UNIT 3

- **Architectural Design:**
  - Architectural Design Decisions
  - Architectural Views
  - Architectural Patterns
  - Application Architectures

- **System modeling:**
  - Context models
  - Interaction models
  - Structural models
  - Behavioural models
  - Model-driven engineering.

Prof.Rinku Sharma
Assistant Professor
Computer Department

SYSTEM MODELING:

- CONTEXT MODELS
- INTERACTION MODELS
- STRUCTURAL MODELS
- BEHAVIORAL MODELS
- MODEL-DRIVEN ENGINEERING

SYSTEM MODELING

- System models in software engineering are **abstract, graphical representations of a system used to understand its structure, behavior, and interactions**, thereby facilitating communication with stakeholders and guiding the development process.
- Common notations like the **unified modeling language (UML) are used to create** these models providing different perspectives on the system..

UML DIAGRAM TYPES

Activity diagrams

- show the activities involved in a process or in data processing .

Use case diagrams

- show the interactions between a system and its environment.

Sequence diagrams

- show interactions between actors and the system and between system components.

Class diagrams

- show the object classes in the system and the associations between these classes.

State diagrams

- show how the system reacts to internal and external events.

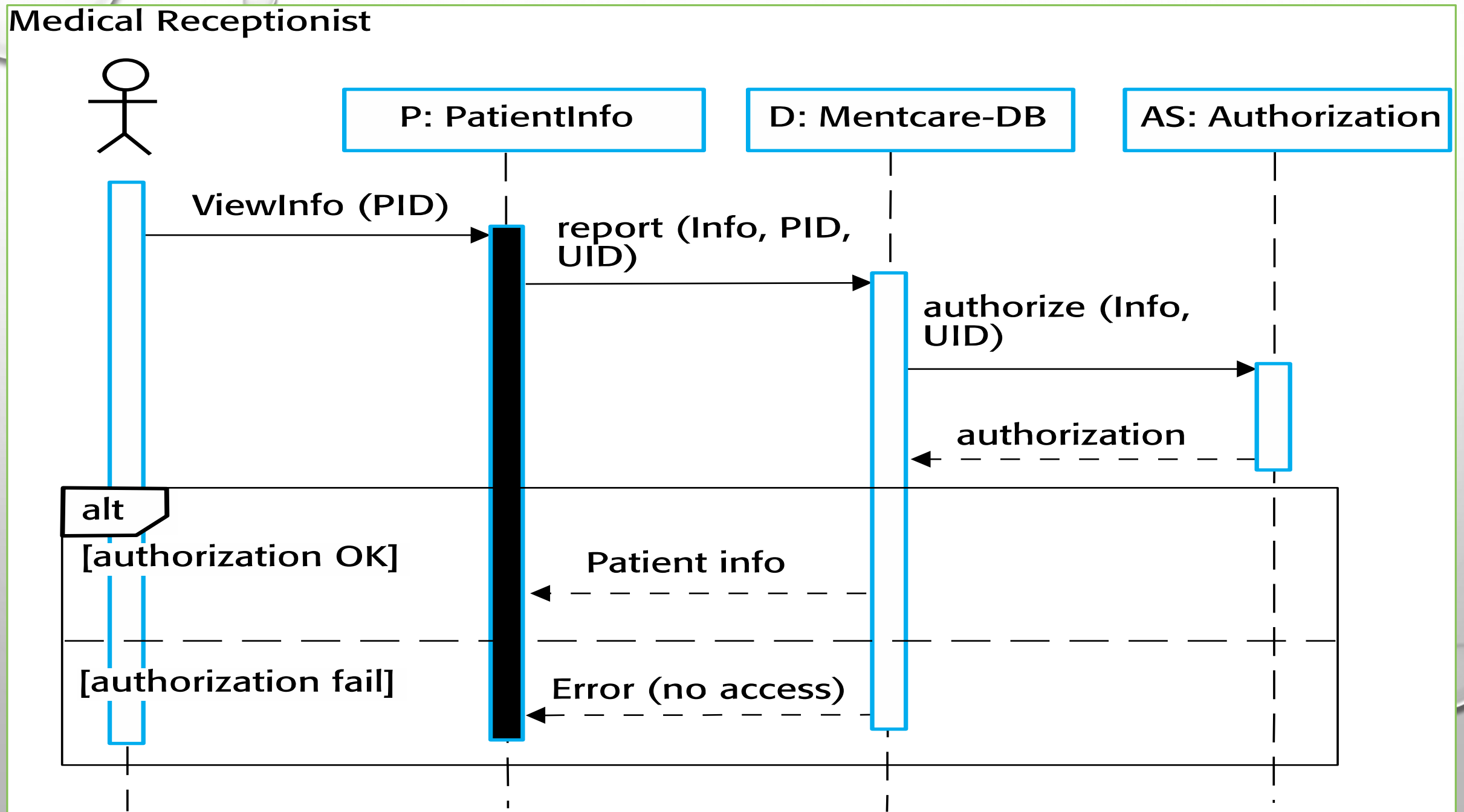
TYPES OF SYSTEM MODELS (BASED ON PERSPECTIVE)

Context Models:	• Define the boundaries of the system and its relationship with its external environment.
Interaction Models:	• Capture how users or different components interact with each other, often using use case diagrams and sequence diagrams.
Structural Models:	• Show the internal structure and architecture of the system, commonly employing class diagrams and generalization diagrams.
Behavioral Models:	• Depict how the system behaves over time, using diagrams like state diagrams and activity diagrams.

