# Introduction ArchiMate

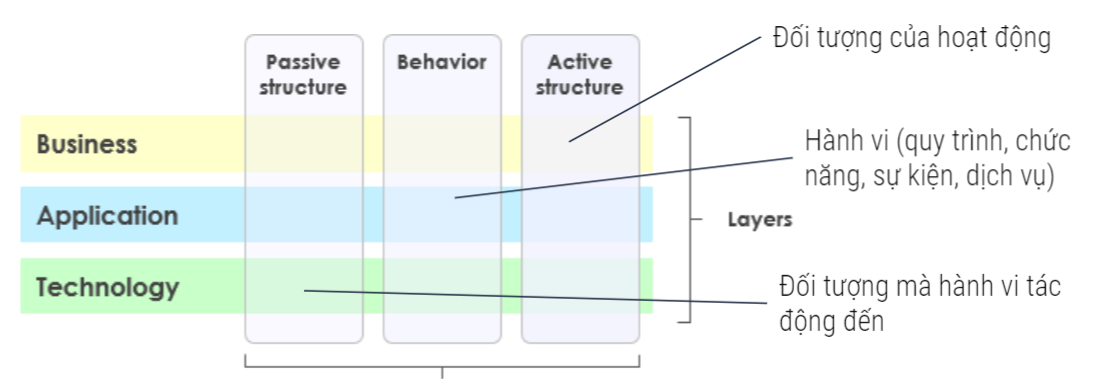
## Câu 1 : Archimate là gì ?

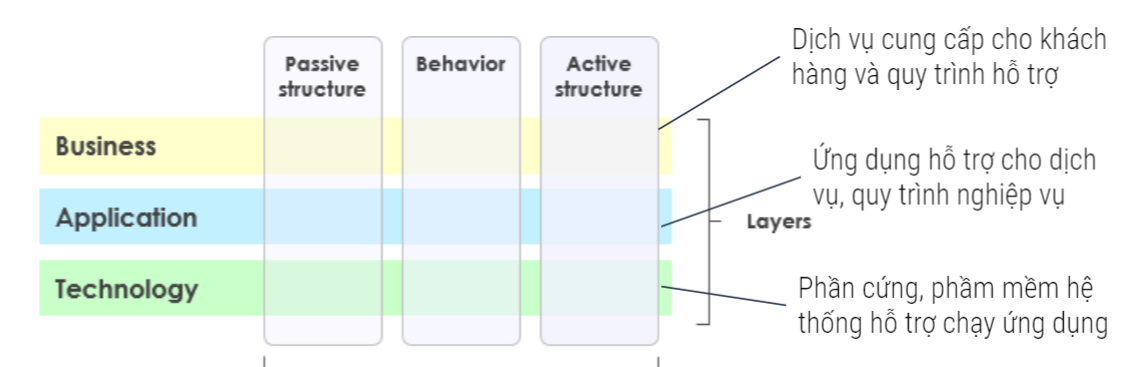
* Archimate = Architecture Animate.
* ArchiMate là một ngôn ngữ mô hình hóa kiến ​​trúc doanh nghiệp mở và độc lập để hỗ trợ mô tả, phân tích và trực quan hóa kiến ​​trúc trong và trên các lĩnh vực kinh doanh một cách rõ ràng
* Mô tả gọn gàng, rõ ràng, dễ áp dụng một Kiến trúc Doanh nghiệp.

## Câu 2 : Kiến trúc doanh nghiệp là gì ?

* Kiến trúc Doanh nghiệp của một tổ chức, hệ thống làbản thiết kế, quy hoạch tổng thể và thống nhất từ đầu đến cuối cho toàn bộ quá trình xây dựng và phát triển của tổ chức, hệ thống đó.

## Câu 3 : Khung lỗi archimate ?





## Câu 4 : Các loại biểu đồ Archimate ?

* Biểu diễn mục tiêu : Phân tích những động cơ, lý do dẫn đến sự định hướng thiết kế hoặc thay đổi Kiến trúc Doanh nghiệp
* Biểu diễn mô hình kinh doanh : Mô hình Kinh doanh là kế hoạch tạo ra lợi nhuận của một công ty, nhằm xác định:
* Sản phẩm, dịch vụ kinh doanh
* Thị trường mục tiêu
* Chi phí dự kiến
* Phân lớp
* Tương tác
* Quy TRình
* Khái niệm mô hình dữ liệu : Mô hình dữ liệu này xác định những gì hệ thống chứa. Mô hình này thường được tạo bởi các bên liên quan của doanh nghiệp và Kiến trúc sư dữ liệu. Mục đích là để tổ chức, phạm vi và xác định các khái niệm và quy tắc kinh doanh.
* Mô hình dữ liệu : có thể được sử dụng để mô hình hóa thông tin mức ứng dụng chi tiết, logic và các mối quan hệ của chúng với mục tiêu phát triển.
* Lợi ích của data modeling:
* Cải thiện các nguồn dữ liệu
* Thiết kế database hiệu quả
* Quản lý doanh nghiệp hiệu quả
* Hỗ trợ nâng cấp BI
* Tăng khả năng tích hợp
* Nền tảng công nghệ
* Application : Kiến trúc giải pháp (The Solution Architecture) là một khối xây dựng tự chủ, độc lập về mặt logic và vật lí của EA. Xác định hành vi và cấu trúc của một giải pháp duy nhất. Nó cũng là một phần của EA (mô hình Nghiệp vụ).
* Mẫu thiết kế ứng dụng.
* Mô hình thành phần.
* Database
* Tích hợp ứng dụng.
* Sơ đồ tuần tự.
* Các kiểu tích hợp ứng dụng.
* Use case.

# AN INTRODUCTION  TO MODELS AND RML

**(Mô hình hóa yêu cầu)**

## Câu 1 : Mô hình hóa yêu cầu là gì ? (RML: Requirements Modeling Language)

* Là một trong những cách hiệu quả nhất để xác định các yêu cầu phần mềm.
* Tạo ra 1 “bức tranh” giúp nhà phân tích đảm bảo các bên liên quan hiểu giải pháp.
* Là một ngôn ngữ đơn giản.
* Để mô hình hóa các yêu cầu một cách trực quan cho các bên liên quan (điều hành, kinh doanh, kỹ thuật).
* Sắp xếp và trình bày một lượng lớn thông tin, giúp xác định những thông tin còn thiếu và cung cấp ngữ cảnh cho các chi tiết.
* Cho phép phân tích nhanh chóng một lượng lớn thông tin bằng trí nhớ ngắn hạn.

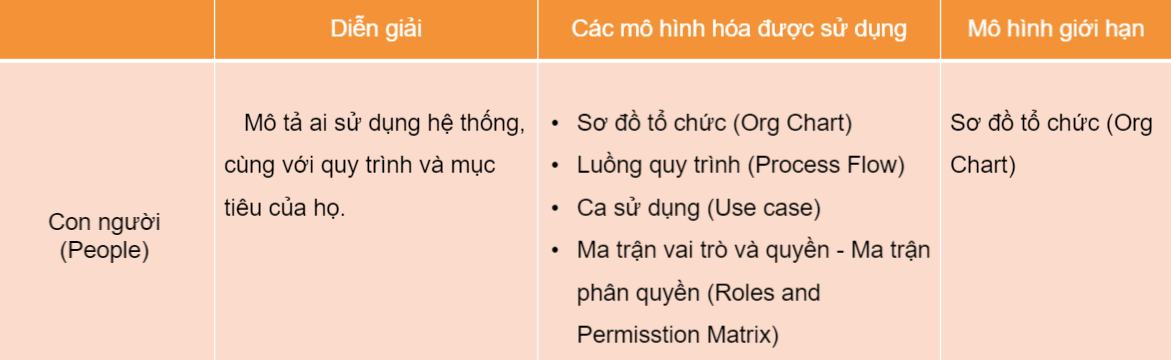
## Câu 2 : So sánh UML (ngôn ngữ mô hình thống nhât) và RML ( Ngôn ngữ mô hình yêu cầu)?

