

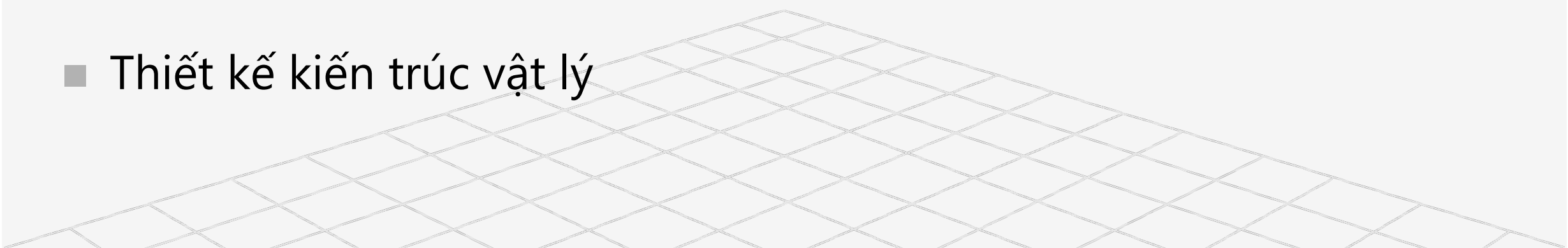
THIẾT KẾ KIẾN TRÚC PHẦN MỀM

Nguyễn Thị Thanh Huyền

Bộ môn CNPM – Khoa CNTT

ntthuyen@hnue.edu.vn

NỘI DUNG CHÍNH

- Khái niệm kiến trúc phần mềm
 - Các bước thiết kế kiến trúc
 - Các mô hình kiến trúc
 - Thiết kế kiến trúc logic
 - Thiết kế kiến trúc vật lý
- 
- A decorative perspective grid pattern, resembling a tiled floor or a wireframe mesh, is located at the bottom of the slide, extending from the left towards the right.

THIẾT KẾ KIẾN TRÚC PHẦN MỀM LÀ GÌ?

- Thiết kế kiến trúc hệ thống là việc hiểu một hệ thống được tổ chức như thế nào và thiết kế lại toàn bộ kiến trúc của hệ thống đó.
- Là giai đoạn đầu tiên của một quy trình thiết kế hệ thống. Dùng để biểu diễn mối liên kết giữa đặc tả và quy trình thiết kế, thường được tiến hành song song với các hoạt động đặc tả.
- Bước này giúp nhận biết các thành phần chính của hệ thống và cách giao tiếp của chúng với nhau

VAI TRÒ CỦA THIẾT KẾ KIẾN TRÚC

- Phân tích được tính hiệu quả của sự đáp ứng yêu cầu phần mềm
- Nghiên cứu các giải pháp thay thế ở giai đoạn sớm khi các thay đổi còn tương đối dễ dàng
 - thay đổi mô hình kiến trúc
- Giảm các rủi ro gắn với phát triển phần mềm
 - tính khả thi của mô hình

ĐỊNH HƯỚNG THIẾT KẾ KIẾN TRÚC

- Quyết định thiết kế kiến trúc
 - Có thể sử dụng kiến trúc tổng quát nào không?
 - Hệ thống được phân tán như thế nào?
 - Mẫu kiến trúc nào phù hợp?
 - Phương pháp nào được sử dụng để cấu trúc hóa hệ thống?
 - Hệ thống được phân rã thành các module như thế nào?
 - Chiến thuật điều khiển nào được sử dụng?
 - Thiết kế kiến trúc sẽ được đánh giá bằng cách nào?
 - Kiến trúc được viết thành tài liệu như thế nào?

CÁC BƯỚC THIẾT KẾ KIẾN TRÚC

1. Cấu trúc hóa hệ thống

- Phân chia hệ thống thành các hệ con (sub-system) độc lập và xác định trao đổi thông tin giữa các hệ con