SEQUENCE DIAGRAMS

- Sequence Diagrams Are Part Of The UML And Are Used To Model The Interactions Between The Actors And The Objects Within A System.
- A Sequence Diagram **Shows The Sequence Of Interactions That Take Place During A Particular Use Case Or Use Case Instance.**
- The Objects And Actors Involved Are Listed Along The Top Of The Diagram, With A Dotted Line Drawn Vertically From These.
- Interactions Between Objects Are Indicated By Annotated Arrows.

SEQUENCE DIAGRAM FOR VIEW PATIENT INFORMATION



STRUCTURAL MODELS

STRUCTURAL MODELS

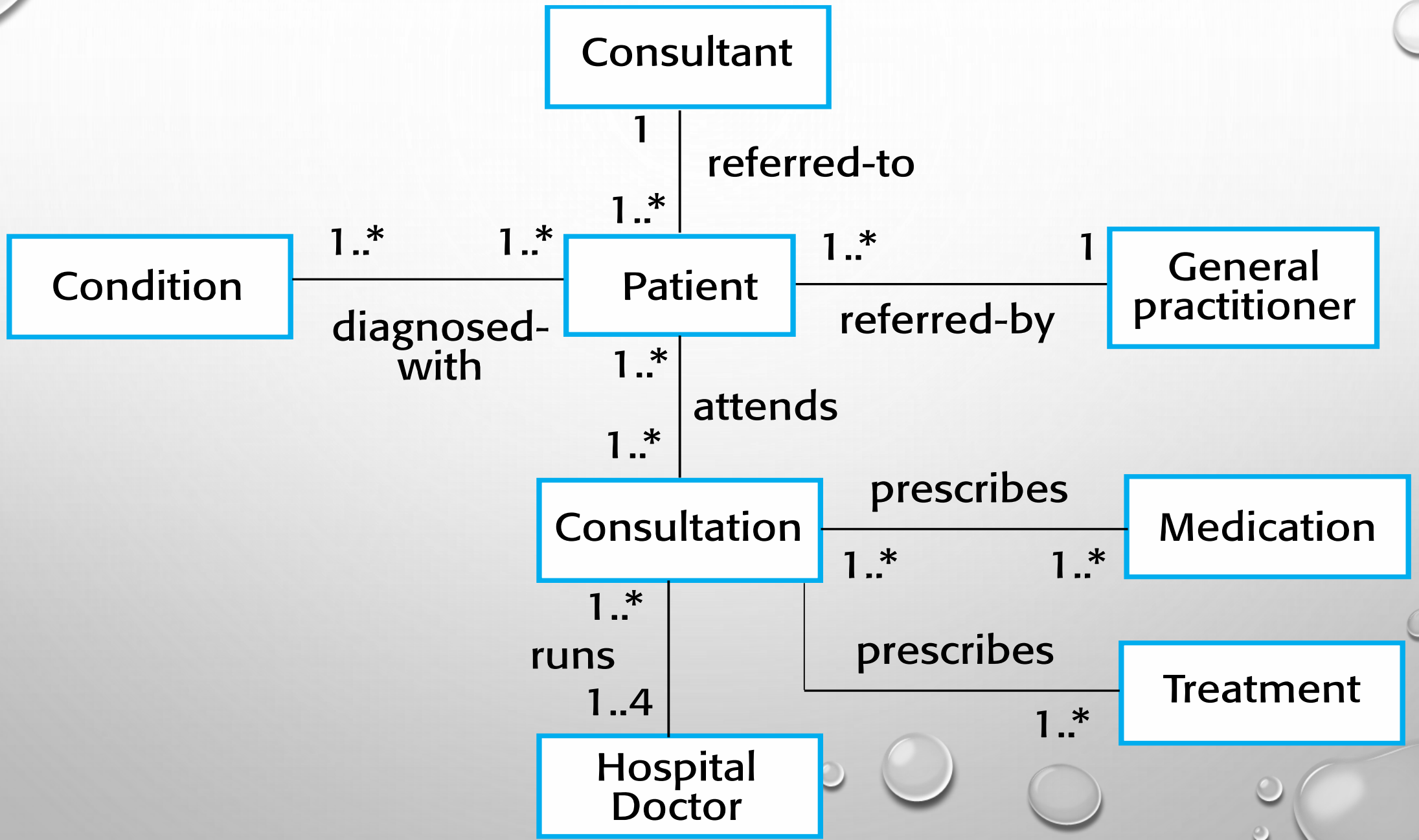
- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- Structural models may be static models, which show the **structure of the system design**, or dynamic models, which show the organization of the system when it is executing.
- you create structural models of a system when you are discussing and designing the system architecture.