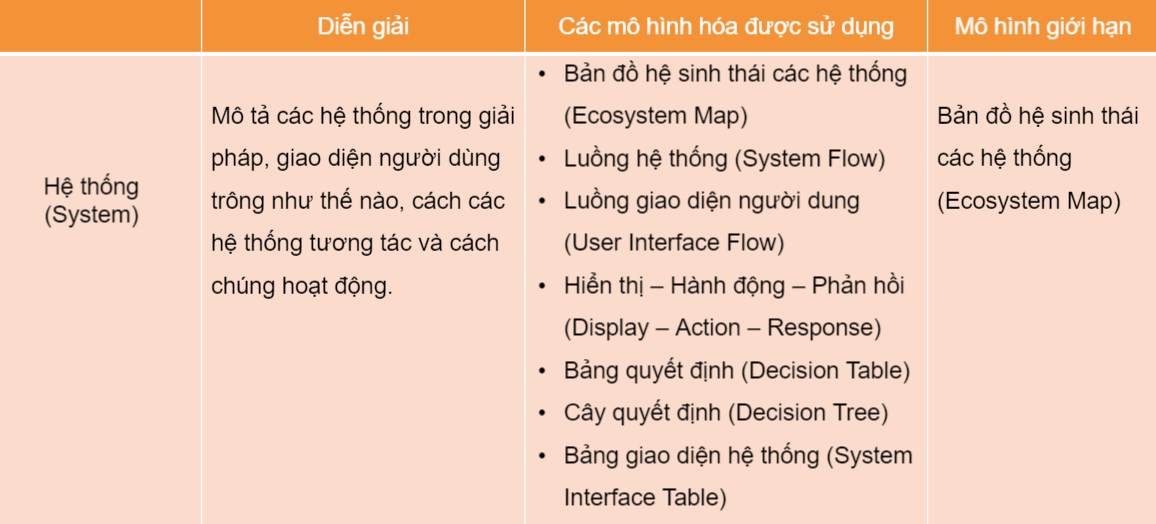
* UML là ngôn ngữ:
* Chỉ định thiết kế trực quan của hệ thống phần mềm
* Thiếu các mô hình ràng buộc các yêu cầu kinh doanh
* Phức tạp với các bên kinh doanh
* RML là ngôn ngữ:
* Cú pháp đơn giản
* Mục đích cung cấp ngữ nghĩa và cú pháp nhất quán
* Sử dụng cho toàn bộ nhóm, không giới hạn lĩnh vực của các bên liên quan

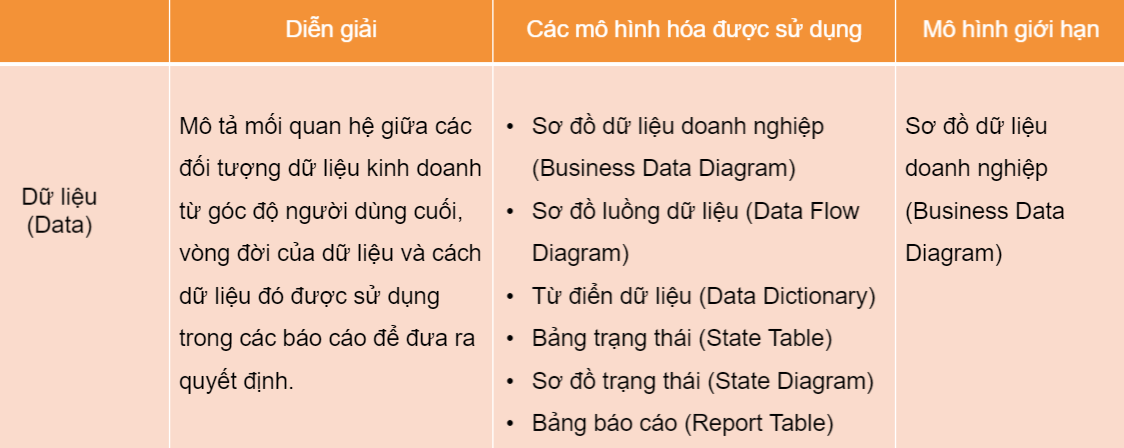
## Câu 3 : Phân loại RML :

* Các mô hình RML được phân thành:
* Mô hình mục tiêu (Objectives Models)
* Mô hình con người (People Models)
* Mô hình hệ thống (System Models)
* Mô hình dữ liệu (Data Models)









# ArchiMate Elements

**(Mô hình kiến trúc)**

Câu 1: chưa tổng hợp đươc @@

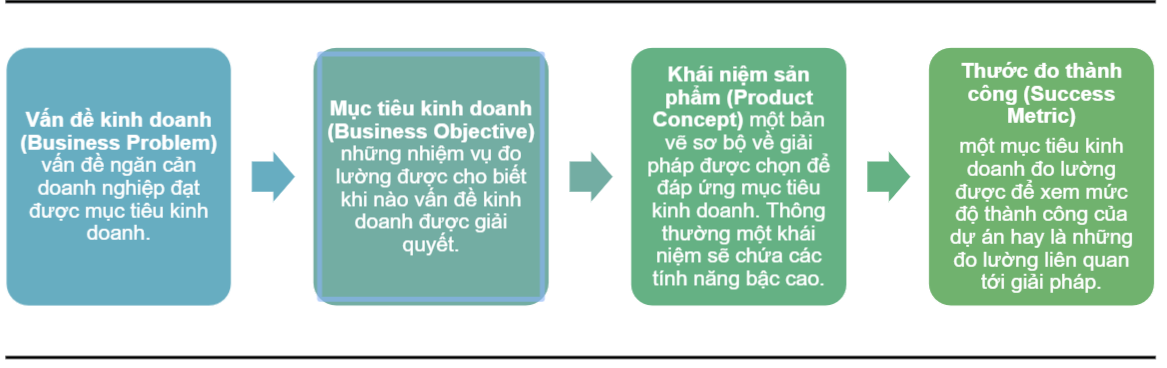
# Objectives Model(Mô hình mục tiêu kinh doanh)

**(Mô hình mục tiêu kinh doanh)**

## Câu 1 : Mô hình mục tiêu kinh doanh là gì ? (Business Objectives Model)

* Mô hình Mục tiêu Kinh Doanh - Business Objectives Model là một mô hình mục tiêu được các doanh nghiệp sử dụng để giải quyết các vấn đề kinh doanh bằng cách đưa ra những mục tiêu, từ đó đưa ra một sản phẩm (chương trình, chính sách,...) nhằm giải quyết vấn đề kinh doanh đó. Mô hình này giúp các chủ doanh nghiệp tập trung phát triển những giải pháp hợp lý để giải quyết vấn đề của họ và không đi lệch hướng.

