Vietnam National University of HCMC
International University
School of Computer Science and Engineering

Object – Oriented Analysis and Design

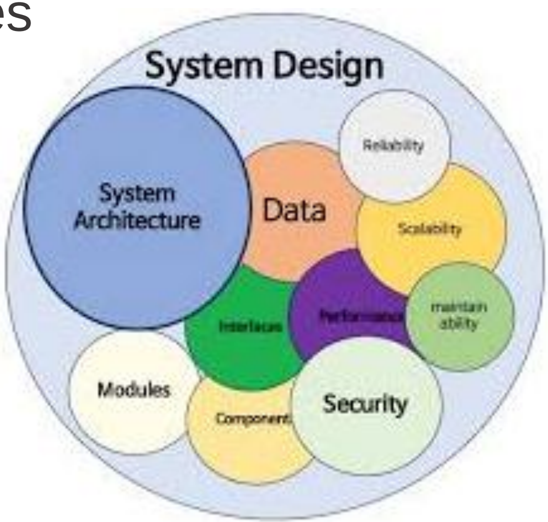
System Architecture & Testing and Quality Assurance

Instructor: Le Thi Ngoc Hanh, Ph.D

ltnhanh@hcmiu.edu.vn

System Architecture Overview

System architecture describes how components in a software system interact, the technologies used, and how it meets the functional and non-functional requirements.



⇨ **What if without system architecture?**

Source image: https://medium.com

System Architecture (SA) in Software Development (SD)

- Ensures Scalability and Performance
- Facilitates Maintainability and Flexibility
- Aligns with Business Goals
- Enhances Team Collaboration
- Mitigates Risks

SA in OOAD

How SA fits within OOAD?