CLASS DIAGRAMS

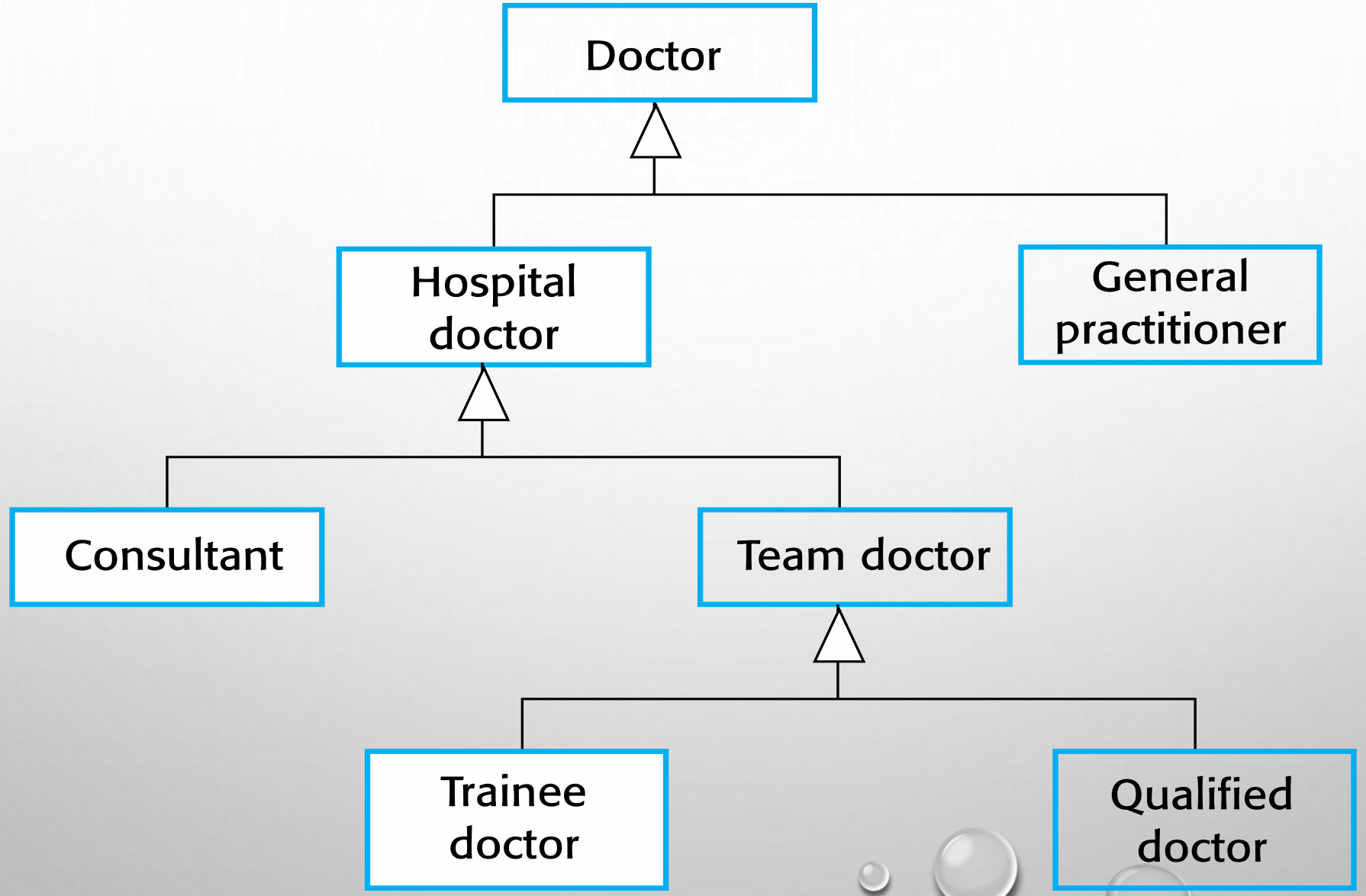
- Class Diagrams Are Used When Developing An Object-oriented System Model To Show The Classes In A System And The Associations Between These Classes.
- An Object Class Can Be Thought Of As A General Definition Of One Kind Of System Object.
- An Association Is A Link Between Classes That Indicates That There Is Some Relationship Between These Classes.
- When You Are Developing Models During The Early Stages Of The Software Engineering Process, Objects Represent Something In The Real World, Such As A Patient, A Prescription, Doctor, Etc.

UML CLASSES AND ASSOCIATION

CLASSES AND ASSOCIATIONS IN THE MHC-PMS



A GENERALIZATION HIERARCHY



MODEL-DRIVEN ENGINEERING: MDE

- **Model-driven Engineering (MDE)** Is An Approach To Software Development Where **Models Are The Principal Outputs** Of The Development Process.
- The Programs That Execute On A Hardware/Software Platform Are Then Generated Automatically From The Models.
- Proponents Of Mde Argue That This Raises The Level Of Abstraction In Software Engineering So That Engineers No Longer Have To Be Concerned With Programming Language Details Or The Specifics Of Execution Platforms.

TYPES OF MODEL

Computation independent model (**CIM**)

These model the important domain abstractions used in a system. CIMs are sometimes called domain models.

Platform independent model (**PIM**)

These model the operation of the system without reference to its implementation. The PIM is usually described using UML models that show the static system structure and how it responds to external and internal events.

Platform specific models (**PSM**)

These are transformations of the platform-independent model with a separate PSM for each application platform. In principle, there may be layers of PSM, with each layer adding some platform-specific detail.

