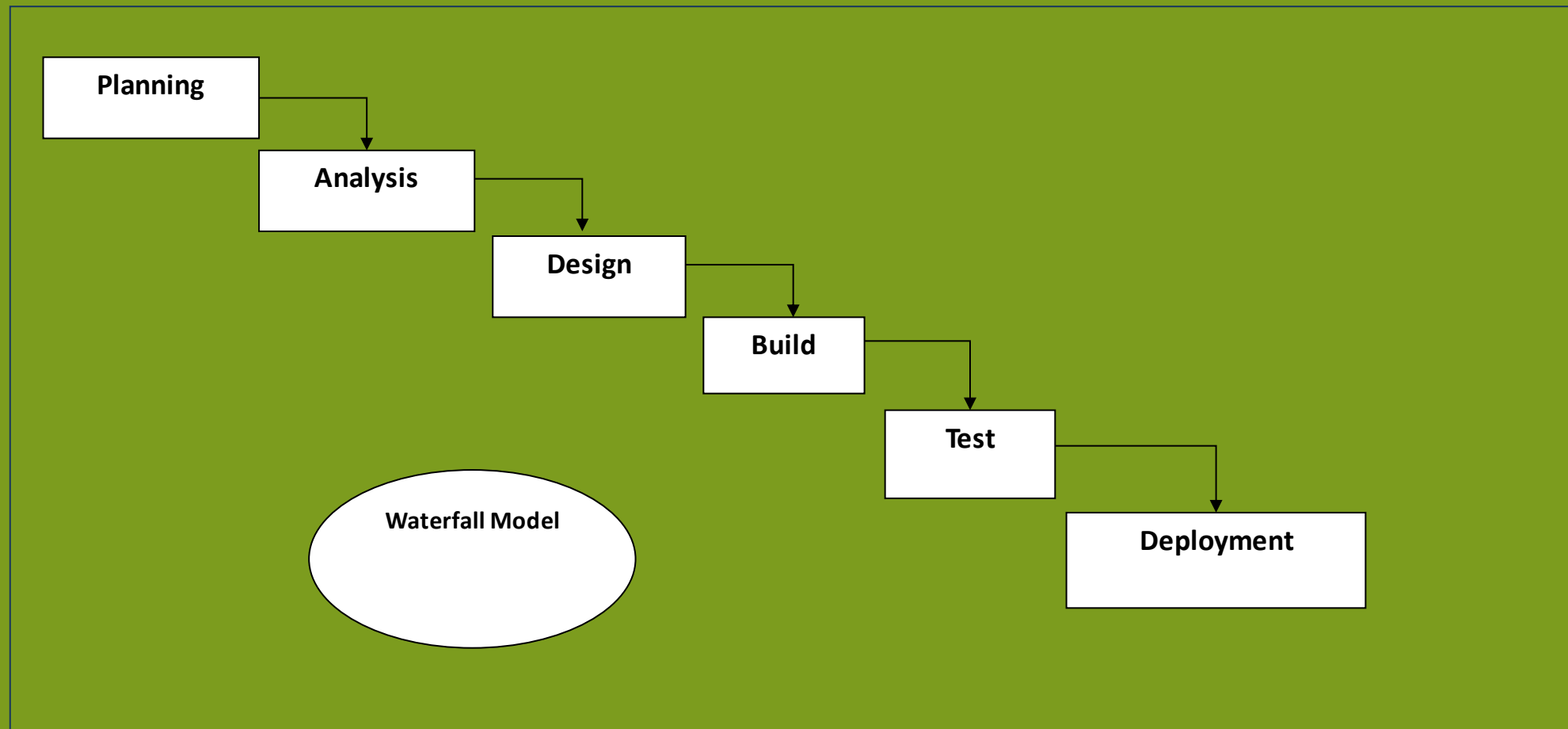
Each of the phases in this model has its own mini-plan and each stage waterfalls into the next.

A drawback that holds back this model is that even the small details left incomplete can hold an entire process.