amazon

Layered Architecture

NETFLIX

Microservices Architecture

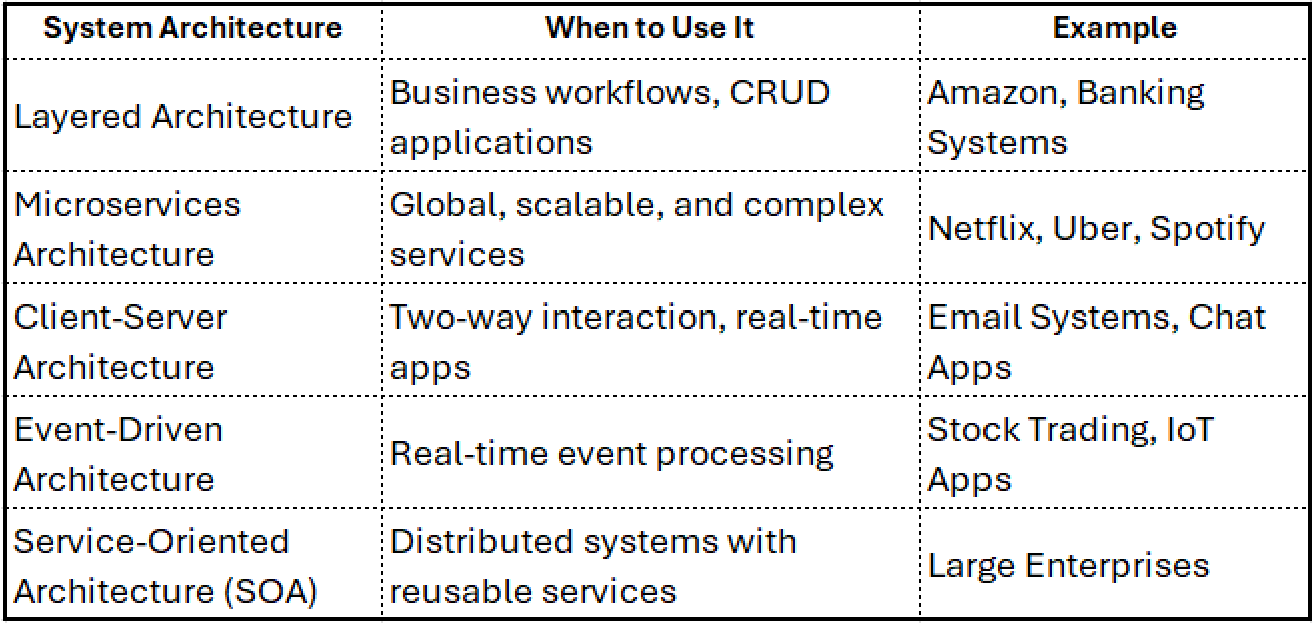
YouTube

Distributed system architecture

X

Event-Driven Architecture

Reference - System Architecture types



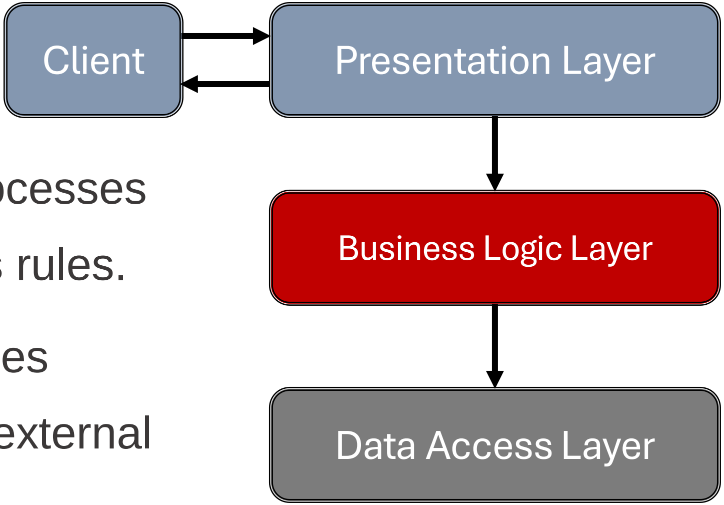
System Architecture	When to Use It	Example
Layered Architecture	Business workflows, CRUD applications	Amazon, Banking Systems
Microservices Architecture	Global, scalable, and complex services	Netflix, Uber, Spotify
Client-Server Architecture	Two-way interaction, real-time apps	Email Systems, Chat Apps
Event-Driven Architecture	Real-time event processing	Stock Trading, IoT Apps
Service-Oriented Architecture (SOA)	Distributed systems with reusable services	Large Enterprises

Key Concepts of System Architecture

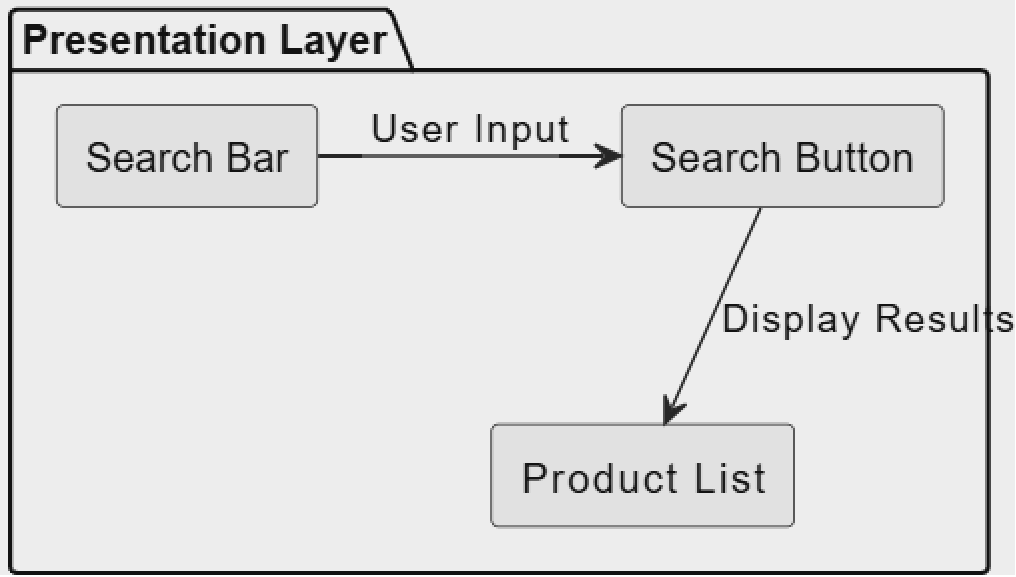
- **Layers of Architecture:** provide the structural framework.
- **Architectural Patterns:** offer reusable solutions tailored to specific needs.
- **Non-Functional Requirements:** ensure the system meets performance and operational goals.
- **Stakeholders and Their Roles:** ensure the architecture serves its intended purpose and satisfies all involved parties.

Layers in Architecture

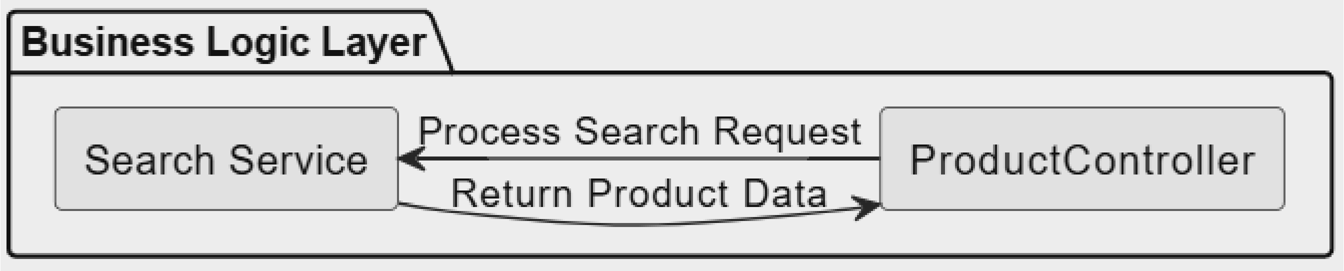
- Presentation Layer: Manages user interface and user interactions.
- Business Logic Layer: Processes data and applies business rules.
- Data Access Layer: Handles database operations and external APIs.