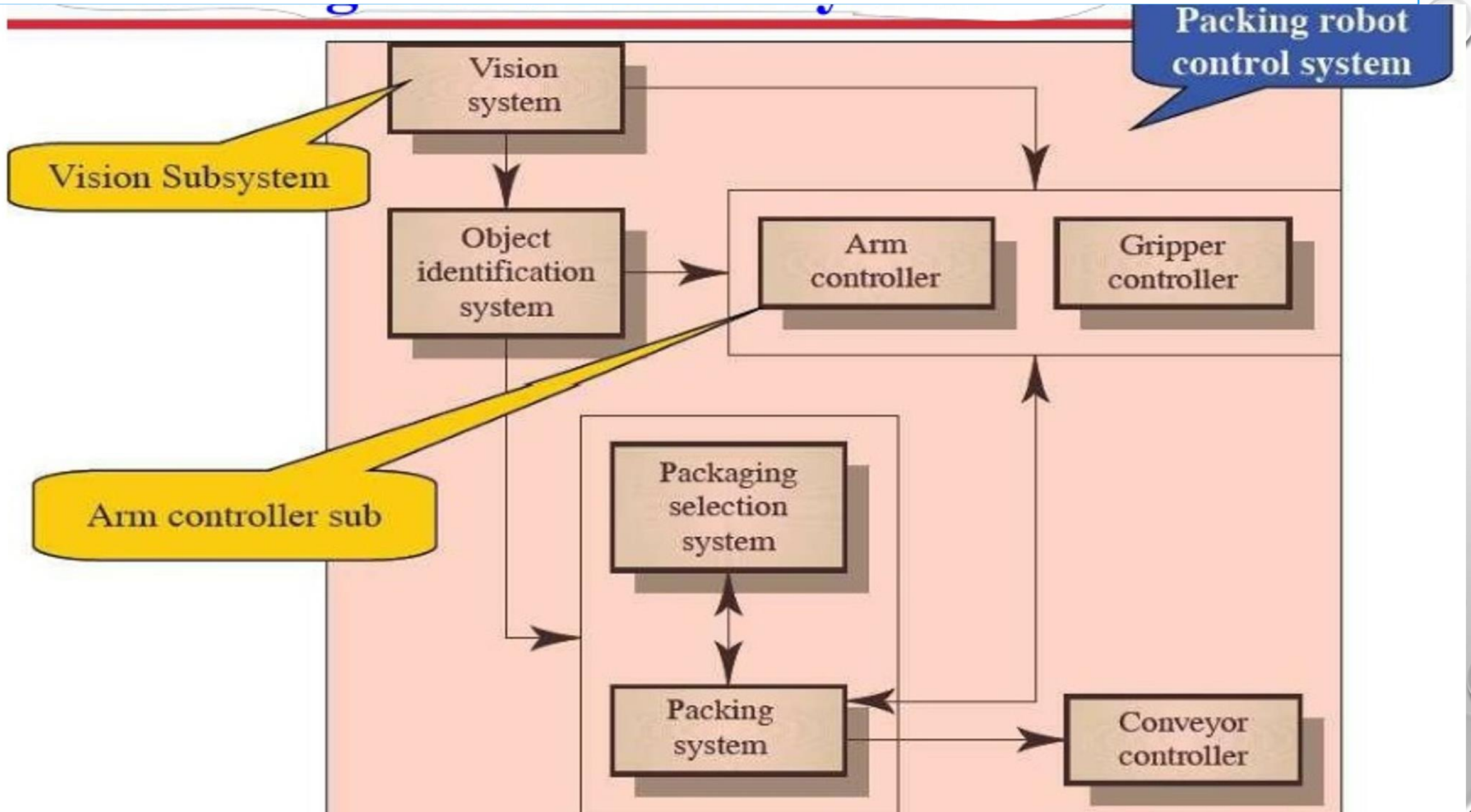
ARCHITECTURAL DESIGN

- ARCHITECTURAL DESIGN DECISIONS
- ARCHITECTURAL VIEWS
- ARCHITECTURAL PATTERNS
- APPLICATION ARCHITECTURES

ARCHITECTURAL DESIGN

- Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system.
- Architectural design is the **critical link between design and requirements** engineering, as it identifies the main structural components in a system and the relationships between them.
- **It is generally accepted that an early stage of agile processes is to design an overall systems architecture.**

THE ARCHITECTURE OF A PACKING ROBOT CONTROL SYSTEM



BOX AND LINE DIAGRAMS

- ***VERY ABSTRACT -*** THEY DO NOT SHOW THE NATURE OF COMPONENT RELATIONSHIPS NOR THE EXTERNALLY VISIBLE PROPERTIES OF THE SUB-SYSTEMS.
- HOWEVER, **USEFUL FOR COMMUNICATION WITH STAKEHOLDERS AND FOR PROJECT PLANNING.**

ARCHITECTURAL DESIGN DECISIONS

- Architectural design is a creative process so the process differs depending on the type of system being developed.
- However, a number of common decisions span all design processes and these decisions affect the non-functional characteristics of the system.


Site Selection and Analysis
Aesthetics and Style
Building Orientation
Sustainability
Decisions for Architectural Design
Design Concept
Space Planning
Materials Selection
Structural System

ARCHITECTURE AND SYSTEM CHARACTERISTICS

Performance	• **Localise critical operations and minimise communications. Use large rather than fine-grain components.**
Security	• Use a layered architecture with critical assets in the inner layers.
Safety	• Localise safety-critical features in a small number of sub-systems.
Availability	• Include redundant components and mechanisms for fault tolerance.
Maintainability	• Use fine-grain, replaceable components.

ARCHITECTURAL VIEWS

- **What views or perspectives are useful when designing and documenting a system's architecture?**
- What notations should be used for describing architectural models?
- Each architectural model only shows one view or perspective of the system.
  - **It might show how a system is decomposed into modules, how the run-time processes interact or the different ways in which system components are distributed across a network.** For both design and documentation, you usually need to present multiple views of the software architecture.