## Câu 2 : Các element cơ bản của mô hình kinh doanh?



## Câu 3 : Ưu điểm mô hình mục tiêu kinh doanh ?



## Câu 4 : Object chain là gì ?

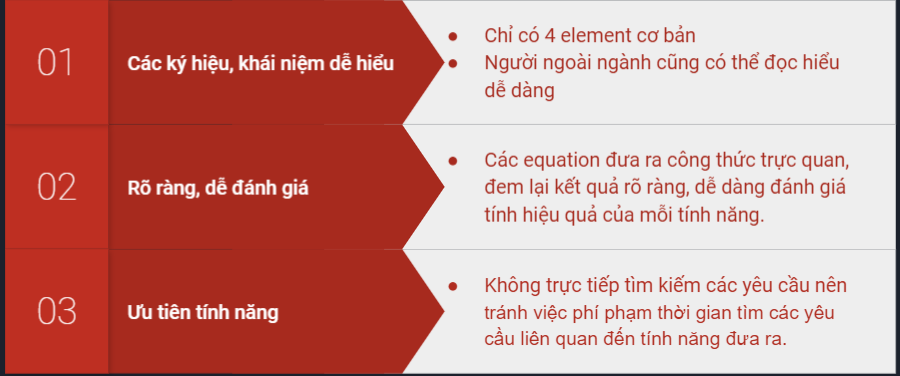
* Objective chain là mô hình chuỗi mục tiêu có thể liên kết tính năng với mục tiêu kinh doanh giúp xác định được giá trị tương đối mà mỗi tính năng đem lại. Bao gồm các mục tiêu kinh doanh ở đầu hệ thống và các tính năng ở cuối hệ thống phân cấp.

## Câu 5 : Các element của Objective Chain ?

Một Objective Chain gồm 4 element cơ bản:

* Business objective là element dùng để xác định mục tiêu kinh doanh mu đạt được.
* Objective factor là element dùng để đưa ra những yếu tố có thể giúp đạt được mục tiêu kinh doanh đưa ra. Mỗi một Objective factor sẽ đóng góp vào việc giải quyết cho vấn đề ở bậc trên nó.
* Objective equation đưa ra phương trình tính toán giá trị mà mỗi Objective factor có thể đem lại được.
* Feature là tính năng và kết quả dựa vào số liệu thực tế đạt được khi  thực hiện mỗi Objective factor.

## Câu 6 : Ưu điểm của Objective Chain?



## Câu 7 : Key Performance Indicator Model là gì ?

* Chỉ báo Hiệu suất Chính (KPI)  là một số liệu được sử dụng để đánh giá mức độ thành công của một bước quy trình hoặc hành động.
* KPIM là mô hình ​​kết hợp các KPI vào một quy trình công nghệ để giúp thêm thông tin ưu tiên xung quanh các yêu cầu, được kết nối với bước quy trình mà KPI tham chiếu.

## Câu 8 : Ưu điểm của Key Performance Indicator Model :



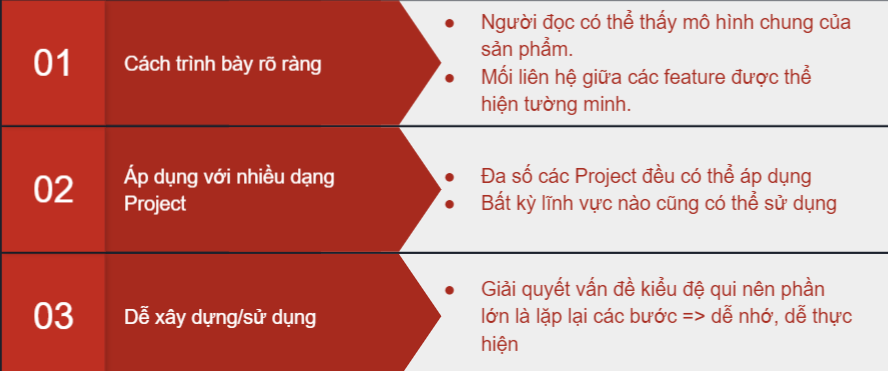
## Câu 10 : Nên sử dụng KPIM khi nào ?

* **Nên :**
* Bất kỳ dự án nào mà có các hệ thống hoặc quy trình hiện có mà bạn cần bảo trì hoặc nâng cấp.
* **Không nên :**
* Đối với các dự án đang triển khai chức năng hoàn toàn mới
* Đối với phần mềm không có quy trình nghiệp vụ nào (chẳng hạn như một số phần mềm đóng gói)

## Câu 11 : Feature tree là gì ?

* Cây tính năng là một mô hình dùng để sắp xếp, tổ chức các tính năng của một sản phẩm, ứng dụng,... giúp cung cấp một cái nhìn tổng quan về sản phẩm

## Câu 12 : ưu điểm



## Câu 13 : Requirements Mapping Matrix là gì ?

* Đây là 1 ma trận dùng để ánh xạ thông tin (information) sang các yêu cầu (requirements) và các nguyên tắc nghiệp vụ (business rules)
* Thường sử dụng nhất là ánh xạ Process Flow steps (bước quy trình) sang Requirements (yêu cầu), và từ Requirements sang Business Rules (quy tắc nghiệp vụ)
* A traceability matrix cũng gần giống như RMM thể hiện mối quan hệ nhiều - nhiều (many to many) của các mối quan hệ, được sử dụng để theo dõi các yêu cầu và kiểm tra các yêu cầu này đã được đáp ứng đủ hay chưa

## Câu 14 : ưu điểm

* Dễ quan sát
* Kiểm tra được yêu cầu bị thiếu
* Xác định các yêu cầu không liên quan hoặc các bước bị thiếu

# ArchiMate Relationships

## Câu 1 : Archimate Relationship là gì ?

* là một kết nối giữa khái niệm nguồn và đích.
* Có 4 loại quan hệ:
* Mối quan hệ có cấu trúc        (Structural relationships)
* Mối quan hệ phụ thuộc   (Dependency relationships)
* Mối quan hệ động                  (Dynamic relationships)
* Các mối quan hệ khác

## Câu 2 : Mối quan hệ có cấu trúc :

+ Mối quan hệ tổ hợp (Composition Relationship)

+ Mối quan hệ tập hợp (Aggregation Relationship)

+ Mối quan hệ hiện thực hóa (Realization Relationship)

+ Mối quan hệ giao việc (Assignment Relationship)

## Câu 3 : Mối quan hệ phụ thuộc :

+ Mối quan hệ phục vụ (Serving Relationship)

+ Mối quan hệ truy cập (Access Relationship)

+ Mối quan hệ ảnh hưởng (Influence Relationship)

+ Mối quan hệ phối hợp (Association Relationship)

## Câu 4 : Mối quan hệ động :

+ Mối quan hệ kích hoạt (Triggering Relationship)

+ Mối quan hệ theo luồng(Flow Relationship)

## Câu 5 : Các mối quan hệ khác :

+ Mối quan hệ chuyên biệt (Specialization Relationship)

+ Mối nối (Junction)

# Metamodel

## Câu 1 : Metamodel là gì ?

* Là  1 dạng mô hình hoá yêu cầu : dùng để  xác định và phân tích các yêu cầu trong các hoạt động kinh doanh của doanh nghiệp.
* Hệ thống lại các đối tượng tham gia.
* Nó được các kỹ sư phần mềm mô tả lại những quy trình phức tạp có trong hệ thống và được kiểm thử bởi các chuyên gia.
* Xác định trước những vấn đề sẽ gặp phải trong quá trình xây dựng phần mềm.
* Chỉ ra mối quan hệ giữa các quy trình thông qua các sơ đồ ký hiệu (UML, BPMN, GPML,...)