2. Mô hình hóa điều khiển

- Xác lập mô hình điều khiển giữa các phần của hệ thống

3. Phân rã module

- Phân rã các hệ con thành các module

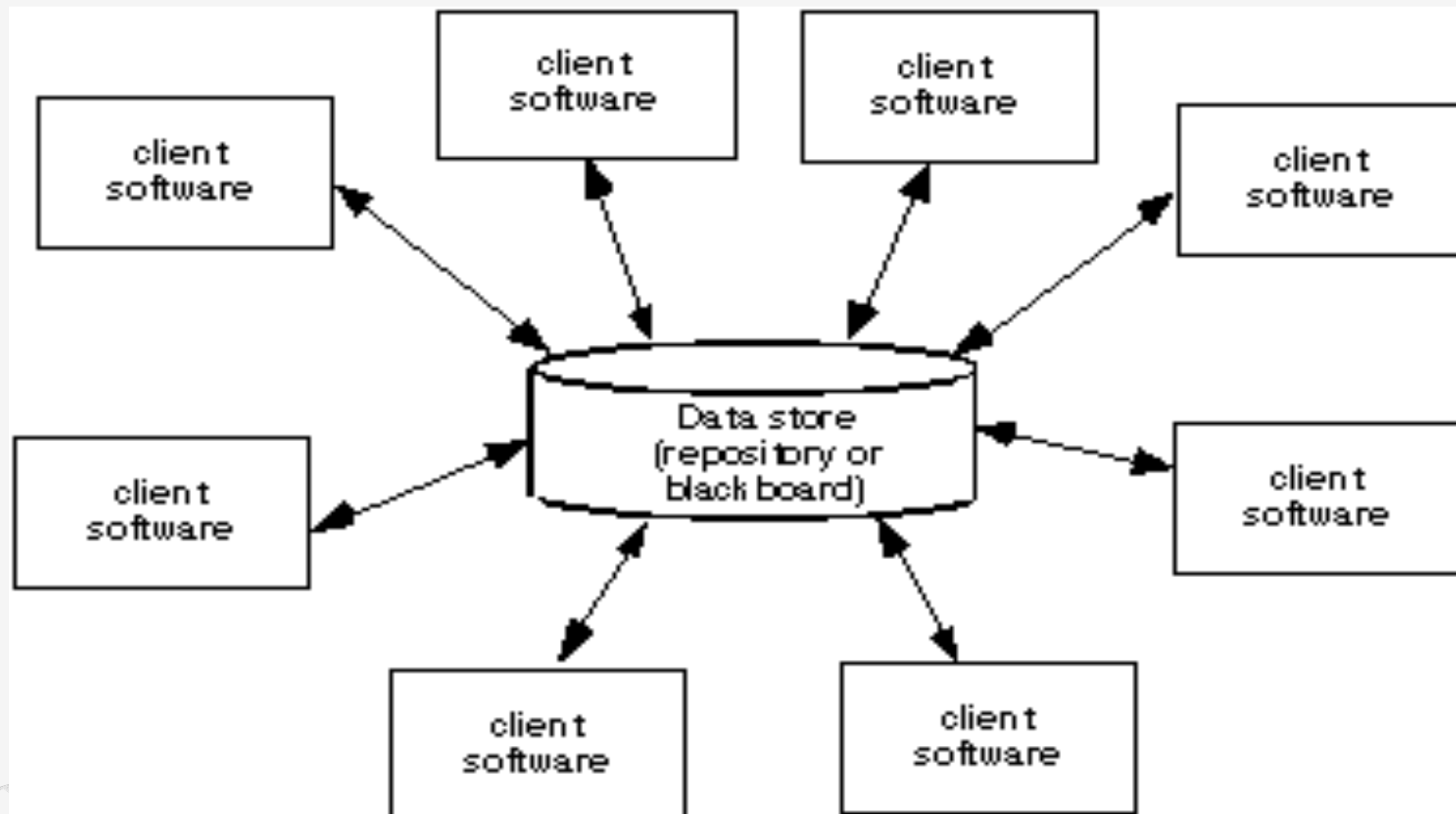
ĐỊNH HƯỚNG THIẾT KẾ KIẾN TRÚC

- Hướng tiếp cận thực hành
 - Xác định kiến trúc tổng thể dựa trên các mô hình kiến trúc
 - Xác định kiến trúc mức cao bằng cách đóng gói các thành phần UML và thiết lập sơ đồ gói (package diagrams)
 - Xác định kiến trúc ở mức thấp (vật lý) biểu diễn bằng sơ đồ thành phần (component diagram) và sơ đồ triển khai (deployment diagram)

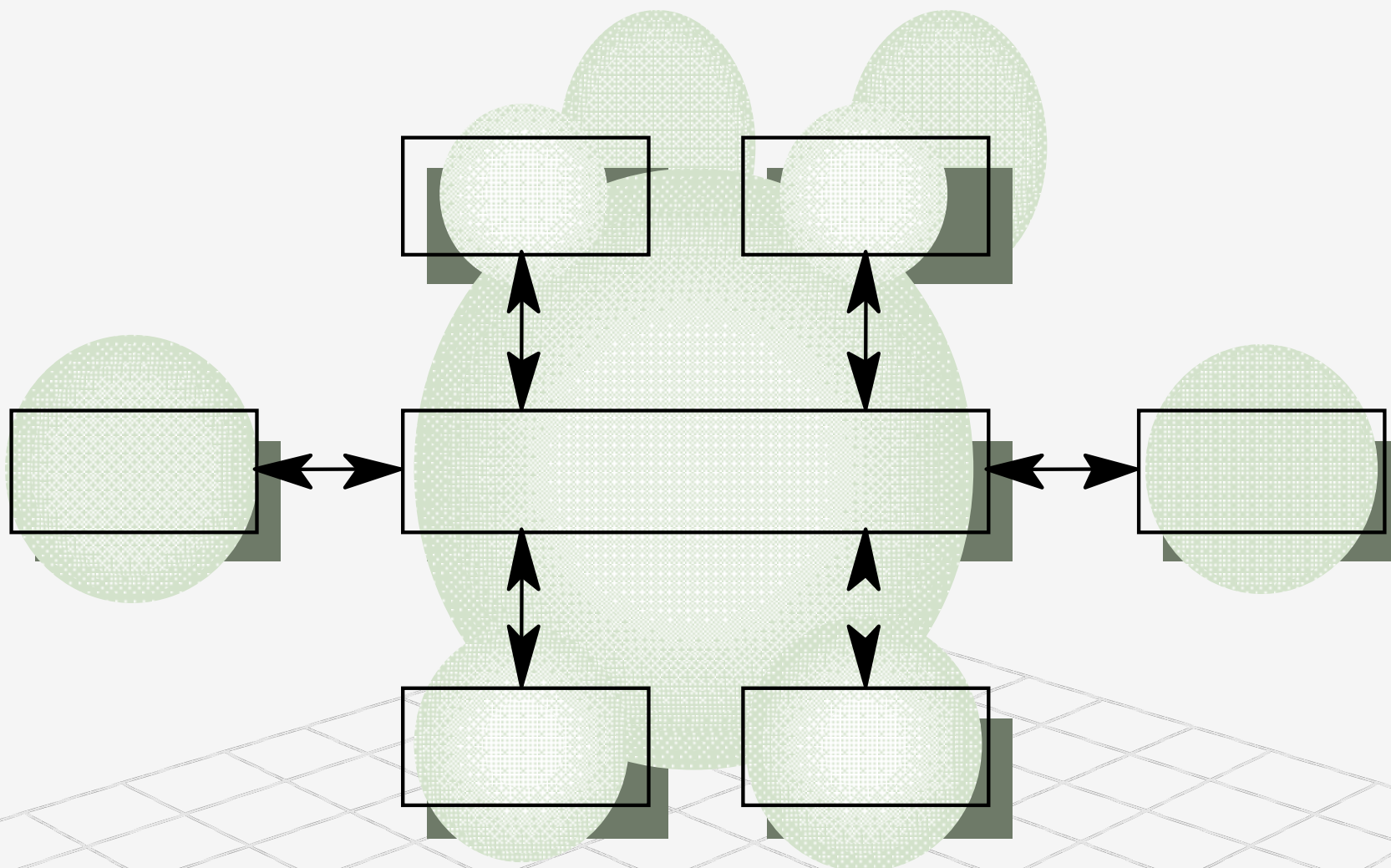
CÁC MÔ HÌNH KIẾN TRÚC

1. Data-centered architectures
2. Client-server architectures
3. Layered architectures
4. Call and return architectures
5. Data flow architectures
6. Object-oriented architectures
7. MVC