# WORKING OF WATERFALL MODEL:



# Waterfall

## Advantages



Presence of a clear structure



Smooth transfer of information



Easy to manage



Early determination of goals



Extremely stable

## Disadvantages



Costly and inflexible



Doesn't prioritize the client or end-user



Delayed testing

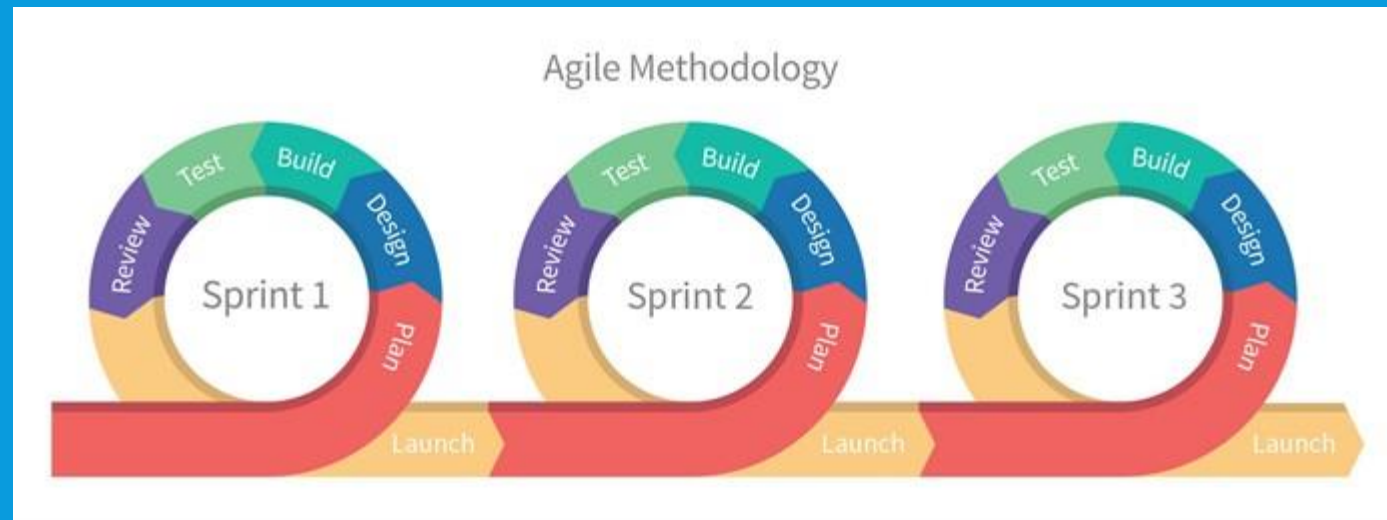


No scope for revision or reflection

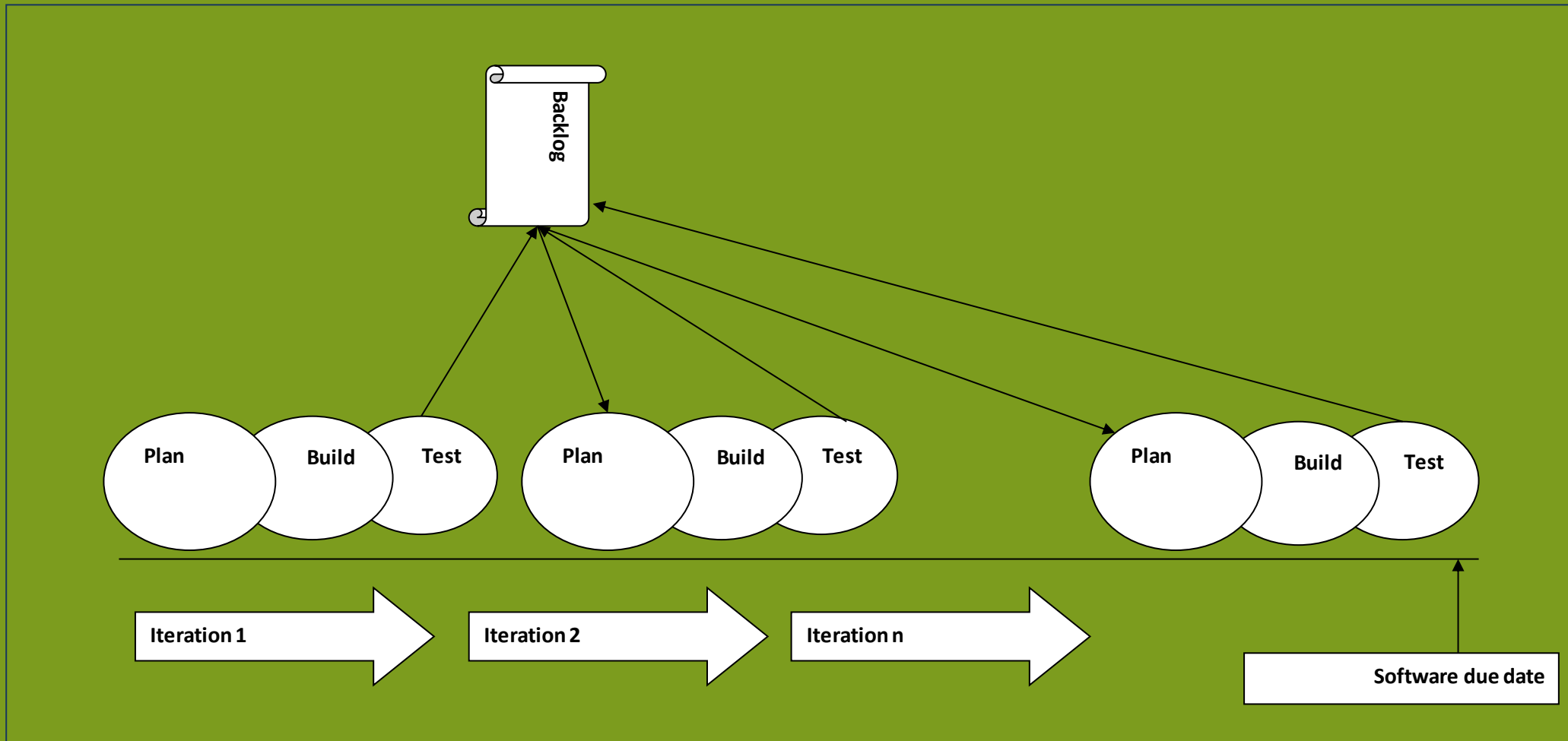
# AGILE MODEL

Agile Model is a flexible software development approach that focuses on iterative development, customer collaboration, and continuous improvement. By breaking projects into smaller sprints and involving customers throughout the process of development.

Agile teams can adapt to changing requirements or requests and deliver working software more efficiently. The main goal of the Agile model is to facilitate quick project completion.



# WORKING OF AGILE MODEL:



## Agile Pros

- Ability to handle shifting priorities
- An improved project's visibility
- Better alignment between IT and business
- Speed of delivery and time to market
- Ease of collaboration
- Increased performance
- Rapid delivery