=> **Đưa ra cái nhìn tổng quan về các quy trình có  trong hệ thống.**

## Câu 2 : Tính năng chính trong metamodel :

* Classifier
* Relationship
* Style

## Câu 3 : Số loại trong metadata model :

* Dùng để phân tích và xây dựng mô hình phục vụ cho các vấn đề được xác định trước.
* Giúp đưa ra những khuôn mẫu, ràng buộc, quy tắc cho các  quy trình.
* Tập hợp các ngữ cảnh, đối tượng, thuộc tính của các đối tượng và quan hệ

## Câu 4 : Metadata model là gì ?

* Dùng để phân tích và xây dựng mô hình phục vụ cho các vấn đề được xác định trước.
* Giúp đưa ra những khuôn mẫu, ràng buộc, quy tắc cho các  quy trình.
* Tập hợp các ngữ cảnh, đối tượng, thuộc tính của các đối tượng và quan hệ

## Câu 5 : Meta-process modeling

* Nó tạo ra các quy trình linh hoạt dựa trên các tài liệu và có cần có sự tham gia  trực tiếp từ các các chuyên gia. Nhằm tăng cường tính tái sữ dụng và kế thừa
* Nó tập trung vào quá trình phân tích và xây dựng quy trình.
* Khó khăn của quy trình này là cải thiện và làm cho chúng phát triển vì theo thời gian thì các mô hình và quy trình kinh doanh cũng thay đổi mà các quy trình nền tảng cũng cần được thay đổi. Chính vì vậy các quy trình mới có thể được xây dựng và cải tiến dựa trên các quy trình hiện có.

## Câu 6 : Profile diagram là gì ?

* Về cơ bản nó là cơ chế cho phép bạn mở rộng và tuỳ chỉnh UML bằng cách thêm mới các thành phần mới.
* Tạo ra các thuộc tính mới và chỉ định ngữ nghĩa mới để UML giải quyết những vấn dề của bạn.
* Nó có 3 thành phần cụ thể: Stereotypes, tagged values, contraints

# System models

**(Mô hình hệ thống)**

## Câu 1 : Ecosystem Map

* Bản đồ hệ sinh thái (Ecosystem Map) mô tả bản thân các hệ thống trông như thế nào và cách chúng tương tác với người dùng và hệ thống khác.
* Là một mô hình hệ thống RML, hiển thị tất cả các hệ thống trong hệ sinh thái giải pháp (Solution Ecosystem) , cho phép bạn đảm bảo đã phân tích toàn bộ hệ thống.
* Mỗi hệ thống không nhất thiết là một hệ thống vật lý, mà thường đại diện cho một ứng dụng hoặc một hệ thống logic.
* Các bược tạo Ecosystem map :

+ Bước 1 : Xác định các hệ thống

+ Bước 2 : Xác định các giao diện

+ Bước 3 : Nối các biểu đồ với nhau

* Sử dụng :
* Nên là một trong những mô hình đầu tiên của dự án vì nó xác định được bức tranh tổng quan về dự án và dễ dàng truyền tải nó đến những người trong dự án
* Giúp kiểm tra dễ dàng các giao diện giữa các hệ thống, đảm bảo tất cả các giao diện đều được xét tới, việc tìm ra lỗi thiếu một quy trình liên quan một hệ thống cũng dễ dàng hơn
* Không cung cấp thông tin chi tiết về các tương tác cũng như trình tự các tương tác hay quyết định xử lý.

## Câu 2 : System flow :

* Luồng hệ thống (System Flow) là một mô hình RML mô tả hoạt động của hệ thống, trình tự và logic của chúng. Giúp doanh nghiệp hiểu các tương tác phức tạp của hệ thống theo cách trực quan
* Tuy giống Luồng quy trình (Process Flow) về thuật ngữ, khuôn mẫu và kết quả cuối cùng. Nhưng về nội dung Luồng hệ thống dùng để mô tả các hành động tự động của hệ thống, không phải của người dùng hay toàn bộ hệ thống

## Câu 3 : User interface flow :

* Luồng giao diện người dùng (UI Flow) là một mô hình hệ thống RML thể hiện cách mà người dùng sẽ định hướng giao diện của họ, tương tự như một bản đồ thiết kế công viên. Một UI Flow bao gồm các màn hình và những đường dẫn giữa chúng.
* Khả năng sử dụng của phần mềm thường xác định mức độ hiệu quả và hài lòng của người dùng. Cho dù mọi tính năng đều có trong phần mềm, nếu việc truy cập đến chúng trở nên khó khăn, người dùng cũng sẽ không chấp nhận phần mềm của bạn.
* Các bước tạo ui flow :

+ Xác định phạm vi màn hình

+ Xác định màn hình

+ Tạo các chuyển tiếp

+ Các nhãn dán kích hoạt

* Sử dụng Ui Flow:
* UI Flow là một mô hình tốt để giao tiếp với các nhóm kinh doanh và kỹ thuật về cách người dùng sẽ điều hướng trong toàn bộ hệ thống. Các UI Flow giúp xác định điều hướng, xác thực đường dẫn điều hướng và tối ưu hóa khả năng sử dụng.
* Hỗ trợ trong việc thiết kế giao diện người dùng
* Tối ưu hoá khả năng sử dung.
* Dùng trong phát triển các trường hợp thử nghiệm

## Câu 4 : Display Action - Respone

* Mô hình Hiển thị-Hành động-Phản hồi (DAR) là một mô hình hệ thống RML cho phép bạn ghi lại một cách có hệ thống các cách thức hợp lệ trong đó hệ thống hiển thị những điều sau:
  + 1. Các phần tử giao diện người dùng (UI) phụ thuộc vào dữ liệu trong màn hình
  + 2. Cách hệ thống phản hồi lại các hành động mà người dùng có thể thực hiện
* Mô hình DAR nắm bắt cả cách chúng hiển thị và cách hệ thống phản ứng với các hành động của người dùng với các phần tử đó.
* Các bước tạo mô hình DAR :

+ Chuẩn bị màn hình

+ Hoàn thành mô tả

+ Mô tả hiển thị

+ Mô tả hành vi

* Nên dùng khi nào:
* Sử dụng:
  + Sử dụng mô hình DAR khi sợ rằng giao diện người dùng không hoạt động như mong đợi
* Không nên sử dụng:
  + Nếu không có giao diện người dùng, các trang chỉ hiện thị các thông tin cứng như giới thiệu, đơn giản, rất ít các phần tử hướng dữ liệu thì không nên dùng. Vì mô hình Dar rất tốn thời gian để tạo nên chỉ tạo trong trường hợp thật sự cần thiết.

## Câu 5 : Decision Tables:

* Bảng Quyết định là một mô hình hệ thống RML giúp phân tích tất cả các hoán vị của logic phức tạp một cách toàn diện.
* Bảng Quyết định được sử dụng để trả lời câu hỏi “Trong điều kiện nào thì kết quả này sẽ xảy ra?” Hoặc “Với những điều kiện này, tôi nên chọn kết quả nào?”
* Định dạng của mô hình cho phép bạn dễ dàng đảm bảo rằng tất cả các điều kiện có thể đang được kiểm tra và hành động đúng đắn.
* **Tạo Decision tables :**

+ Xác định điều kiện

+ Xác định lựa chọn

+ Xác định kết quả

+ Đánh dấu kết quả hợp lệ

+ Đơn giản hóa bảng quyết định

## Câu 6 : Decision trees

* Cây quyết định là một mô hình hệ thống RML. Cây Quyết định cho phép bạn lập mô hình logic phức tạp và phân tích một loạt các quyết định.
* Cây Quyết định có thể được sắp xếp theo thứ tự hoặc không có thứ tự.
  + Cây quyết định có thứ tự được sử dụng nếu có thứ tự trong đó các quyết định được đưa ra và là loại cây quyết định phổ biến nhất.
  + Cây quyết định không có thứ tự được sử dụng để đơn giản hóa một tập hợp các quyết định không có thứ tự

## Câu 7 : System Interface Tables

* Bảng Giao diện Hệ thống (The System Interface Table) là một mô hình hệ thống RML (RML systems model)
* Bảng Giao diện Hệ thống (The System Interface Table) hiển thị chi tiết về sự tương tác giữa hai hệ thống theo quan điểm về những gì mà chúng có liên quan với nhau về nội dung hơn là quan điểm về mặt kỹ thuật.
* Bảng Giao diện Hệ thống mô tả thông tin được chuyển, tần suất chuyển và mức độ truyền của thông tin. Mục đích là để cung cấp cho nhóm kỹ thuật sự hiểu biết về các ràng buộc kinh doanh xung quanh giao diện có thể ảnh hưởng đến các quyết định thiết kế.