KIẾN TRÚC DỮ LIỆU TẬP TRUNG



VÍ DỤ: CASE TOOLSET



KIẾN TRÚC DỮ LIỆU TẬP TRUNG

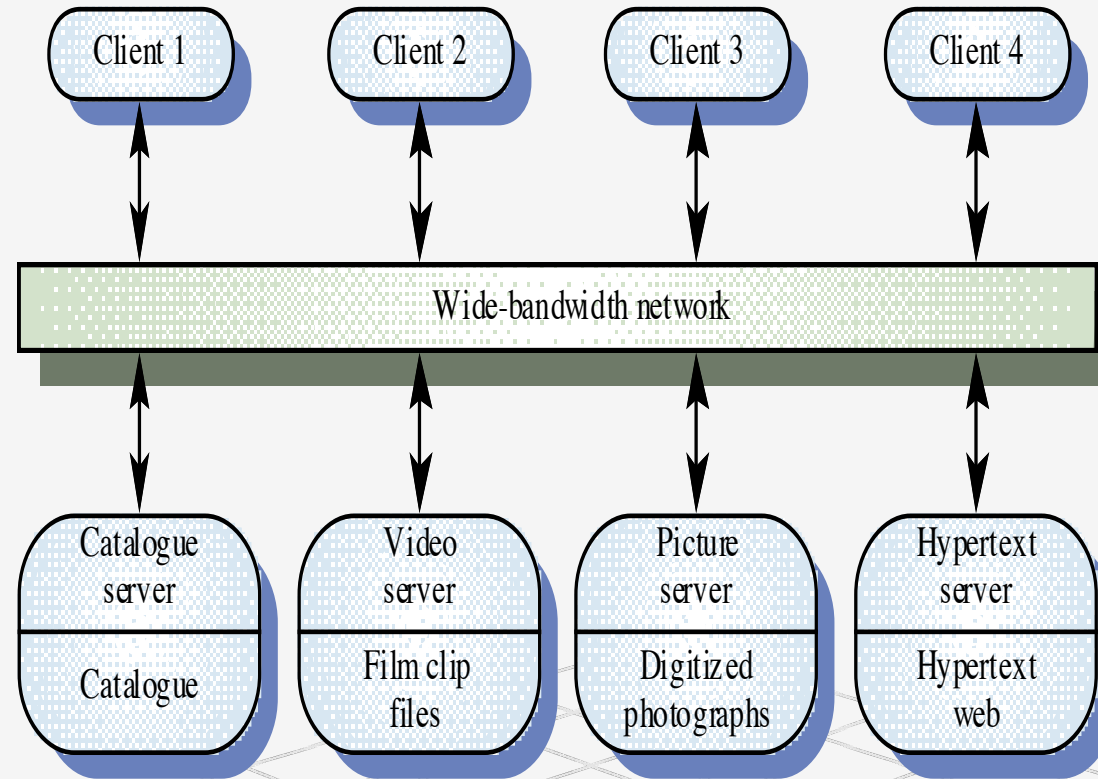
■ Ưu điểm

- Tiện lợi cho chia sẻ dữ liệu lớn
- Các phân hệ không cần biết cách thức xử lý dữ liệu của phân hệ khác

■ Nhược điểm

- Khó thay đổi cấu trúc dữ liệu
- Không cài đặt được các phương thức quản lý riêng cho các phân hệ
- Cần cài đặt cơ chế đảm bảo tính nhất quán

KIẾN TRÚC CLIENT-SERVER

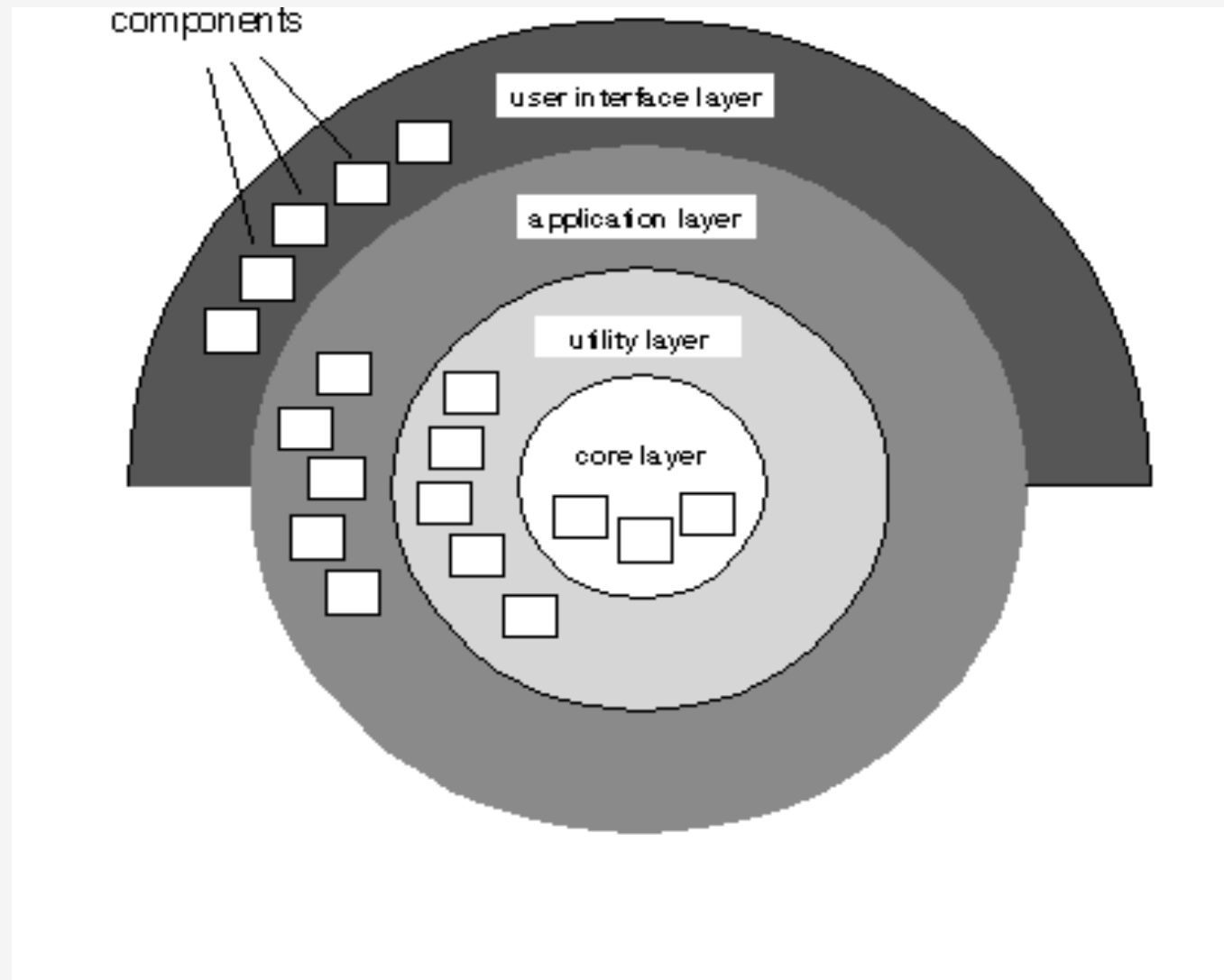


KIẾN TRÚC CLIENT-SERVER

- Ưu điểm
 - Sử dụng hiệu quả mạng, không đòi hỏi thiết bị đắt tiền
 - Dễ dàng mở rộng, thêm dịch vụ
- Nhược điểm
 - Các hệ con dùng cấu trúc dữ liệu khác nhau, khó tích hợp
 - Đòi hỏi cơ chế bảo toàn dữ liệu cho từng server