## Agile Cons

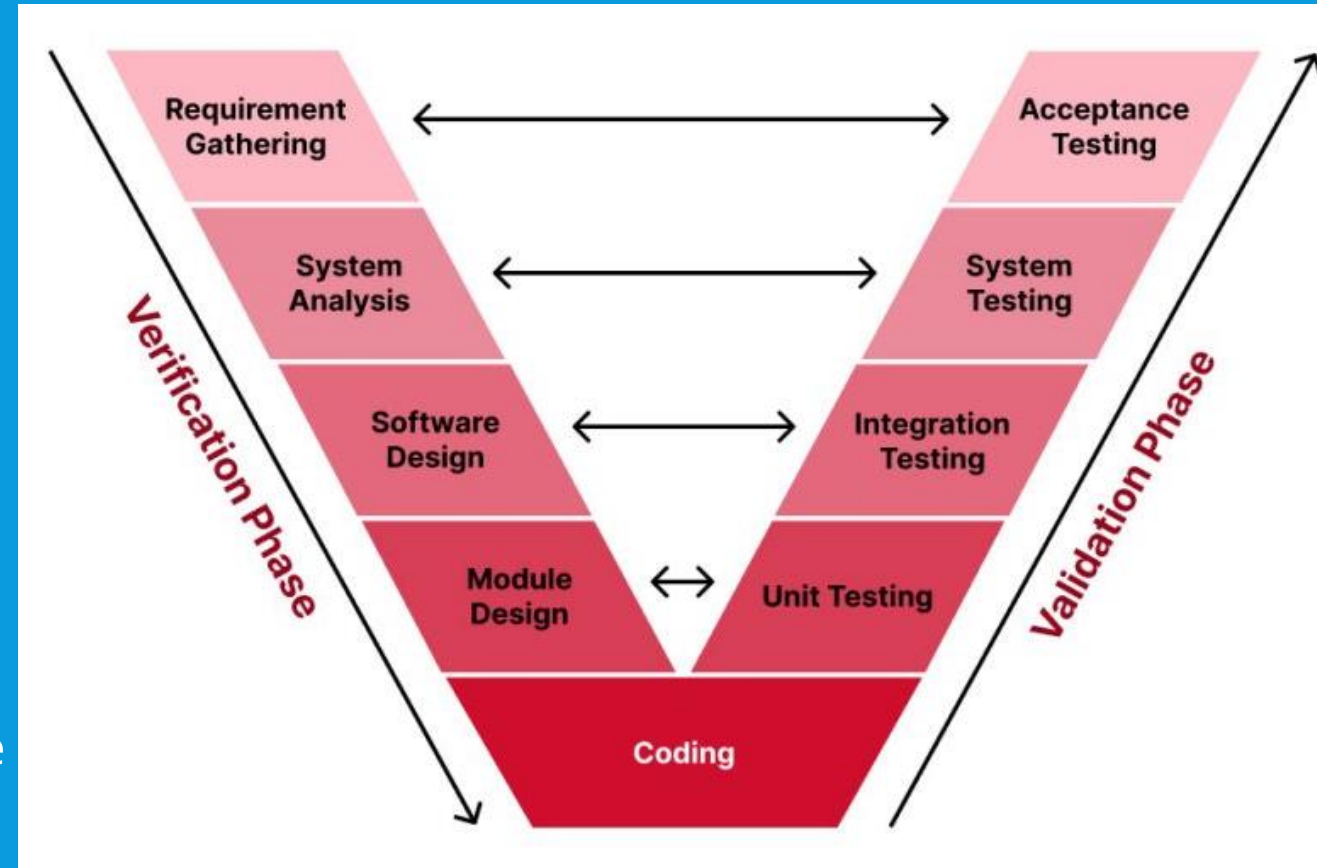
- Companies may be resistant to embracing change
- Teams could engage in inconsistent methods
- Requires assistance from management and leadership
- Agile values and organizational culture may conflict
- Lack of documentation
- Less predictability
- Lack of overall cohesion

# V SHAPED MODEL

A linear software development process that emphasizes early testing. It's shaped like a "V" with the left side representing the development phases (requirements, design, implementation) and the right side representing the corresponding testing phases (unit, integration, system).

This ensures that testing is planned alongside development, leading to higher quality software.

While the V-Model is effective for projects with well-defined requirements, it can be less flexible for those with changing needs.