End user

Logical view

Development view

Scenarios

Process View

Physical View

Programmers & software managers

System integrators

System engineer

4 + 1 VIEW MODEL OF SOFTWARE ARCHITECTURE

A logical view, which shows the key abstractions in the system as objects or object classes.

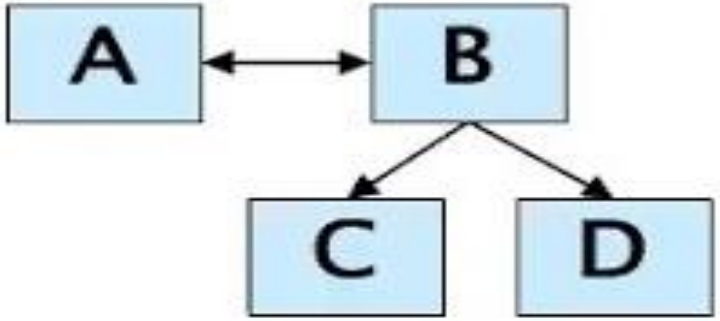
A process view, which shows how, at run-time, the system is composed of interacting processes.

A development view, which shows how the software is decomposed for development.

A physical view, which shows the system hardware and how software components are distributed across the processors in the system.

Related using use cases or scenarios (+1)

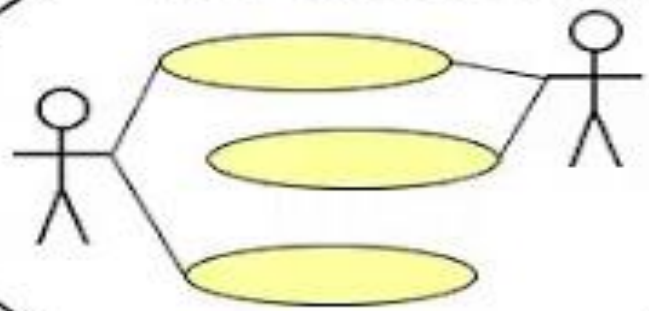
Structure view:
components, packages, interfaces



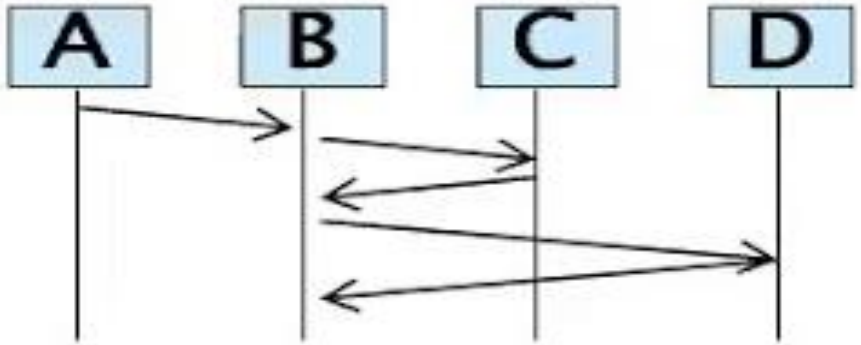
Development view

versioning policies
file ownership
…

Stakeholders & Use cases view

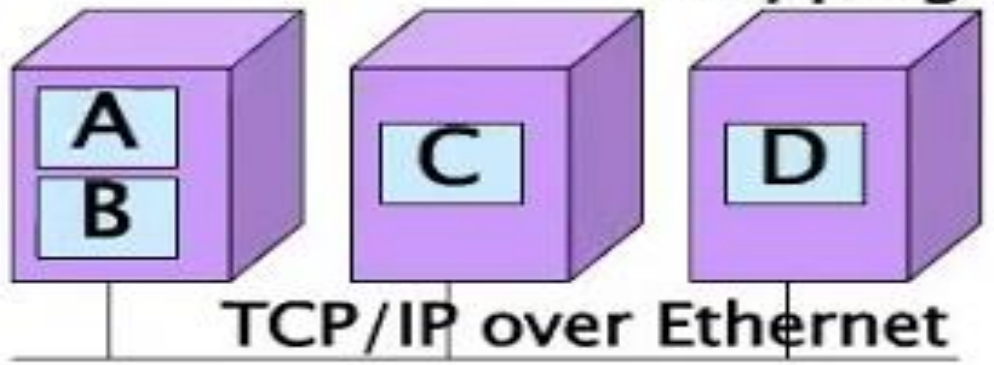


Behaviour view:
MSC, state-diagrams



BC/WC e2e-response times, freq.