Đang là mô hình phát triển ứng dụng phổ biến

KIẾN TRÚC PHÂN TẦNG - LAYERED ARCHITECTURE

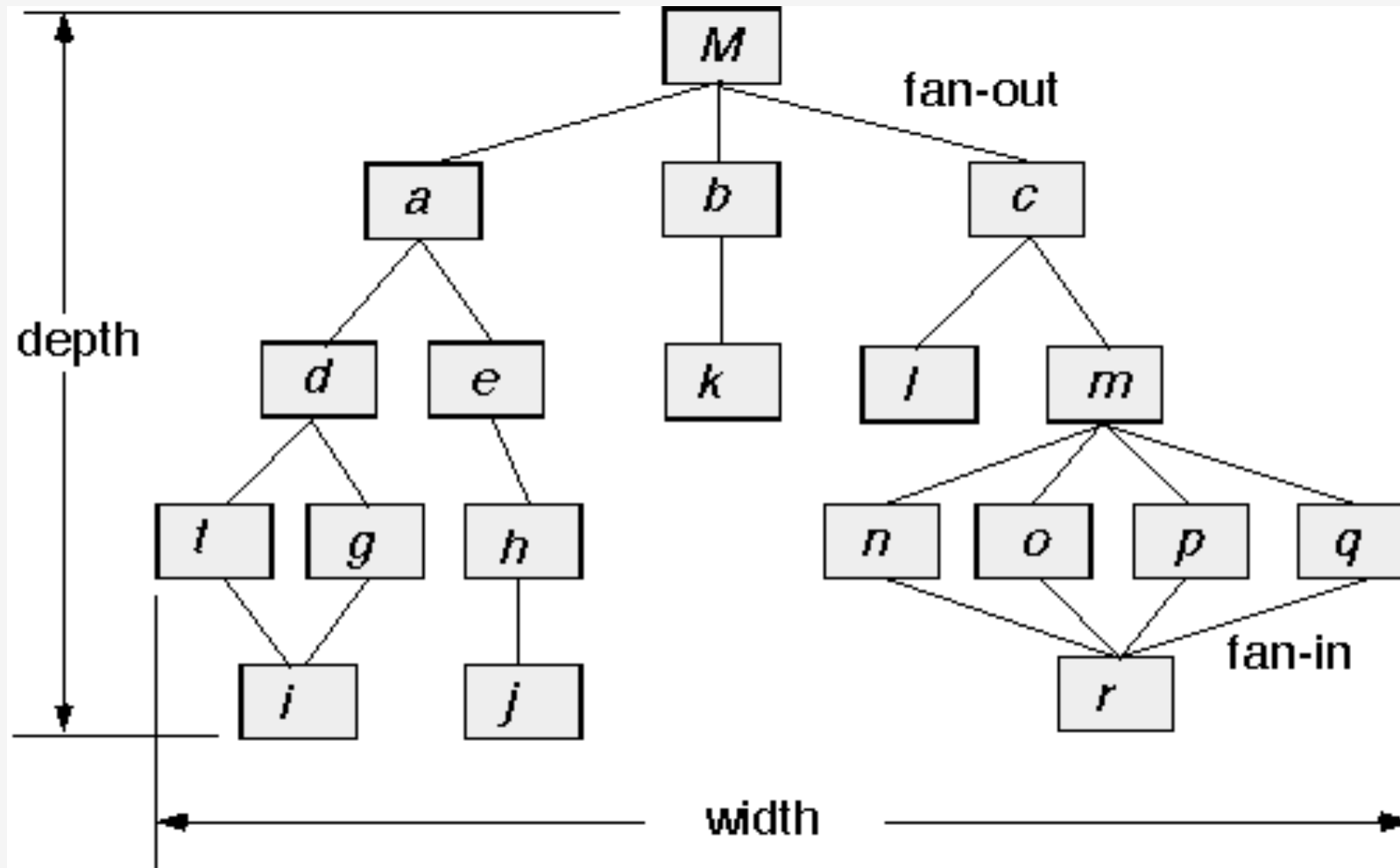


KIẾN TRÚC PHÂN TẦNG - LAYERED ARCHITECTURE

- Dùng để mô hình hóa giao diện của các phân hệ (sub-systems)
- Phân rã hệ thống thành các tầng, mỗi tầng là một tập các dịch vụ
- Hỗ trợ sự phát triển tăng trưởng của các tầng, khi giao diện mỗi tầng thay đổi chỉ ảnh hưởng tới các tầng liền kề
- Không phải hệ thống nào cũng dễ dàng phân chia theo mô hình này