# Archimate Diagram Types

## Câu 1 : Most useful diagram :

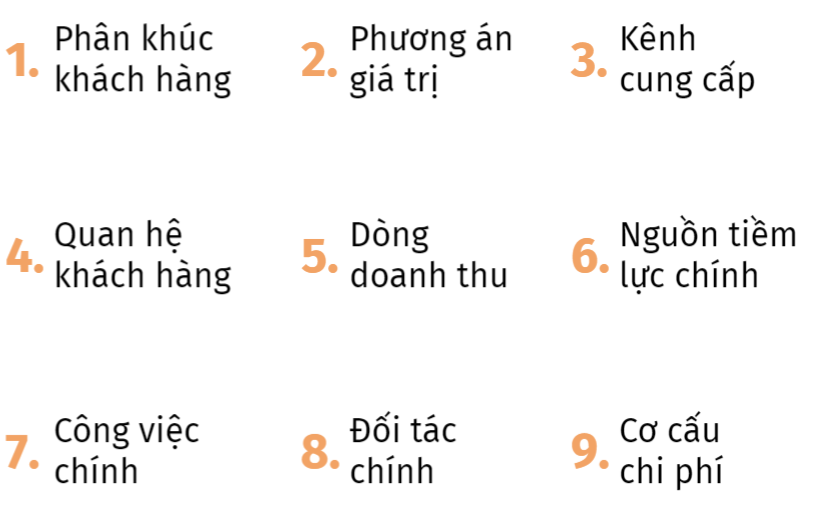
* **Goals view :**
* Goal view là 1 loại diagram đa dụng, có thể được sử dụng cho nhiều mục đích khác nhau như:
* Biểu diễn kế hoạch của 1 tổ chức
* Hoặc yêu cầu của 1 mục tiêu phát triển
* …
* Goal view được mô hình chủ yếu bằng các thành phần thuộc ArchiMate Motivation và Strategy
* **Layered View:**
* Layered View sử dụng các thành phần thuộc các lớp **Business**, **Application** và **Technology.**
* Cho phép mô tả mục tiêu phát triển (development target) thành các “stack” phân lớp: business ở trên, tiếp theo là application và sau đó là technology
* Tổng thể hệ thống được biểu diễn rõ ràng với các quan hệ cần thiết giữa các đối tượng, giúp nắm rõ được mục tiêu phát triển
* Một số dạng layered view:
* Customer Service Journey (dưới góc nhìn của khách hàng)
* Service Blueprint
* Swimline Process View
* Service Design View
* **Interaction View:**
* Biểu đồ tương tác, được sử dụng để mô hình hóa các mối quan hệ giữa các tác nhân, các tiến trình, ứng dụng
* Về cơ bản, Interaction View diagram thể hiện được luồn thông tin GÌ theo HƯỚNG nào, từ ĐÂU đến ĐÂU.
* Gồm 3 loại:
* Actor Interaction View (Biểu đồ tác nhân)
* Process Interaction View (Biểu đồ tiến trình)
* Application Interaction View (Biểu đồ ứng dụng)

## Câu 2 : Valueble Diagram :

* **Conceptual Data Model View** : Mô hình dữ liệu khái quát, mô hình này được sử dụng để mô hình hóa những khái niệm (concepts) và các mối quan hệ (relations) của mục tiêu phát triển (development target)
* **Data Model View :** Mô hình dữ liệu, là dạng mô hình cụ thể chi tiết, ứng dụng thực tế và mối quan hệ của chúng với mục tiêu phát triển.
* **Technology Platfrom View :** Mô hình nền tảng công nghệ, được sử dụng để mô hình hóa các nền tảng cơ sở hạ tầng (underlaying infrastructure) và triển khai của một ứng dụng (phần mềm, severs, networks,..)

## Câu 3 : Additional Diagram :

* **Process View :**
* Process view được mô hình chủ yếu bằng các thành phần ở lớp Business: Business Process, Business Actor, Business Role, Business Object and Business Event.
* Process view sẽ thể hiện từng bước chi tiết của một quy trình diễn ra thông qua Business Process, mối liên hệ tới những Business Object, Business Actor khác và sẽ ảnh hưởng tới nhũng Business Event nào.
* Cụ thể hoá một quy trình nghiệp vụ phức tạp.
* Dễ dàng cho việc quản lí cũng như theo dõi quy trình đó.
* **Business Model Canvas :**
* Business Model Canvas (BMC) là một công cụ xây dựng mô hình bán hàng tối tân được thiết kế bởi Alexander Osterwalder và Yves Pigneur
* Các thành tố :



* Lợi ích : cô động, linh hoạt, bài bản.
* **Service Blueprint :**
* Service Blueprinting (viết tắt là SBP) được thể hiện như một mối tương quan giữa nhiều thành phần khác nhau trong một quy trình dịch vụ/sản phẩm và có mối liên quan mật thiết với Customer Journey.
* Được sử dụng khi cần khái quát một sơ đồ hoạt động phức tạp với sự liên quan của nhiều dịch vụ/sản phẩm có quan hệ với nhau.
* Tác dụng :
* Khái quát đơn giản hoạt động của dịch vụ/sản phẩm.
* Đánh giá hiệu quả hoạt động của dịch vụ/sản phẩm.
* Dễ dàng mở rộng, phát triển dịch vụ mới.

# Data Model

## Câu 1 : Khái niệm ?

* Data model (mô hình dữ liệu) là một mô hình trừu tượng tổ chức các phần tử của dữ liệu và chuẩn hóa cách chúng liên quan với nhau và với các thuộc tính của các thực thể trong thế giới thực.

## Câu 2 : Business data diagram :

* Là mô hình dữ liệu **biểu diễn mối quan hệ giữa các đối tượng dữ liệu.**
* Người dùng BDD **sắp xếp dữ liệu giống cách mà họ thực hiện tác vụ trên những dữ liệu mà họ thấy hợp lý.**
* Cách tổ chức dữ liệu có thể rất khác so với dữ liệu được tổ chức trên cơ sở dữ liệu.
* BDD **biểu diễn số lượng quan hệ mà một đối tượng sở hữu với một đối tượng khác,** Đó được gọi là ***cardinality*** của mối quan hệ.
* **Sử dụng :**
* BDD biểu diễn các đối tư ợng dữ liệu từ quan điểm của doanh nghiệp chứ không phải của nhóm kĩ thuật.
* BDD chỉ ra cho bạn thấy cái nhìn tổng quát về những đối tượng có liên quan và đặc biệt là các mối quan hệ từ cái nhìn đầu tiên.
* Giúp bạn dễ dàng xác định các đối tượng dữ liệu nghiệp vụ và các mối quan hệ bị thiếu một cách dễ dàng bằng các câu hỏi đơn giản như: “Thực sự có cần…”, “Tại sao...lại..”, “Liệu…” …
* Mặc dù BDD không đại diện trực tiếp cho thiết kế dữ liệu nhưng nó giúp các Team Database hiểu các mối quan hệ để hỗ trợ cho việc thiết kế.