Deployment view:
physical model + mapping

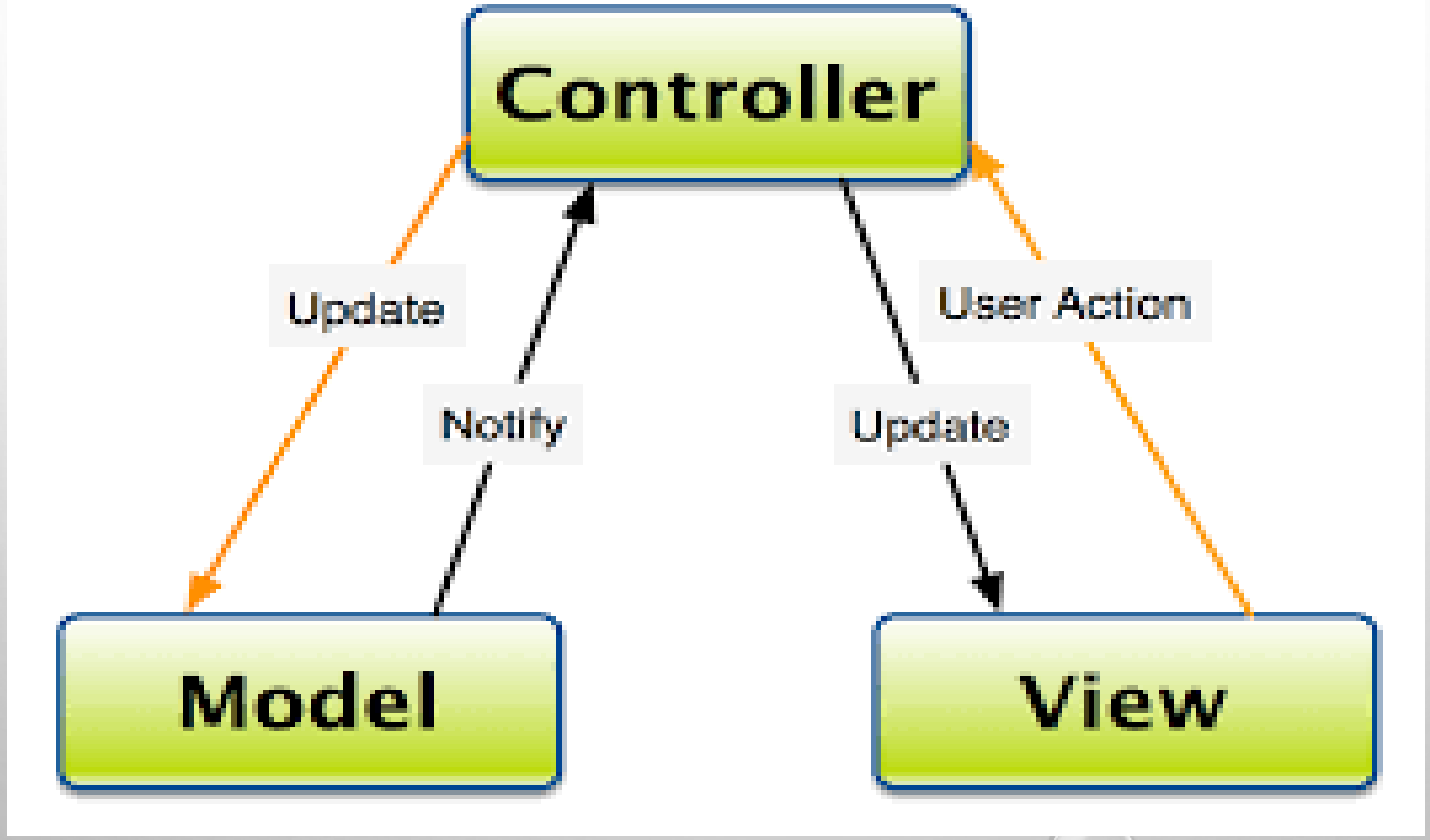


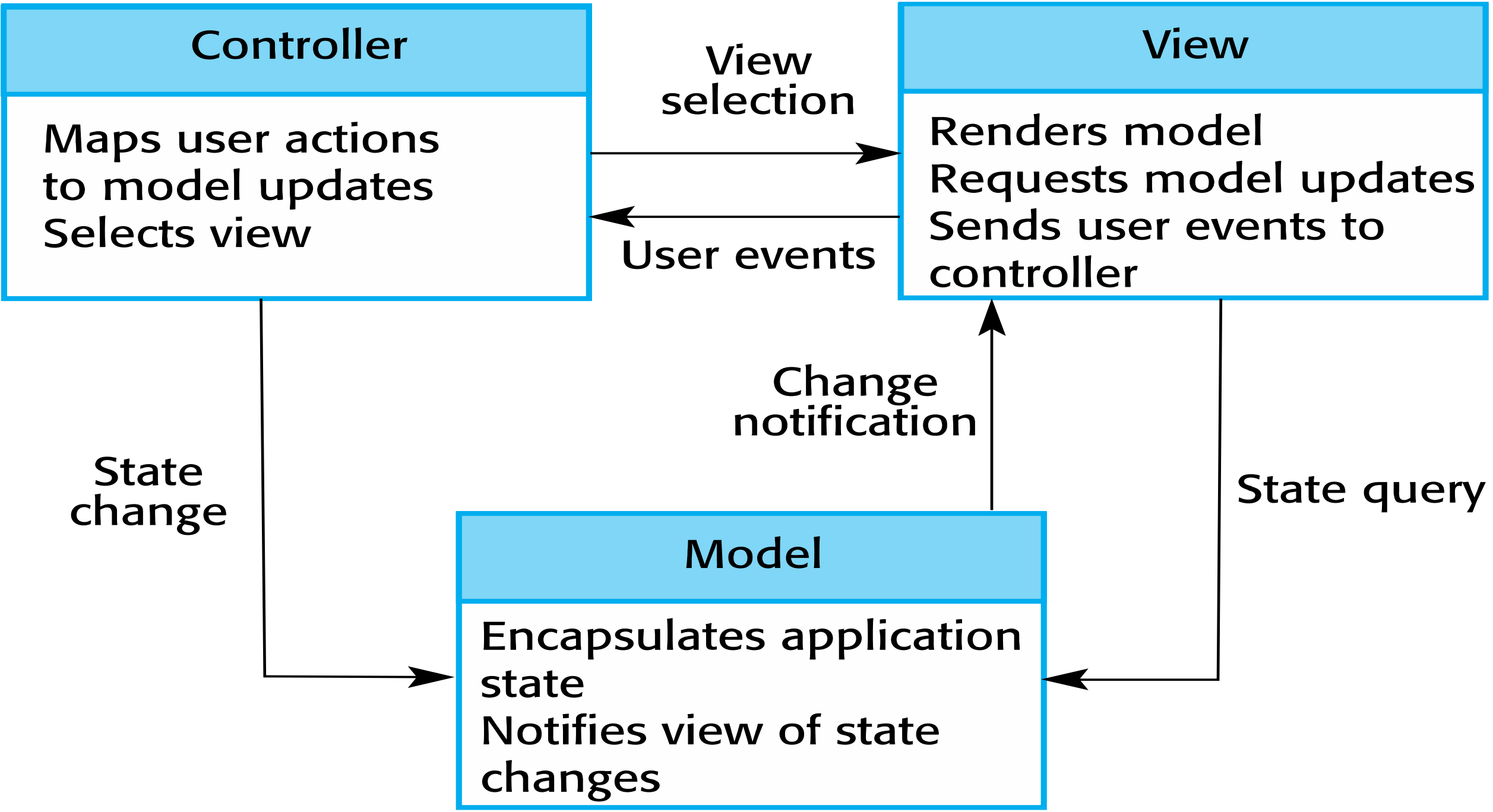
bandwidth, availability

ARCHITECTURAL PATTERNS

- Patterns are a means of representing, sharing and reusing knowledge.
- An architectural pattern is a stylized description of good design practice, which has been tried and tested in different environments.
- Patterns should include information about when they are and when the are not useful.