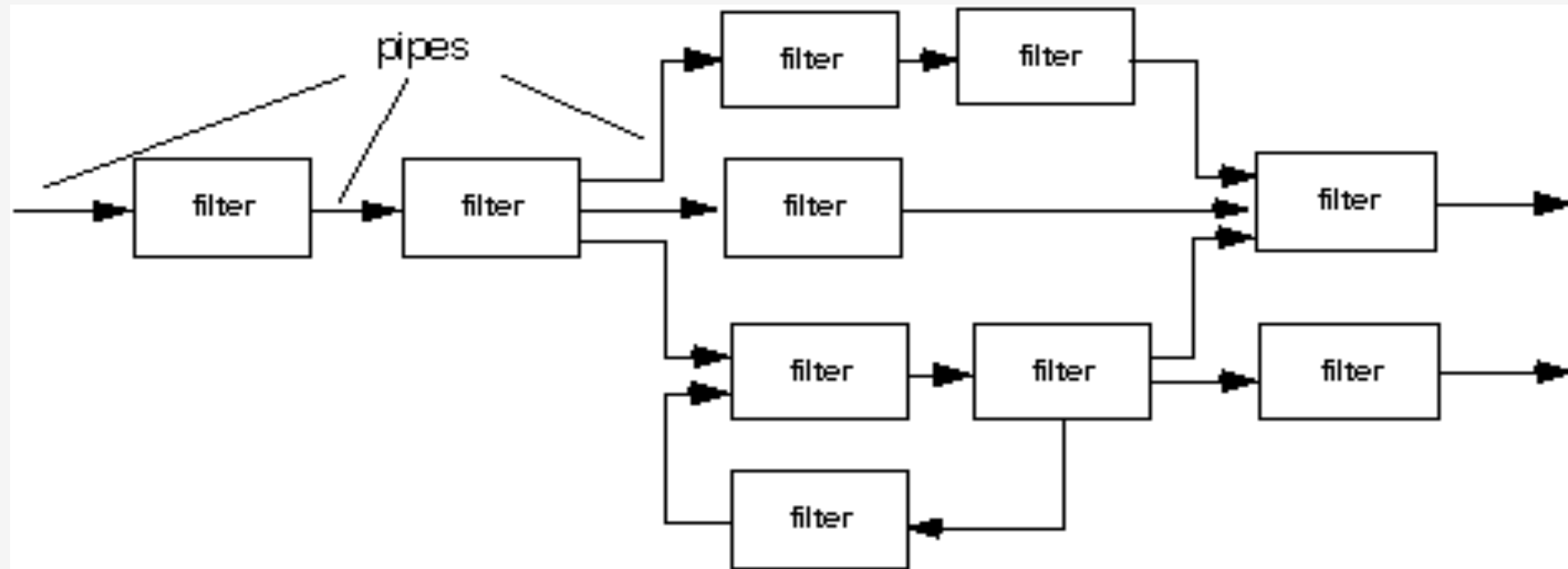
KIẾN TRÚC GỌI – TRẢ LẠI

CALL AND RETURN ARCHITECTURE



KIẾN TRÚC LUỒNG DỮ LIỆU

DATA FLOW ARCHITECTURE



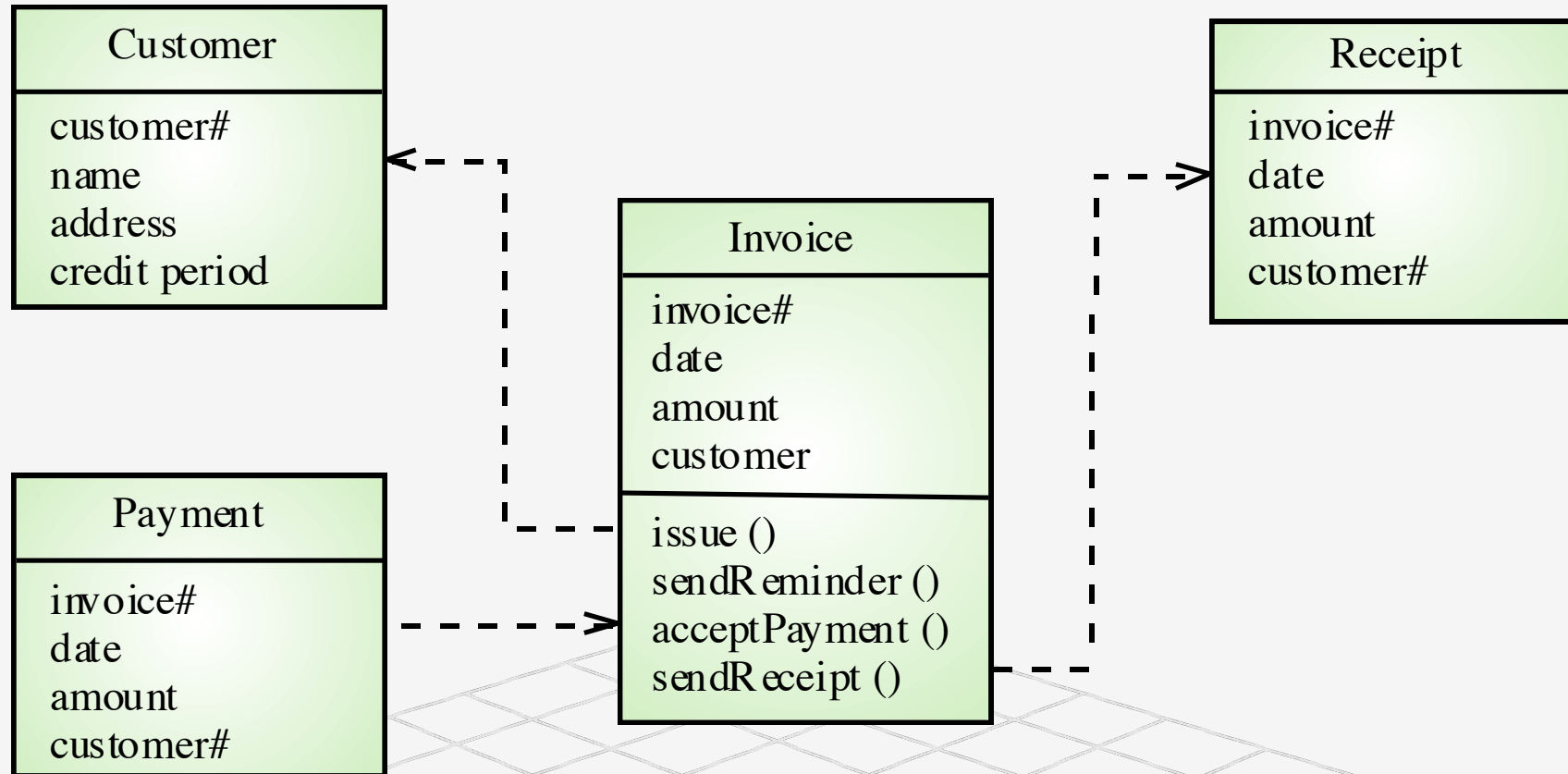
(a) pipes and filters



(b) batch sequential

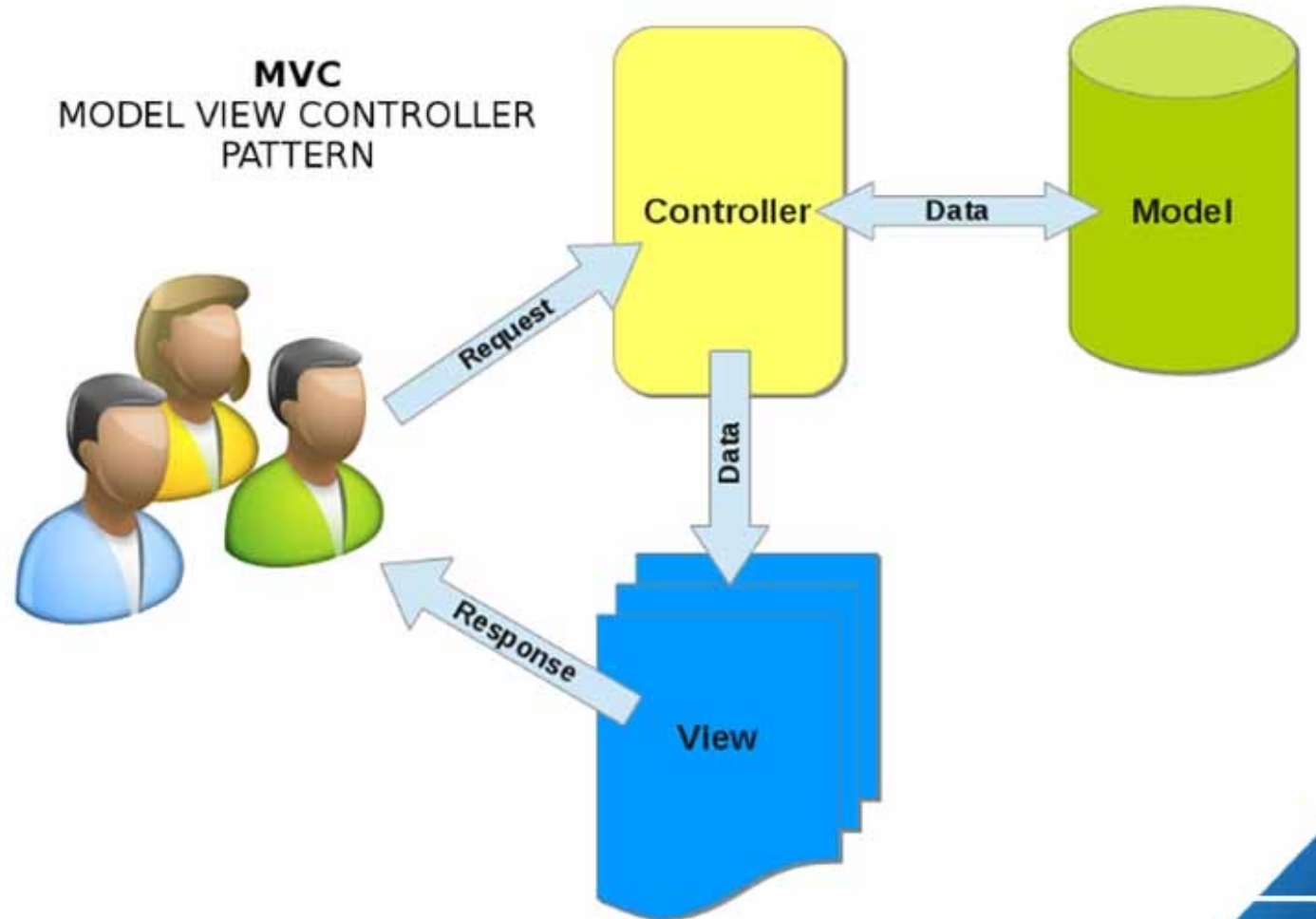
KIẾN TRÚC HƯỚNG ĐỐI TƯỢNG

OBJECT-ORIENTED ARCHITECTURE



KIẾN TRÚC MVC

- Kiến trúc MVC gồm 3 thành phần:
 - Model – Mô hình
 - View - Khung nhìn
 - Controller - Điều khiển