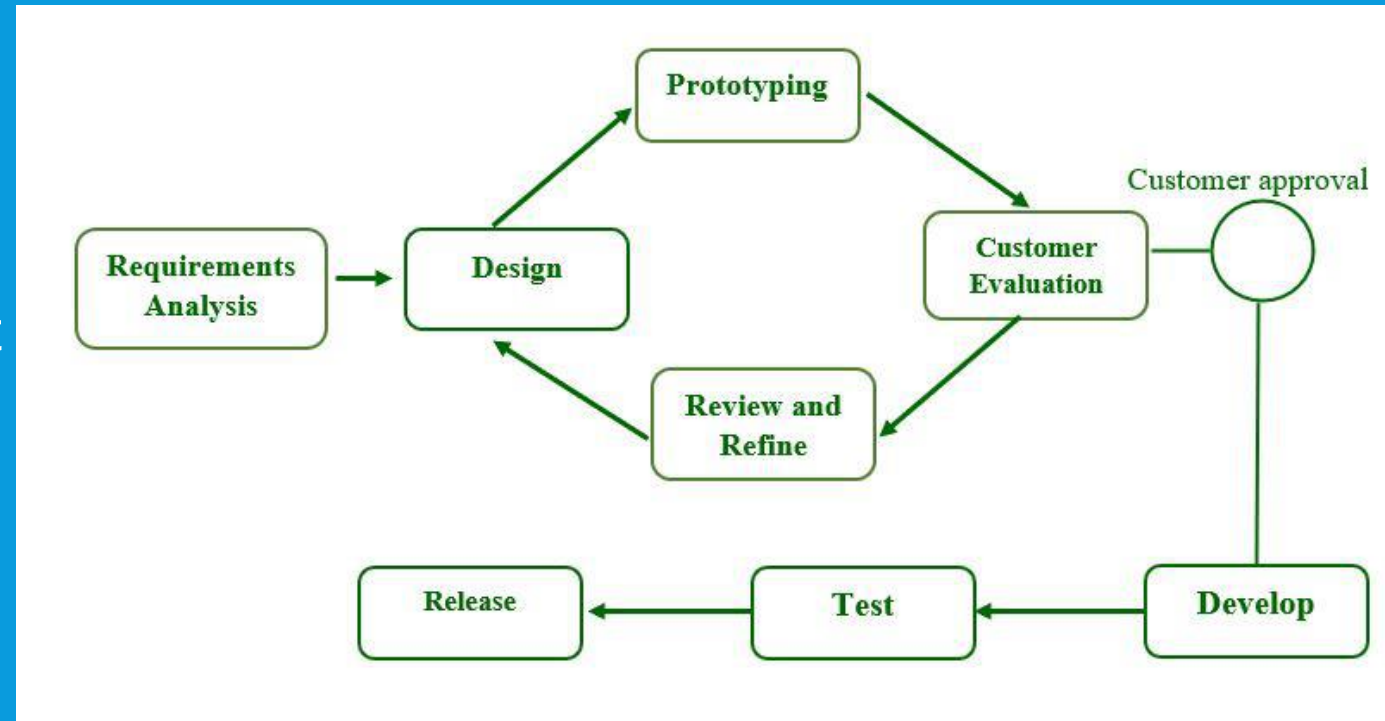


# PROTOTYPE MODEL

**The Prototype Model** is a software development approach that emphasizes early creation of a working model, called a prototype, to gather feedback and refine requirements before full-scale development.

This iterative process helps reduce risks, improve user satisfaction, and ensure the final product meets expectations.

However, it can be time-consuming if not managed effectively and may lead to scope creep if not carefully controlled.