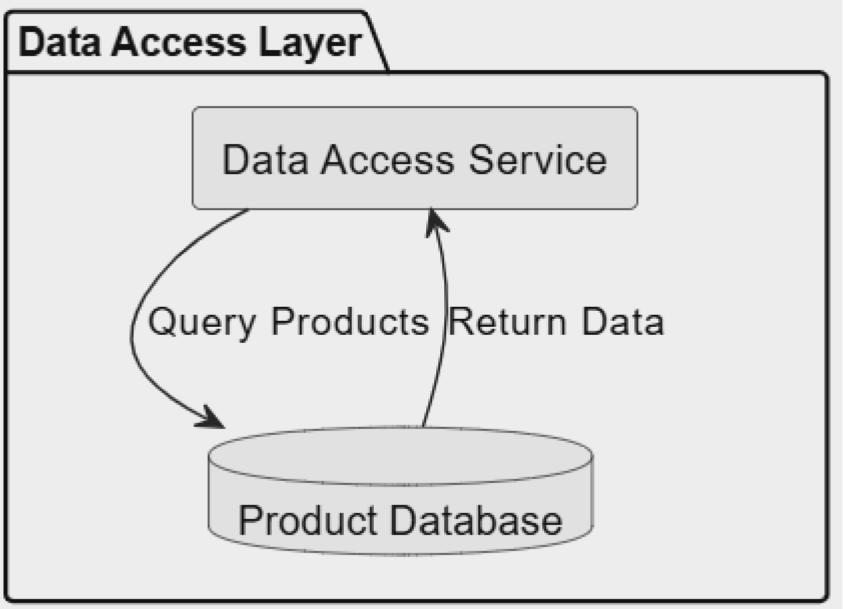
Layers in Architecture

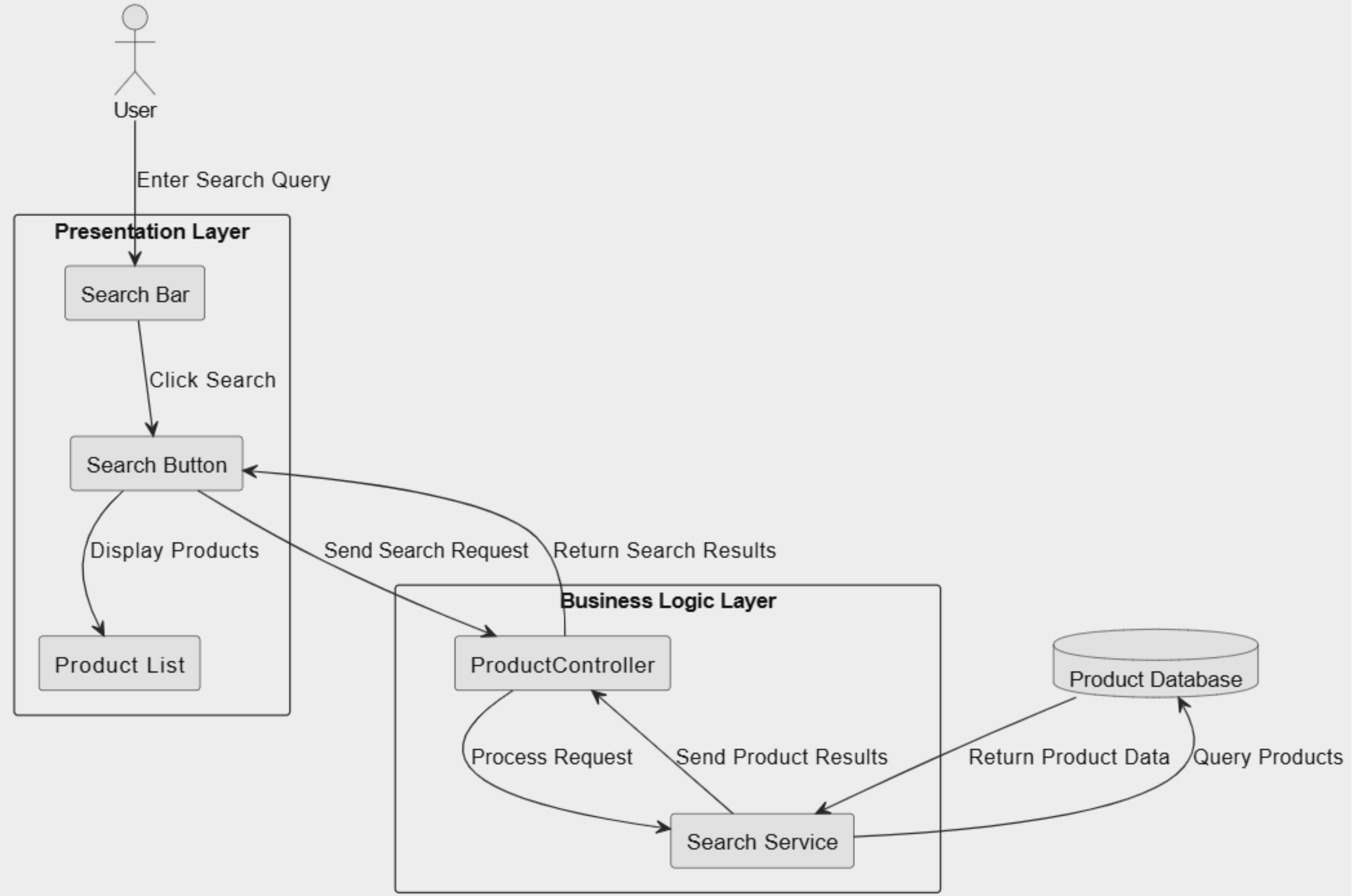


Layers in Architecture

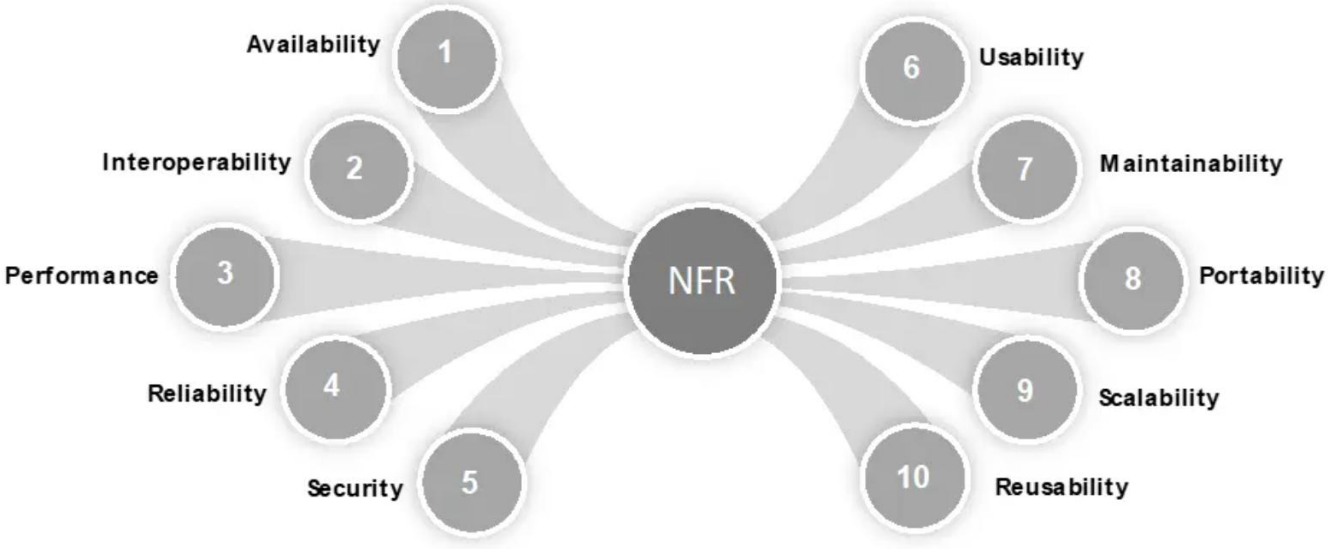


Layers in Architecture




User
Enter Search Query
Presentation Layer
Search Bar
Click Search
Search Button
Display Products
Product List
Send Search Request
Return Search Results
Business Logic Layer
ProductController
Process Request
Send Product Results
Search Service
Product Database
Return Product Data
Query Products