MÔ HÌNH MVC

■ Model:

- Mô hình hóa các đối tượng chứa dữ liệu cần xử lý
- Cung cấp các phương thức để truy cập dữ liệu
- Mô hình hóa các hoạt động nghiệp vụ

■ View:

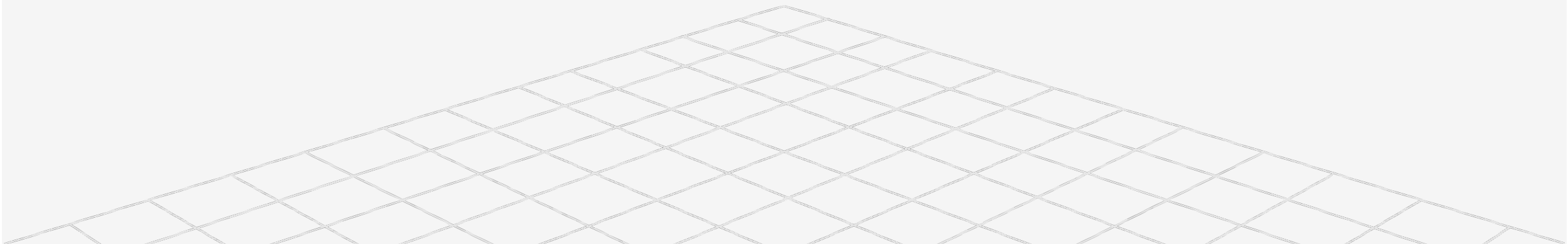
- Cung cấp giao diện cho người dùng nhập/xuất dữ liệu
- Kiểm tra tính hợp lệ của dữ liệu vào
- Bắt các sự kiện trên giao diện

■ Controller:

- Nhận các sự kiện được truyền từ View
- Gọi đến các phương thức tương ứng của Model
- Hiển thị dữ liệu trả về trên View

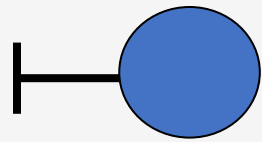
KIẾN TRÚC MVC

- Nguyên lý cơ bản của MVC là phân bổ các trách nhiệm và các hành vi ứng xử cần thiết để hỗ trợ cho một chức năng nào đó của hệ thống.
- Ba thành phần cùng hợp tác để thực hiện một dịch vụ/chức năng/ca sử dụng của hệ thống.
- MVC là kiến trúc thuận tiện cho việc quản lý, bảo trì, mở rộng và trừu tượng hóa đối tượng

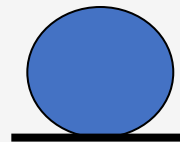


MÔ HÌNH KIẾN TRÚC MVC

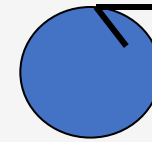
- Hệ thống gồm có 3 loại lớp đối tượng: Boundary, Entity, Control theo mô hình MVC