## Câu 3 : Data Flow Diagram :

* Là mô hình dữ liệu RML **minh họa trực quan các luồng dữ liệu.**
* DFD chỉ **tập trung vào cách dữ liệu truyền đi**, được thao tác hoặc sử dụng trong các tiến trình.
* Điều quan trọng nhất là nó chỉ ra những **thông tin nào cần phải có trước khi cho thực hiện một tiến trình**.
* **Cách tạo :**
* Là xác định các **đối tượng dữ liệu nghiệp vụ**.
* Là xác định các **tiến trình, quy trình** thao tác dữ liệu.
* Là xác định các **thực thể bên ngoài**.
* Là xác định các **luồng dữ liệu** một cách thích hợp. Kiểm tra lại và bổ sung.

## Câu 4 : Data Dictionary :

* Là mô hình dữ liệu mô tả chi tiết về dữ liệu trong hệ thống mức độ trường dữ liệu.
* Mô hình hóa các trường dữ liệu thuộc đối tượng dữ liệu.
* Không tập trung vào dữ liệu thực tế trong CSDL hay bản thiết kế CSDL mà chỉ tập trung vào cách mà các bên liên quan mô tả nó.
* **Thành phần :**
* Đối tượng dữ liệu (Business Data Objects).
* Trường dữ liệu (Fields).
* Thuộc tính (Properties).
* **Thuộc tính :**
* Các thuộc tính định nghĩa đối tượng dữ liệu và trường dữ liệu.
* Các thuộc tính mô tả các quy tắc nghiệp vụ quản lý dữ liệu đó.
* Các thuộc tính quyền hạn.
* **Cách xác định :**
* Xem xét danh sách các thuộc tính trong phần Template và xác định những thuộc tính cần thiết.
* Xác định các đối tượng dữ liệu và các trường của nó.
* Mô tả các trường dữ liệu.
* Cung cấp bảng danh lục dữ liệu.

## Câu 5 : State table :

* Là mô hình dữ liệu dùng để xác định tất cả các trạng thái và tất cả các bước chuyển tiếp giữa các trạng thái cho đối tượng dữ liệu.
* Tất cả các trạng thái đều rời rạc với nhau, và mỗi đối tượng chỉ tồn tại ở một trạng thái nhất định trong một thời điểm nhất định.
* **Sử dụng :**
* Hỗ trợ các mô hình khác:
  + Các trạng thái có thể trở thành điều kiện tiên quyết để kích hoạt hiển thị hoặc thực hiện hành vi trong giao diện người dùng.
* Yêu cầu phát sinh:
  + Cần những điều kiện để quá trình chuyển đổi xảy ra?
  + Hành động nào bắt đầu quá trình chuyển đổi?
  + Đầu ra của quá trình chuyển đổi là gì?
  + Những hành động hoặc dữ liệu chuyển đổi nào xảy ra do quá trình chuyển đổi?
* Nên sử dụng:
  + các đối tượng trải qua các trạng thái thay đổi phức tạp.
* Không nên sử dụng:
  + Khi đối tượng có ít trạng thái, hoặc các trạng thái đơn giản.
  + Khi quá trình biến đổi là tuyến tính, không phân nhánh nhiều.

## Câu 6 : Report Table

* Là mô hình dữ liệu RML cung cấp hệ thống có cấu trúc ghi lại toàn bộ thông tin cần thiết để hoàn thành bản report, bao gồm: dữ liệu được trình bày, dạng mẫu xuất ra, cái nhìn tổng quát đến chi tiết của bản report, các yêu cầu cho việc vận dụng và tương tác với dữ liệu.
* Mặc dù phù hợp cho việc báo cáo thuần túy, Report Table cũng hữu ích trong việc ghi lại những yêu cầu về khả năng hiển thị cấu trúc của thông tin chứa nhiều trường dữ liệu.
* Tồn tại với nhiều lý do, có thể một trong số chúng không rõ ràng, mang tính tạm thời nhưng chủ yếu là hỗ trợ các quyết định trong các quy trình hệ thống.

# Model in the big pictures

## Câu 1 : Lựa chọn mô hình cho dự án

* **Giai đoạn hình dung :**
* Xảy ra trước khi một dự án được tiến hành.
* Giúp khám phá, xác định các mục tiêu

-> Phát triển tầm nhìn, mở rộng quy mô cho dự án.

* Mô hình thường được sử dụng:

+  Business objectives model ( Xác định độ ưu tiên)

+  Feature tree

+ Org chart (Mô hình con người)

+ Ecosystem map (Xác định giao diện)

* Nên sử dụng thêm công cụ quản lý yêu cầu.
* **Giai đoạn lập kế hoạch :**
* Xác định cách thức hoạt động để  đạt được giá trị mong đợi.
* Kiểm tra kiến trúc và quy trình yêu cầu trên 1 phần của dự án.
* Thực hiện hầu hết các thao tác và phân tích.
* Tiếp tục cập nhật danh sách các vấn đề yêu cầu.
* Mô hình thường được sử dụng:

+  KPIM (Hiệu suất)

+  Objective Chain ( Phạm vi các yêu cầu)

+ Business Data Diagram

+ Process Flow

* Các tài liệu có thể có (yêu cầu nghiệp vụ, đặc điểm yêu cầu hệ thống, đặc điểm yêu cầu phần mềm, thông số kỹ thuật,...)
* **Giai đoạn phát triển :**
* Xác thực, xác minh, giải thích việc sử dụng mô hình và phát triển thử nghiệm
* Cập nhật các mô hình ở trạng thái locked khi các thay đổi xảy ra.
* Nên tương tác với các nhà phát triển khi họ xây dựng các giải pháp và với những người thử nghiệm.
* Giải thích rõ ràng các yêu cầu chức năng thông qua

+  Process Flow

+  Sau đó dùng BDD

* Cập nhật và tạo thêm các tài liệu ( Kiểm tra sự chấp nhận của người dùng, và các tài liệu của các giai đoạn trước đó.
* **Giai đoạn khởi chạy :**
* Diễn ra sau khi hệ thống đã hoạt động và người dùng chấp nhận sử dụng.
* Có thể đo lường lợi tức đầu tư các mục tiêu kinh doanh.
* Sử dụng chỉ số hiệu suất chính KPI.
* Các tài liệu được tạo ra trong giai đoạn này: Bản trình bày cho giám đốc điều hành, tài liệu rút ra bài học, hồi cứu.
* **Giai đoạn đo lường :**
* Diễn ra sau khi hệ thống đã hoạt động và người dùng chấp nhận sử dụng.
* Có thể đo lường lợi tức đầu tư các mục tiêu kinh doanh.
* Sử dụng chỉ số hiệu suất chính KPI.
* Các tài liệu được tạo ra trong giai đoạn này: Bản trình bày cho giám đốc điều hành, tài liệu rút ra bài học, hồi cứu.