Non-Functional Requirements (NFR)



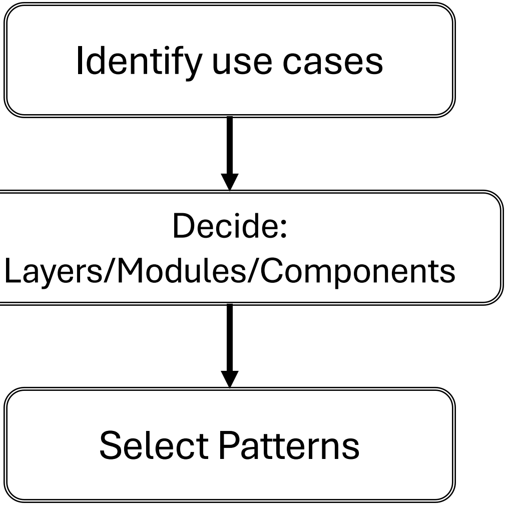
Source image: https://techcanvass.com

Key Concepts of System Architecture

- **Layers of Architecture:** provide the structural framework.
- **Architectural Patterns:** offer reusable solutions tailored to specific needs.
- **Non-Functional Requirements:** ensure the system meets performance and operational goals.
- **Stakeholders and Their Roles:** ensure the architecture serves its intended purpose and satisfies all involved parties.