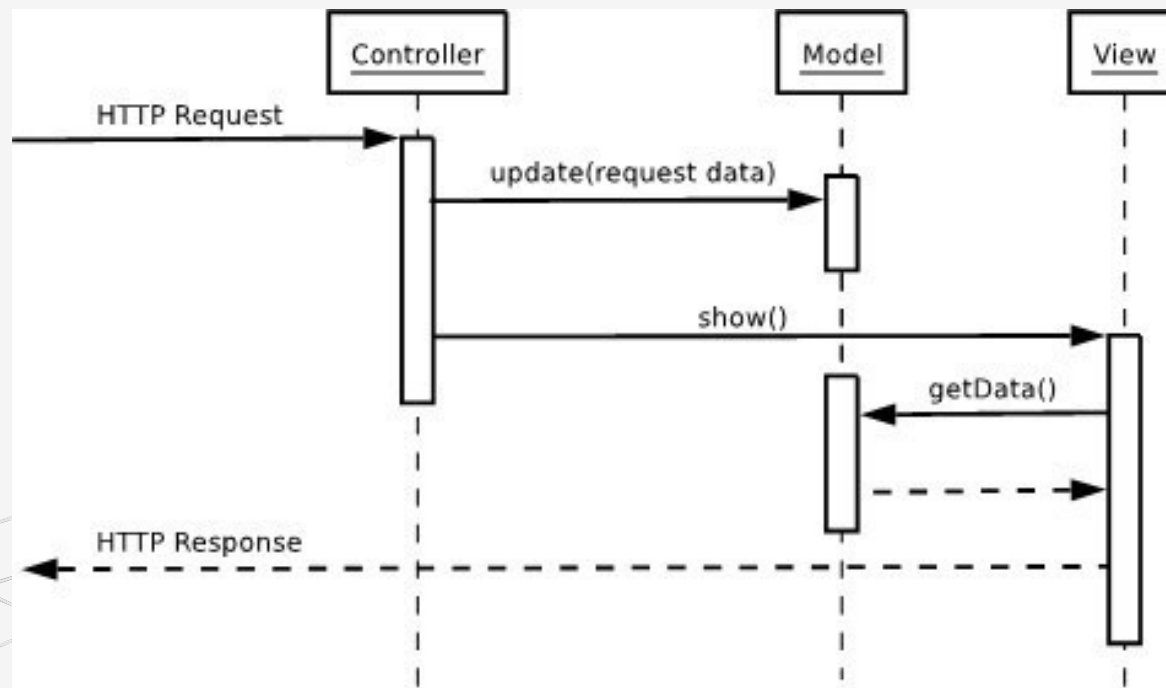
BoundaryClass



EntityClass



ControlClass



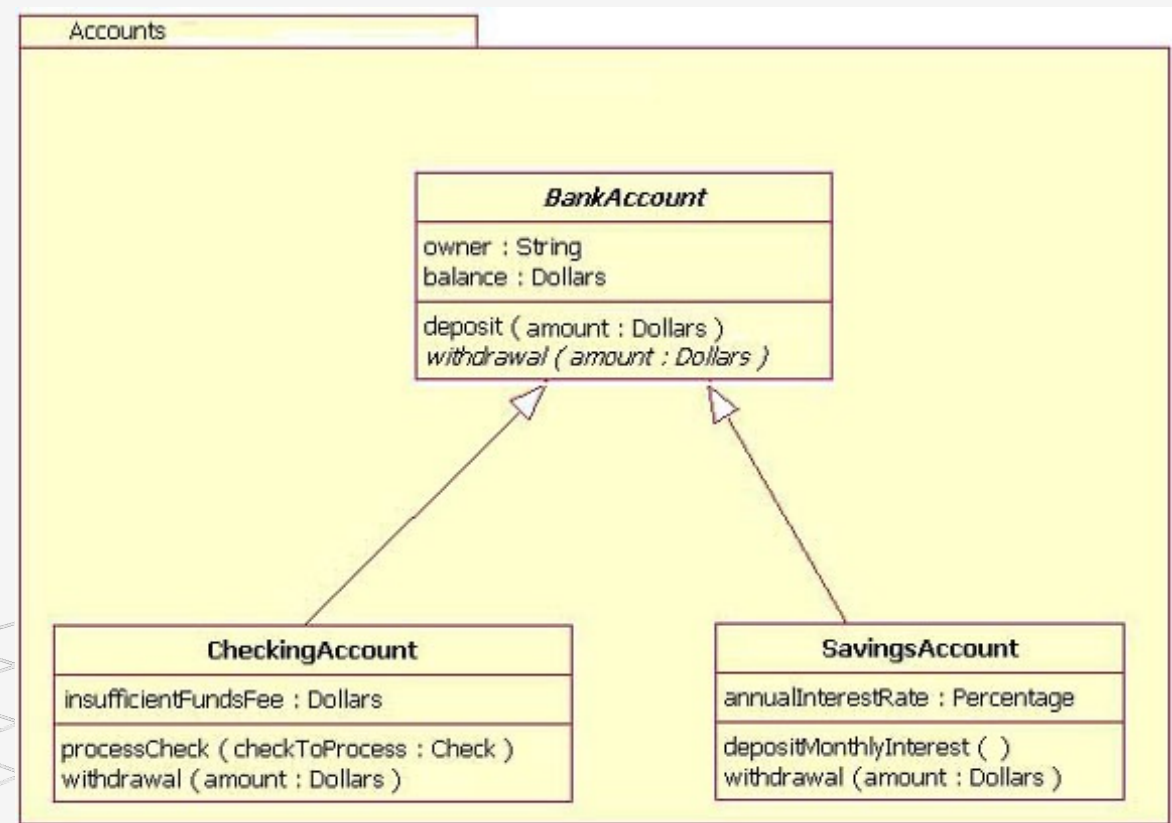
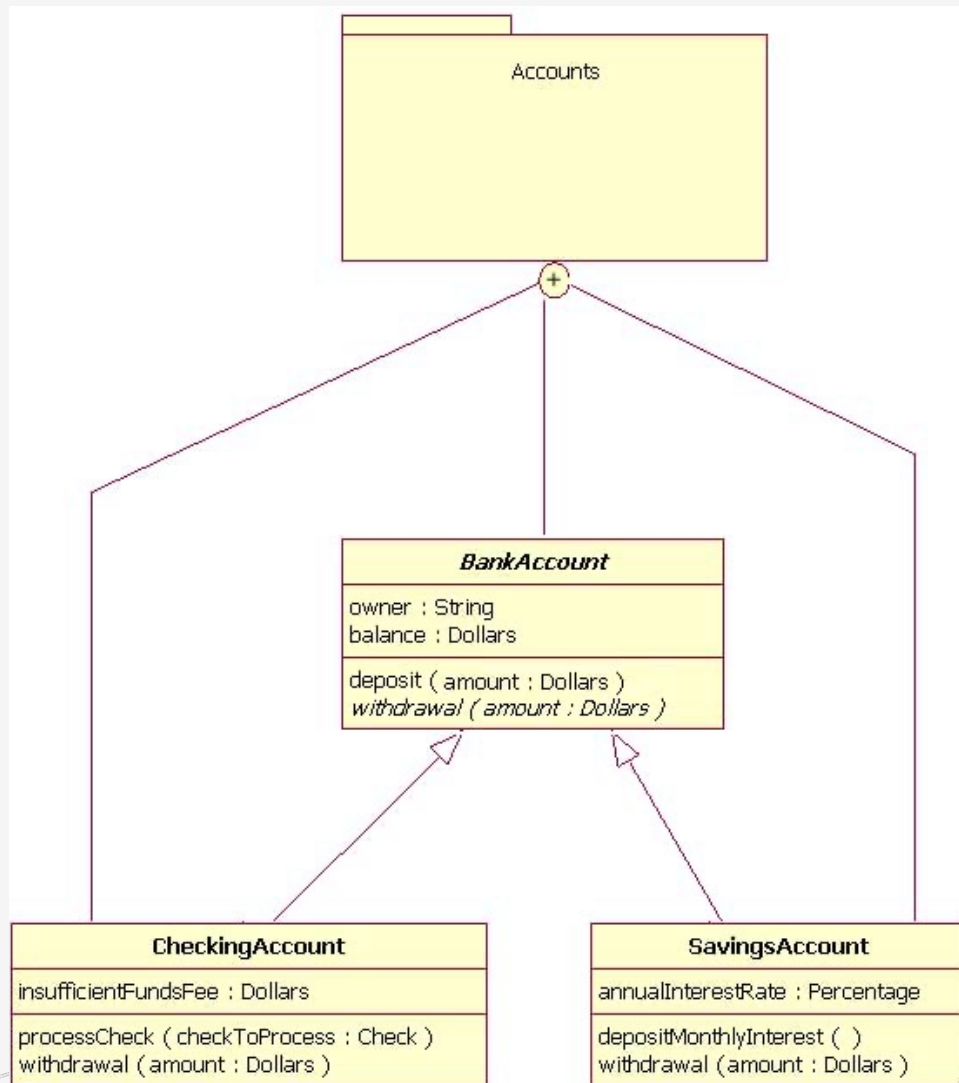
XÂY DỰNG KIẾN TRÚC PHẦN MỀM

- Xây dựng kiến trúc logic
- Xây dựng kiến trúc vật lý

THIẾT KẾ KIẾN TRÚC QUA SƠ ĐỒ GÓI (PACKAGE DIAGRAM)

- Gói và sơ đồ gói
 - Trong UML, các cấu phần và các tầng có thể biểu diễn bởi một cấu trúc ở mức cao hơn gọi là gói.
 - Một gói là một kiến trúc chung có thể áp dụng cho mọi phần tử trong mô hình UML
 - Các ca sử dụng có thể nhóm lại với nhau trong các gói để tạo thành các cấu trúc phức tạp/ có phạm vi lớn hơn
 - Các lớp có thể đóng thành gói tương đương module theo kiến trúc MVC

XÂY DỰNG SƠ ĐỒ GÓI



THIẾT KẾ KIẾN TRÚC VẬT LÝ

- Hầu hết các hệ thống phần mềm ngày nay được thực hiện từ hai máy tính trở lên.
- Thiết kế tầng kiến trúc vật lý cho hệ thống nhằm xác định:
 - Cách hệ thống được phân phối trên các máy tính
 - Các phần cứng và phần mềm nào sẽ được sử dụng
- Việc thiết kế hệ thống thường bị giới hạn bởi các hệ thống phần mềm và mạng hiện có của cơ quan tổ chức.