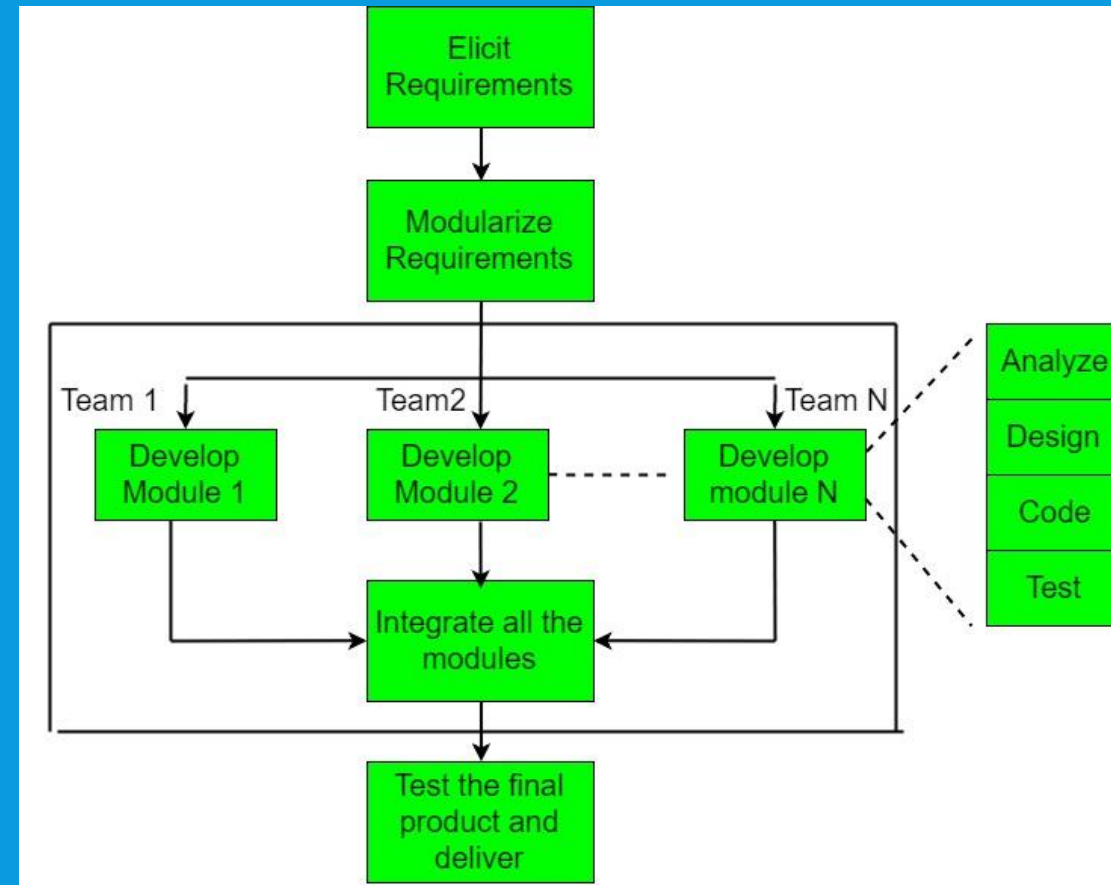


# RAPID APPLICATION DEVELOPMENT MODEL (RAD)

**The Rapid Application Development (RAD) Model** is a time-boxed software development methodology that emphasizes rapid prototyping and iterative development.

It aims to deliver working software quickly by using reusable components and tools, involving end-users early, and minimizing documentation.

RAD is well-suited for projects with well-defined requirements and a need for rapid delivery but may not be suitable for complex projects with uncertain requirements.



# Pros and Cons SDLC framework

SDCL Model	Pros	Cons
Waterfall	Easy to understand and manage due to its linear flow; Clear milestones	Difficulty in going back to a previous phase; Not suitable for complex projects
Agile	Adaptable to changing requirements; Continuous feedback and improvement	Requires experienced team members; Not suitable for small projects
Iterative	Lesser initial delivery cost; Easier to test and debug	Requires more resources; Possible problems can arise linked to system architecture
V-Model	High success rate over the waterfall model; Works well for small projects	Difficult to go back to a previous stage; Not flexible

## Comparison of different SDLC Models

# Conclusion:

- Evaluation

Analyzing the success and scope for improvements in the SDLC process.

- Ongoing Improvements

Identifying areas for enhancement and innovation in future projects.

- Lifecycle Cost Consideration

Lifecycle Cost Consideration in SDLC involves factoring in the total expenses associated with software development, including initial development etc.

THE END.