Designing an Architecture

- Key steps:



Identify use cases

Decide: Layers/Modules/Components

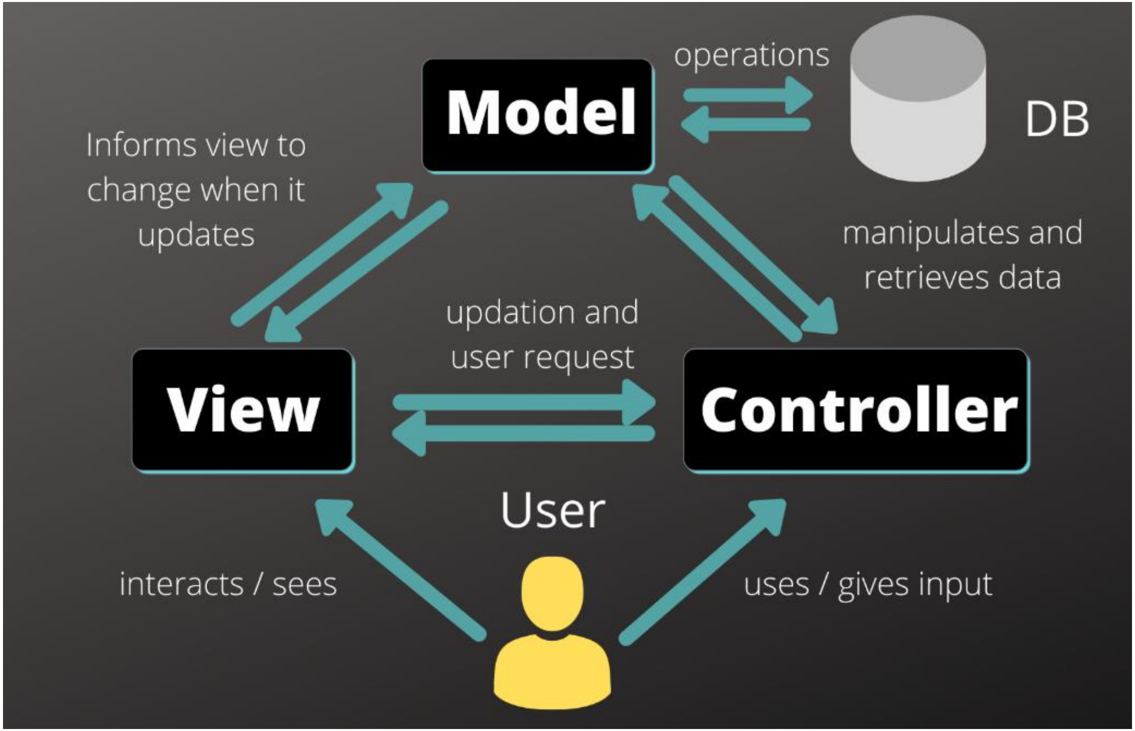
Select Patterns

Model-View-Controller

MVC (Model-View-Controller) is an **architectural pattern** that organizes applications into:

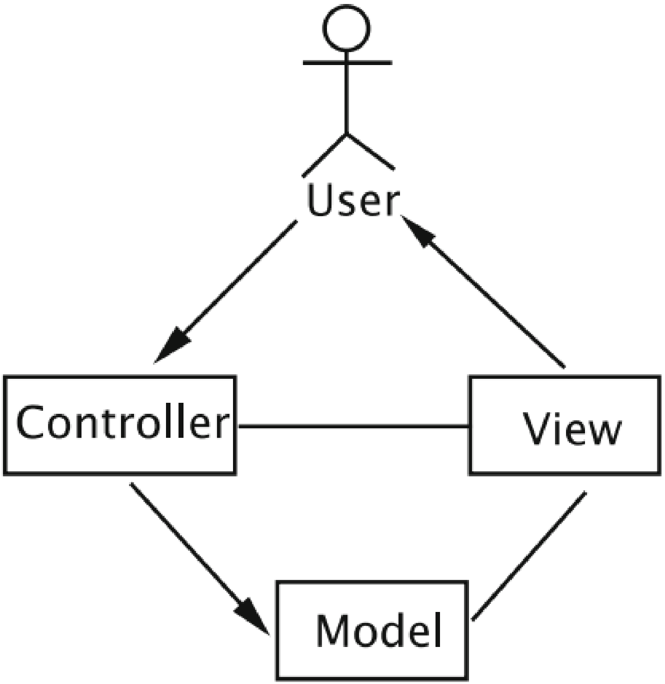
- **Model**: Represents data and business logic.
- **View:** Manages user interface and presentation.
- **Controller**: Handles user input and updates Model and View.