- Patterns may be represented using tabular and graphical descriptions.

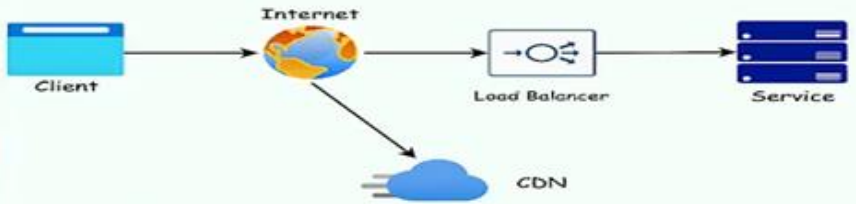
THE MODEL-VIEW-CONTROLLER (MVC) PATTERN



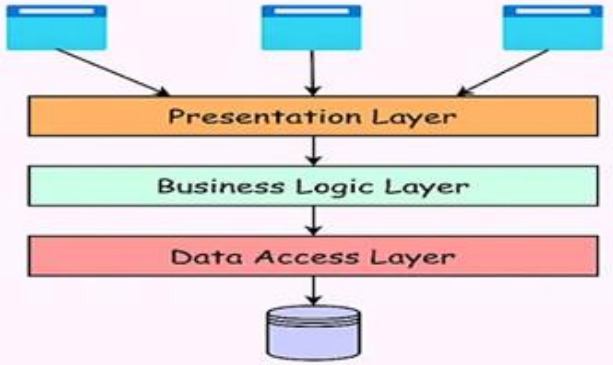

Controller
Maps user actions to model updates
Selects view
View selection
User events
View
Renders model
Requests model updates
Sends user events to controller
State change
Change notification
State query
Model
Encapsulates application state
Notifies view of state changes