## Câu 2: Lựa chọn mô hình theo đặc điểm dự án :

* Đặc điểm mục tiêu
* Đặc điểm con người
* Đặc điểm hệ thống
* Đặc điểm về dữ liệu

## Câu 3 : Mô hình điều chỉnh :

* Gặp khó khăn khi cố gắng đưa thông tin vào cấu trúc không phù hợp
* Tuỳ chỉnh phổ biến: thêm màu để làm nổi bật, tạo khoảng rộng trong Data Dictionary
* Nên sửa đổi trước khi bắt đầu dự án

## Câu 4 : Sử dụng mô hình kết hợp

* Many different views :
* Using more than one model:

## Câu 5 : Lợi ích của kiến trúc tốt :

* Tất cả mô hình cần thiết đều được xem xét và tạo ra trong suốt dự án
* Số lượng mô hình thích hợp.
* Có một cơ chế nhất quán giữa mô hình này với mô hình khác.
* Nhà phân tích có thể tiếp cận dự án sớm và không phải dành nhiều thời gian để tổ chức các thông tin.
* Có thể ước tính đúng được nỗ lực, công sức dành cho mỗi yêu cầu

## Câu 6 : KIẾN TRÚC TỆ SẼ GÂY RA NHỮNG VẤN ĐỀ NGHIÊM TRỌNG

* Không có khả năng tái sử dụng .
* Không có sự nhất quán dẫn đến sự hiểu nhầm giữa bên kinh doanh và kĩ thuật.
* Không thể dự báo, lên kế hoạch, ước tính nỗ lực cần thiết để hoàn thành các yêu cầu.
* Đánh giá, xem xét lại các yêu cầu với những thông tin sai hoặc không liên quan.
* Không ai có thể hiểu được tại sao một yêu cầu bị thay đổi

## Câu 7 : Phụ lục A ( Quick lookup models Grids )

* Tác dụng :

● Là một bảng tóm tắt có dạng bảng lưới

● Mang đến cái nhìn nhanh trong quá trình chọn mô hình cho dự án

## Câu 8 : Phụ lục B ( Hưởng dẫn )

* Các ô/ký hiệu phải cùng kích thước
* Chữ trong cùng một loại ký hiệu  phải cùng kích thước
* Đánh số theo đúng thứ tự nhất quán
* Giữ khoảng cách cân bằng giữa các đối tượng
* Lời khuyên:
* Khi thông tin có thự tự, sắp xếp từ phía trên bên trái -> phía dưới bên phải
* Dùng thể chủ động
* Viết hoa nhất quán (đầu câu, tiêu đề)

# Microservice

## Câu 1 : Microservice là gì ?

* Microservice là một kỹ thuật phát triển phần mềm, một biến thể của kiến trúc hướng dịch vụ cấu trúc một ứng dụng như một tập hợp các dịch vụ được ghép lỏng lẻo. Trong kiến trúc microservice, các dịch vụ được xử lý tốt và các giao thức rất nhẹ.
* **Ưu điểm :**
* Dễ nâng cấp và scale up, scale down.
* Do tách biệt nên nếu 1 service bị lỗi, toàn bộ hệ thống vẫn hoạt động bình thường.
* Các service nằm tách biệt nhau nên chúng có thể được sử dụng các ngôn ngữ lập trình riêng, database riêng.
* Khi chia nhỏ các service, team size sẽ giảm và mọi người làm việc sẽ hiệu quả hơn.
* **Nhược điểm :**
* Các service giao tiếp với nhau qua mạng nên tốc độ không cao bằng monolith.
* Mỗi service sử dụng 1 database riêng nên việc đảm bảo tính đồng nhất trong dữ liệu sẽ trở nên phức tạp
* Sử dụng nhiều service nên việc theo dõi, quản lí sẽ phức tạp hơn
* Cần một đội ngũ có khả năng và trình độ

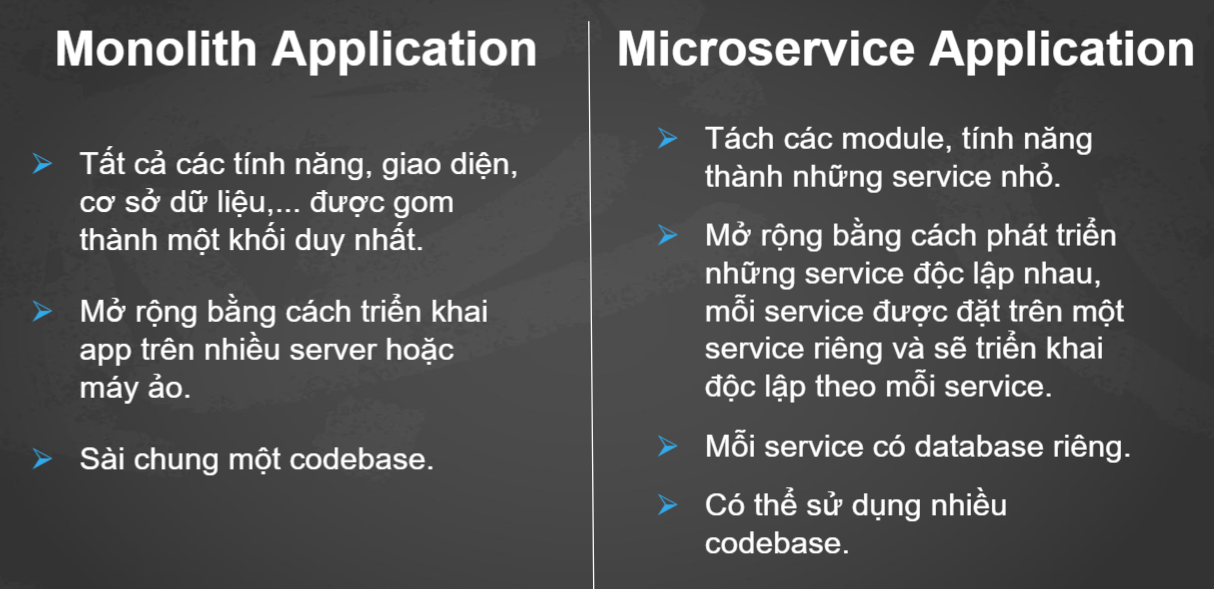
## Câu 2 : Monolithic Architecture :

* Monolithic Architecture được ví như một cái container lớn, trong đó chứa đựng tất cả các thành phần phần mềm của một ứng dụng được lắp ráp lại với nhau và được đóng gói chặt chẽ.
* **Ưu điểm :**
* Dễ dàng build, test, debug, deploy
* Dễ dàng build, test, debug, deploy
* Phù hợp với dự án có quy mô nhỏ
* **Nhược điểm :**
* Khi scale up lên, code base sẽ rất bự, khiến cho 1 member mới sẽ khó biết bắt đầu từ đâu
* Khó thay đổi công nghệ, ngôn ngữ, hoặc framework tiên tiến hơn
* Tính ổn định không cao
* Một tính năng thay đổi coi như cả hệ thống thay đổi

## Câu 3 : Message trong microservice :

* **Synchronous Messaging**
* REST là sự lựa chọn hàng đầu vì nó cung cấp hệ thống truyền tin đơn giản qua giao thức HTTP dạng request – response. Do đó, nhiều microservices sử dụng HTTP với API. Mỗi chức năng xuất ra API.
* Ngoài REST, Thrift, gRPC, GraphQL cũng được sử dụng để truyền tin đồng bộ.
* **Asynchronous Messaging**
* Tin nhắn không đồng bộ được dùng khi Client không muốn phản hồi ngay lập tức hoặc muốn hoàn toàn không phản hồi.
* Các giao thức tin nhắn không đồng bộ như AMQP, STOPM hay MQTT được sử dụng phổ biến.