Model-View-Controller (MVC)


Model
operations
DB
Informs view to change when it updates
manipulates and retrieves data
updation and user request
View
Controller
User
interacts / sees
uses / gives input

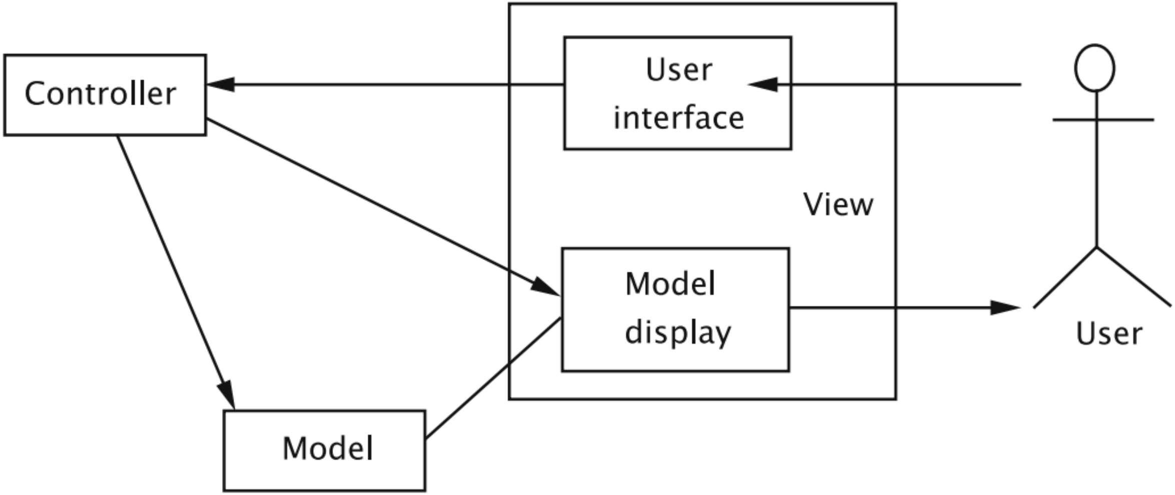
Source image: https://iq.opengenus.org

Model-View-Controller (MVC)



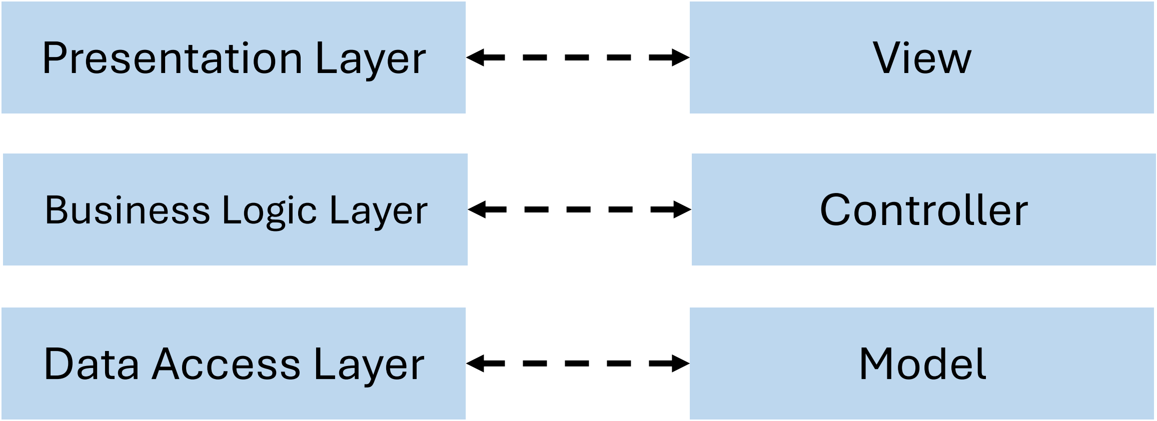
Model-View-Controller (MVC)

An alternate view of the MVC architecture.



Mapping MVC to Layers

MVC provides a dynamic interaction model within the layered architecture.

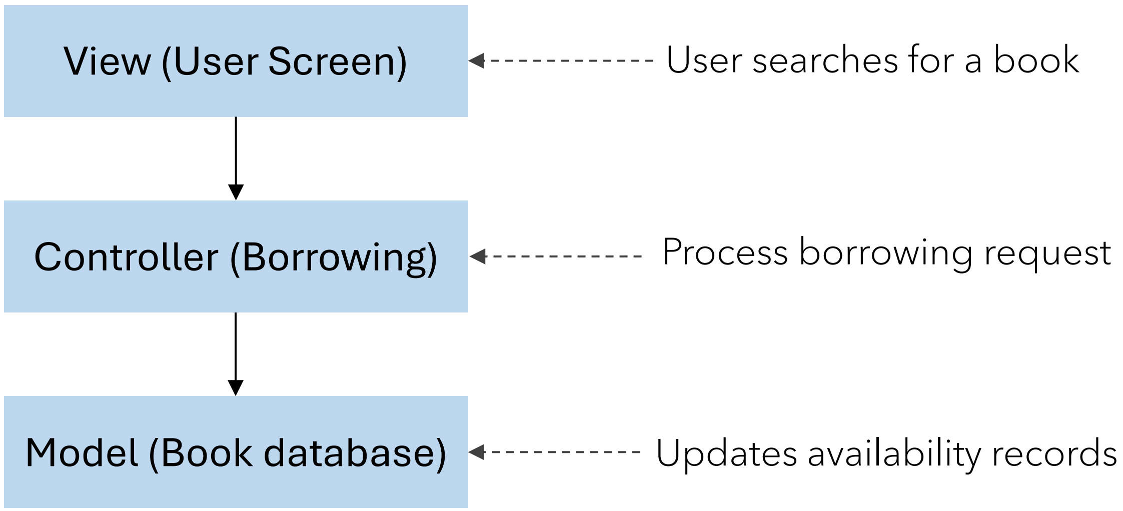


Presentation Layer ←- - - -→ View

Business Logic Layer ←- - - -→ Controller

Data Access Layer ←- - - -→ Model

MVC Example


View (User Screen)
User searches for a book
Controller (Borrowing)
Process borrowing request
Model (Book database)
Updates availability records