Software Architecture Patterns

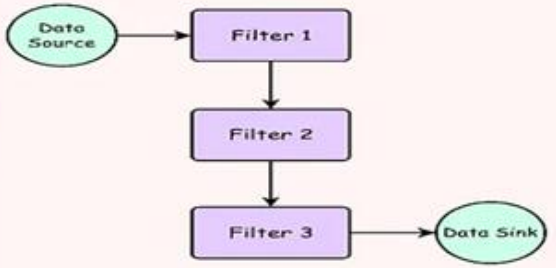
1 Client-Server Architecture



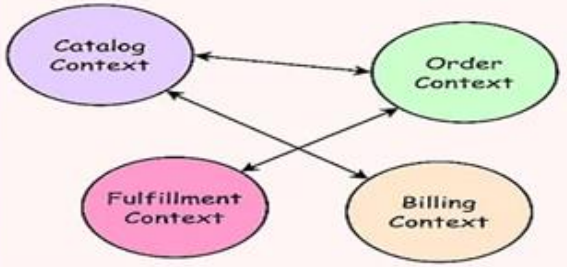
2 Layered Architecture



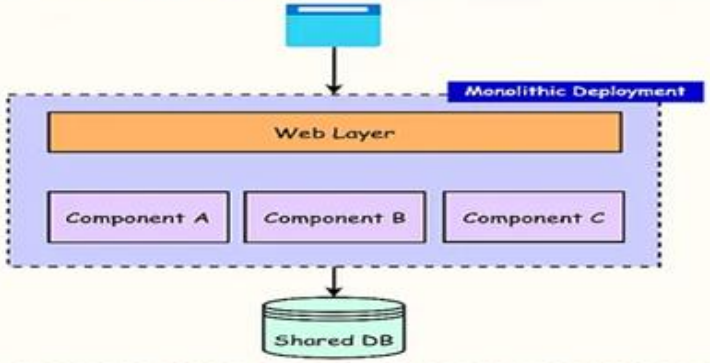
3 Pipes and Filter



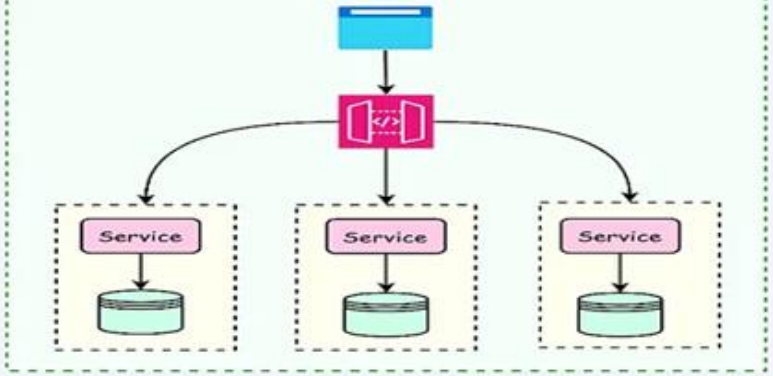
4 Domain Driven Design



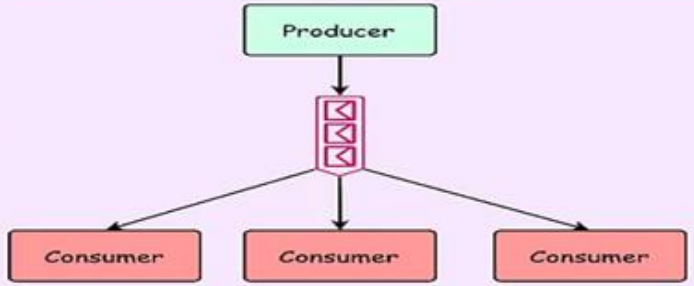
5 Monolothic Architecture



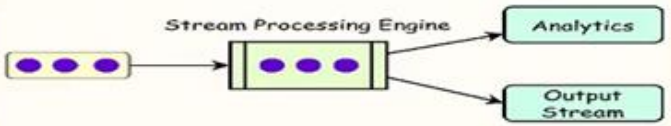
6 Microservices Architecture



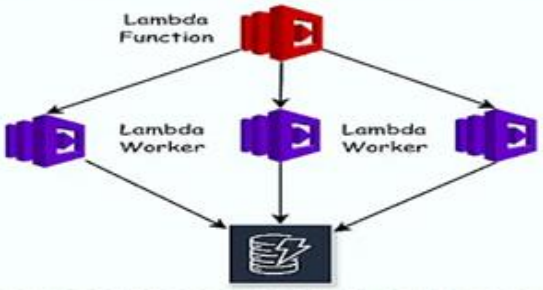
7 Event-Driven Architecture



8 Serverless Architecture



9 Serverless Architecture



APPLICATION ARCHITECTURES

- Application systems are designed to meet an organisational need.
- It is a type of software system that may be configured and adapted to create a system that meets specific requirements.

EXAMPLES OF APPLICATION TYPES

Data processing applications	• Data driven applications that process data in batches without explicit user intervention during the processing.
Transaction processing applications	• Data-centred applications that process user requests and update information in a system database.
Event processing systems	• Applications where system actions depend on interpreting events from the system's environment.
Language processing systems	• Applications where the users' intentions are specified in a formal language that is processed and interpreted by the system.

APPLICATION TYPE EXAMPLES

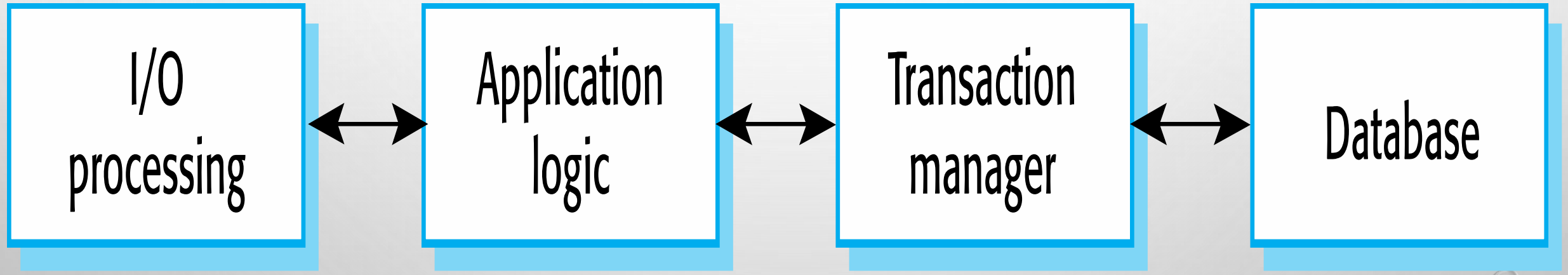
Transaction processing systems

- E-commerce systems
- Reservation systems

Language processing systems

- Compilers
- Command interpreters

THE STRUCTURE OF
TRANSACTION PROCESSING APPLICATIONS



A REPOSITORY ARCHITECTURE FOR A LANGUAGE PROCESSING SYSTEM