CÁC THÀNH PHẦN KIẾN TRÚC

- Các hệ thống phần mềm có thể được chia thành 4 nhóm chức năng:
 - Lưu trữ dữ liệu
 - Truy cập dữ liệu logic
 - Ứng dụng logic
 - Trình bày logic
- Các thành phần phần cứng:
 - Máy tính (máy khách, máy chủ)
 - Mạng

KIẾN TRÚC DỰA TRÊN SERVER

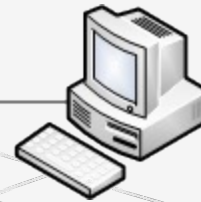
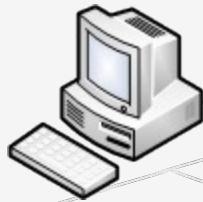
- Máy chủ thực hiện tất cả bốn nhóm chức năng ứng dụng (Data storage, Data access logic, Application logic, Presentation logic)
- Máy trạm/khách chỉ cần một màn hình, bàn phím và thiết bị liên lạc (ví dụ: modem)



KIẾN TRÚC DỰA TRÊN MÁY KHÁCH

- Các nhóm ứng dụng logic nằm trên máy trạm/khách (Data access logic, Application logic, Presentation logic)
- Một máy tính khác được sử dụng để lưu trữ dữ liệu.
- Kiến trúc này dễ phát triển nhưng khó bảo trì

Data Access Logic
Application Logic
Presentation Logic

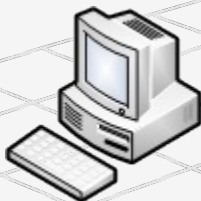


Data Storage

KIẾN TRÚC KHÁCH – CHỦ

- Cố gắng cân bằng việc xử lý giữa máy khách và máy chủ bằng cách phân chia 4 nhóm ứng dụng trên cả hai.
- Đây là kiến trúc được dùng phổ biến trong các hệ thống hiện đại.
- Số lượng xử lý trên các máy khách có thể khác nhau:
 - Máy khách nhẹ (thin client): chỉ chạy giao diện logic (presentation logic)
 - Máy khách nặng (thick client): chạy logic giao diện (presentation logic) và ứng dụng logic (application logic)

Application Logic
Presentation Logic



Data Storage
Data Access Logic

KIẾN TRÚC KHÁCH – CHỦ (CLIENT - SERVER)

- Kiến trúc Khách – Chủ có thể có hai hoặc nhiều tầng tùy theo sự phân chia của ứng dụng logic (Application logic).
 - 2 tầng: tất cả logic ứng dụng và dữ liệu nằm trên máy chủ
 - 3 tầng: logic ứng dụng nằm trên máy chủ, logic dữ liệu nằm trên một máy chủ khác.
 - 4 tầng: logic ứng dụng phân chia giữa hai máy chủ, logic dữ liệu nằm trên một máy chủ khác.
- Đôi khi được gọi là kiến trúc n-tầng

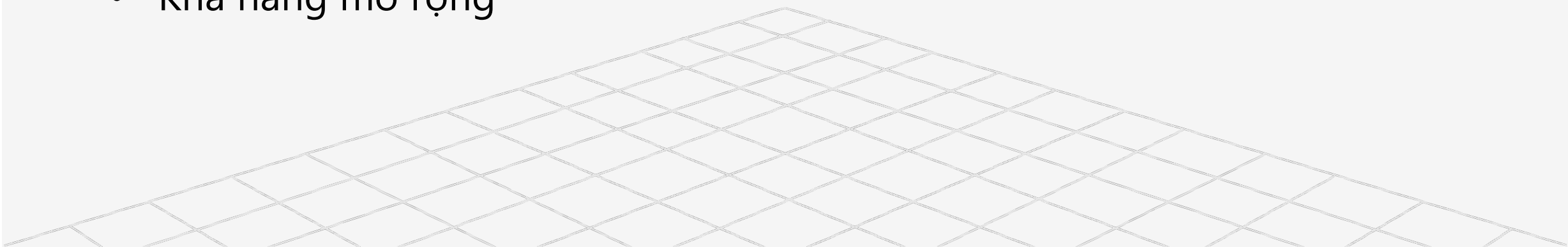
TÍNH TOÁN PHÂN TÁN – DISTRIBUTED COMPUTING

- Thể hệ tiếp theo của kiến trúc Khách - Chủ
- Máy khách không cần biết máy chủ nào để gọi
- Middleware nhận yêu cầu từ máy khách và gửi nó đến máy chủ thích hợp
- Ba cách hướng tiếp cận:
 - CORBA
 - Enterprise Java Beans
 - .NET

LỰA CHỌN KIẾN TRÚC VẬT LÝ

■ Các tiêu chí đánh giá:

- Chi phí cơ sở hạ tầng
- Chi phí phát triển
- Dễ phát triển
- Khả năng giao diện
- Kiểm soát và bảo mật
- Khả năng mở rộng

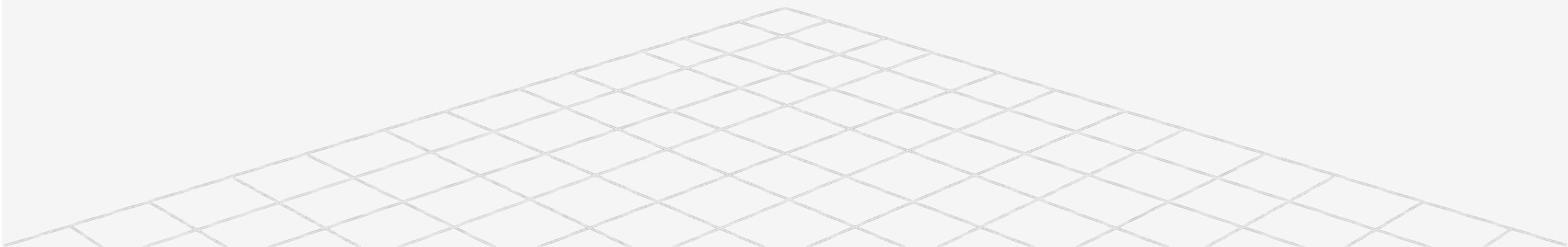


SO SÁNH CÁC KIẾN TRÚC

	Server-Based	Client-Based	Client-Server
Chi phí cơ sở hạ tầng	Rất cao	Trung bình	Thấp
Chi phí phát triển	Trung bình	Thấp	Cao
Mức độ dễ của quá trình phát triển	Thấp	Cao	Trung bình - Thấp
Khả năng giao tiếp	Thấp	Cao	Cao
Kiểm soát và bảo mật	Cao	Thấp	Trung bình
Khả năng bảo mật	Thấp	Trung bình	Cao

BIỂU ĐỒ TRIỂN KHAI - DEPLOYMENT DIAGRAM

- Biểu diễn mối quan hệ giữa các thành phần phần cứng của hệ thống.
- Các thành phần của biểu đồ:
 - Các nút (Nodes): Một nguồn tính toán
 - Các hiện vật (Artifacts): một phần của HTTT sẽ được cài lên một nút
 - Đường kết nối (Communication paths): kết nối mạng, kết nối cáp USB



VÍ DỤ BIỂU ĐỒ TRIỂN KHAI