System Architecture to Testing

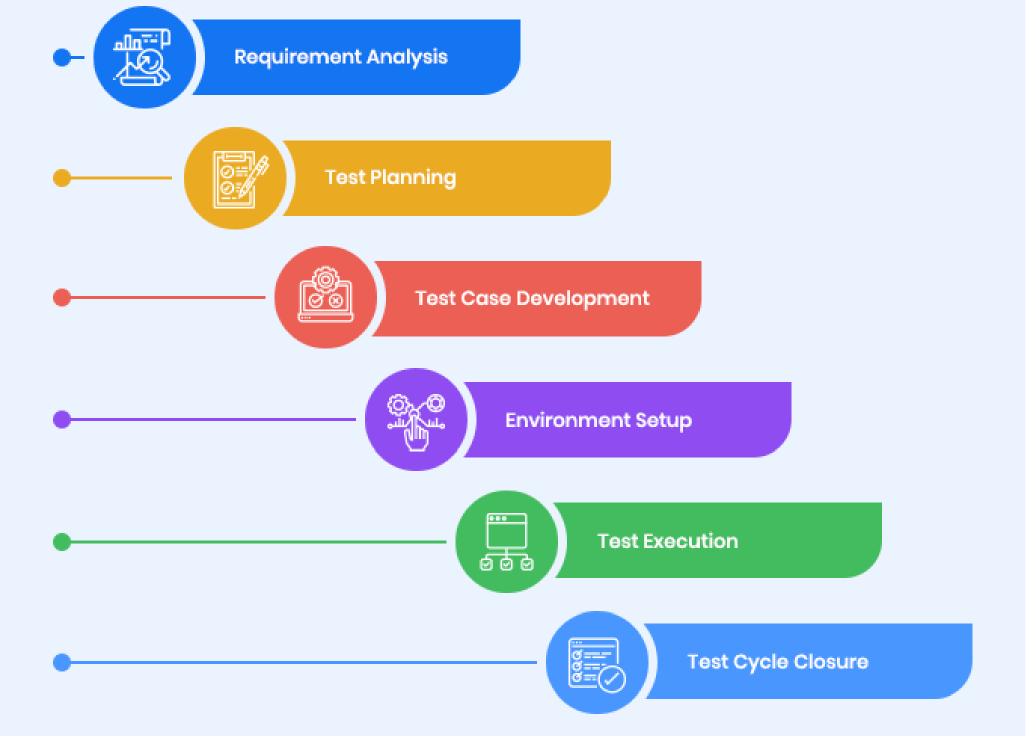
- Architecture defines the structure, influencing what and how to test.
- Non-Functional Requirements (NFRs) define quality goals.
- Architectural patterns (e.g., MVC, Microservices) shape testing strategies.

Testing and Quality Assurance (QA)

Testing and QA professionals work as part of the software development lifecycle (SDLC) to:

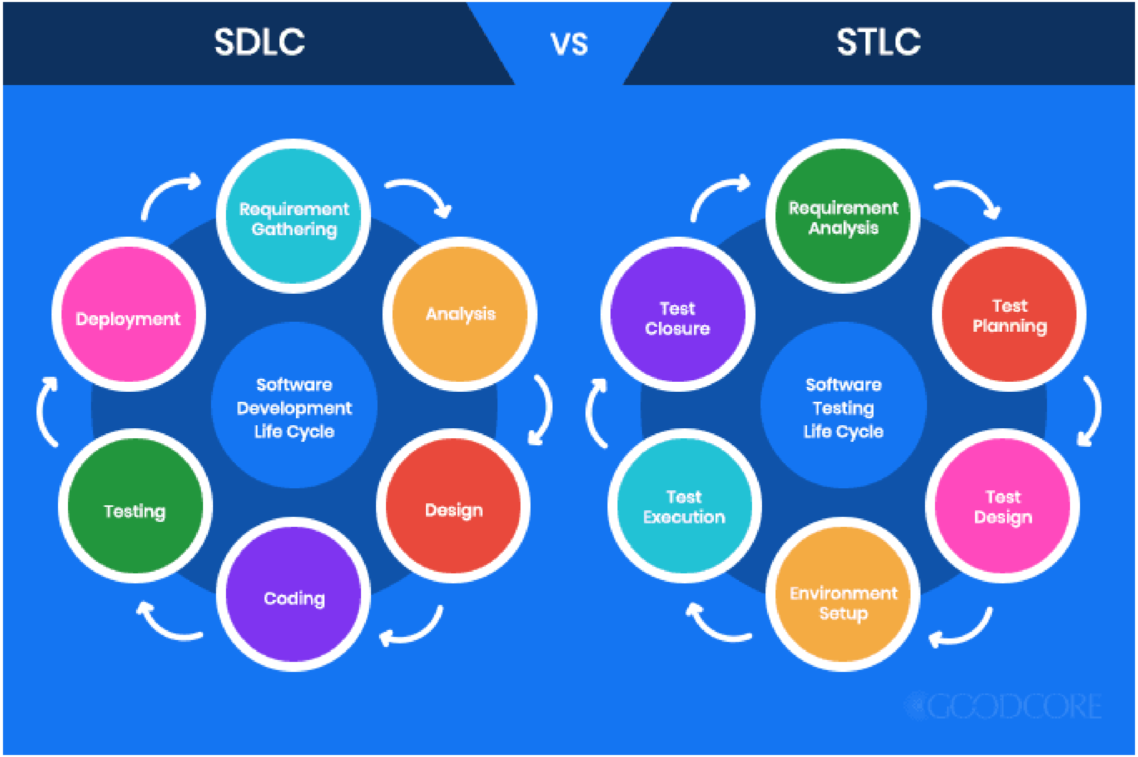
- Identify defects or issues in the software.
- Ensure the product meets both technical and user requirements.
- Maintain high-quality standards through process improvements.

Software Testing Life Cycle (STLC)



Source image: https://www.goodcore.co.uk

STLC vs SDLC



Source image: https://www.goodcore.co.uk

Testing Across Layers of Architecture

- Presentation Layer: UI, usability, and end-to-end testing.
- Business Logic Layer: Unit, integration, and API testing.
- Data Access Layer: Database integrity, performance, and security testing.

Testing Non-Functional Requirements

- Scalability: Load testing to handle increasing workloads.
- Performance: Stress testing to meet response time goals.
- Security: Penetration testing to detect vulnerabilities.
- Maintainability: Testing for ease of updates and debugging.

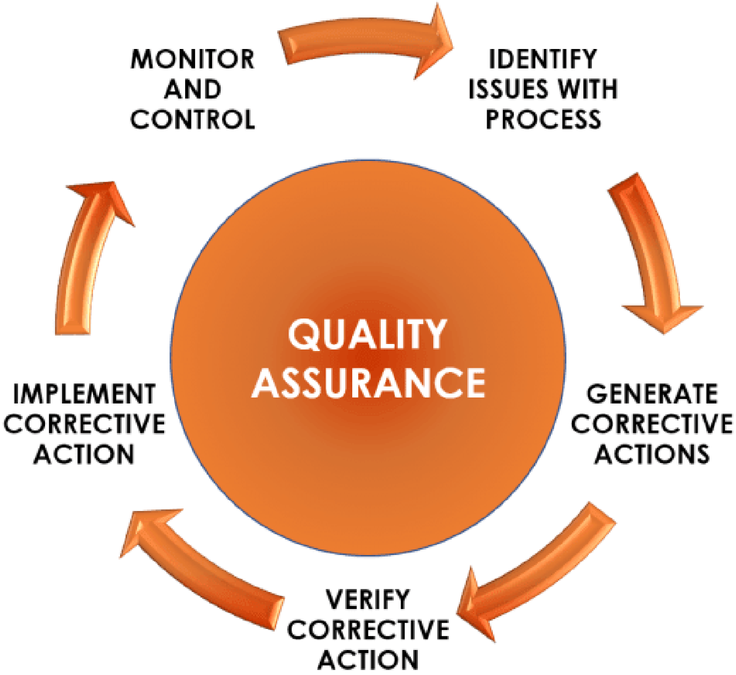
QA Strategies to Validate Architecture

- Test Plans: Align with architectural layers.
- Automation: Automate regression and functional tests.
- CI/CD Pipelines: Ensure tests validate every deployment.
- Code Reviews: Check adherence to architectural principles.

Real-World Example: E-Commerce System

- Architecture: Layered (Presentation, Business Logic, Data Access).
- Testing Plan:
  - Presentation Layer → UI Testing (Selenium).
  - Business Logic → Functional Testing (JUnit).
  - Data Access → Database Testing (SQLMap).
  - NFR Testing: Load (JMeter), Security (Burp Suite).

Feedback Loop: Architecture and QA


QUALITY ASSURANCE
MONITOR AND CONTROL
IDENTIFY ISSUES WITH PROCESS
GENERATE CORRECTIVE ACTIONS
VERIFY CORRECTIVE ACTION
IMPLEMENT CORRECTIVE ACTION

Source image: https://sofeast.com

Testing Techniques

- **Static Testing:** reviewing code, requirements, or documentation without executing the program.
- **Dynamic Testing:** executing the software and validating its behavior under various conditions.
- **Automation Testing**: spans both static and dynamic testing.