## Câu 4 : So sánh Monolith và Microservice



# Micro Frontends

## Câu 1 : Micro Frontends là gì ?

* micro frontends là kiến trúc giúp phân tách những chương trình to lớn và phức tạp thành những phần nhỏ hơn, dễ quản lý hơn , trở nên tách biệt và giảm sự phụ thuộc giữa các thành phần. Công nghệ được lựa chọn, giải pháp phần mềm, các nhóm phát triển, quá trình phát hành của những thành phần, tất cả đều được vận hành và phát triển độc lập, tránh khỏi việc phải lên kế hoạch một cách quá chi tiết.
* **Lợi ích :**

+ Tăng số lượng các bản cập nhập

+ Code đơn giản và được phân chia.

+ Phát triển một cách độc lập

+ Các nhóm làm việc chủ động.

+ Khả năng nâng cấp, cập nhật, hoặc thậm chí làm lại những phần của giao diện một cách đẹp hơn

# Serverless Architectures

## Câu 1 : Serverless Architectures là gì ?

* Serverless được dùng để chỉ 2 khái niệm khác nhau, nhưng lại khá liên quan với nhau:
* Những ứng dụng chuyển phần lớn logic về front-end, không có server để làm back-end (serverless)  mà chỉ sử dụng các API của bên thứ 3 để thay thế. Đây là kiểu kiến trúc **Backend as a Service – BaaS**
* Khi lập trình viên phải tự viết code để làm **back-end**, thay vì deploy code này lên server như mô hình client-server thông thường, ta **deplay nó dưới dạng một Function** (**Function as a Service – FaaS)**. Function này có thể được gọi dưới dạng RestAPI hoặc chạy theo lịch đã sắp sẵn.
* **Các trường hợp sử dụng serverless :**
* Ứng dụng web
* Xử lý tin nhắn không đồng bộ. Chatbots cần khả năng tự động hóa
* Xử lý luồng quy mô cao
* Ứng dụng điện thoại di động back ends
* Công việc hàng loạt
* Xử lý đa phương tiện
* Xử lý sự kiện
* Xử lý dữ liệu
* Chatbots và trợ lý ảo
* Tự động hóa IT.

## Câu 2 : Các trường hợp sử dụng serverless

+ Backend: ứng dụng web, di động , IOT.

+ Xử lý dữ liệu : Xử lý tệp thời gian thực, xử lý luồng tg thực.

+ Chatbox

+ Trợ lý ảo

+ Xử lý đa phương tiện

## Câu 3 : Ưu điểm và nhược điểm serverless

* Ưu điểm : Kiến trúc này giúp thay thế việc quản lý vận hành nhiều máy chủ liên tục, từ đó có thể thay đổi quy mô linh hoạt, giảm chi phí.
* Nhược điểm : Tuy nhiên, cũng vì vậy mà khi sử dụng, ta cũng phải cân nhắc về độ trễ, lỗi, các giới hạn về bộ nhớ, thời gian, sự phụ thuộc các nhà cung cấp và chi phí ngầm.

# Evolutionary Database Design

## Câu 1 : Evolutionary Database Design là gì ?

* **Thiết kế cơ sở dữ liệu có tính tiến hóa (**Evolutionary Database Design**)** là việc thực hiện các cải thiện tăng dần đối với cơ sở dữ liệu, bằng việc tận dụng **tự động tái cấu trú**c và **tích hợp liên tục** để hỗ trợ các phương pháp Agile cho việc phát triển phần mềm,  áp dụng trên hệ thống đang ở trạng thái tiền sản xuất cũng như là những hệ thống đã được chuyển giao/phát hành.
* **Tái cấu trúc tự động** (Automated Refactoring)có nghĩa là thực hiện những sự thay đổi đến cơ sở dữ liệu mà không làm thay đổi đến hành vi của ứng dụng một cách tự động.
* **Tích hợp liên tục (** Continuous Integration**)** là một phương thức hoạt động trong việc phát triển phần mềm, ở đó các lập trình viên phải liên tục tích hợp những thay đổi của mình đến project ít nhất 1 lần trong ngày. Sau đó mỗi tích hợp sẽ được tự động build, test và deploy để sớm phát hiện lỗi, từ đó hạn chế được số lỗi về sau.

## Câu 2 : Các quy tắc trong Evolutionary Database Design :

**+** Những quản trị viên cơ sở dữ liệu phải cộng tác một cách chặt chẽ với các lập trình viên.

**+** Tất cả thay đổi đến cơ sở dữ liệu đều là "migrations".

**+**Những thay đổi đến cơ sở dữ liệu cần được kiểm soát theo phiên bản.

**+** Mỗi người đều có một thể hiện cơ sở dữ liệu riêng.

**+** Lập trình viên phải liên tục tích hợp những thay đổi của cơ sở dữ liệu.

**+** Cơ sở dữ liệu cần có một lượng dữ liệu có sẵn hợp lý.

**+** Tất cả sự thay đổi cơ sở dữ liệu đều phải là "refactoring".

**+** Quá trình "refactoring" cần được tự động hóa.

**+** Lập trình viên có thể cập nhật cơ sở dữ liệu của họ theo yêu cầu.

**+** Tách biệt rõ ràng tất cả "database access code".

**+** Release thường xuyên.

## Câu 3 : Hạn chế của Evolutionary Database Design :

+ Trở ngại văn hóa

+ Cần thời gian học hỏi

+ Phức tạm

# Two-Stack CMS

## Câu 1: Content Management System là gì ?

* **C**ontent **M**anagement **S**ystem (hệ quản trị nội dung) là một phần mềm dùng để tạo, quản lý, chỉnh sửa nội dung một cách dễ dàng trên website.
* Hệ thống CMS giúp tiết kiệm thời gian, chi phí xây dựng, vận hành, quản lý và bảo trì website.
* Được sử dụng rộng rãi bởi các công ty, cá nhân.

## Câu 2 : Phân loại :

**Open source CMS**

* Miễn phí sử dụng.
* Đầy đủ những tính năng cơ bản
* Không yêu cầu hiểu biết kĩ thuật cao

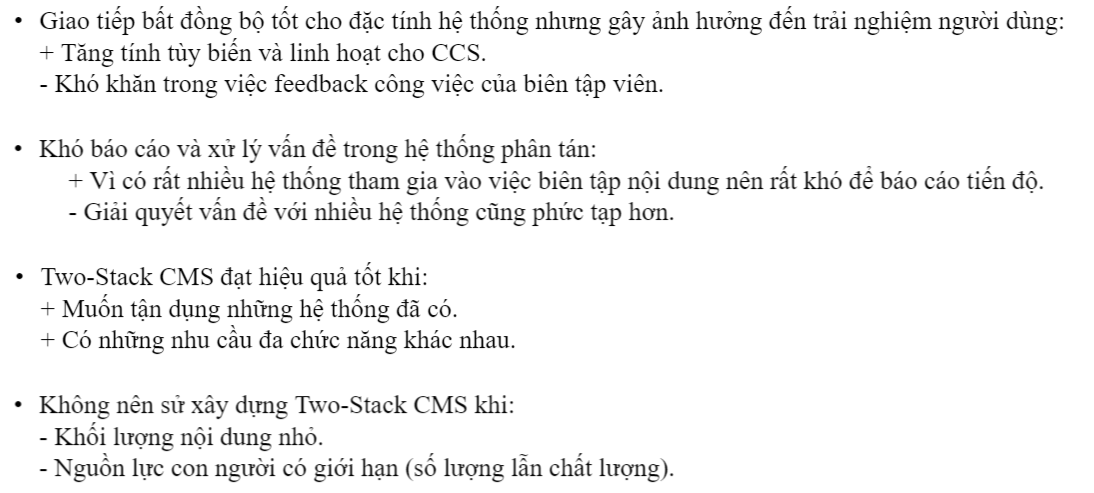
**Custom CMS**

* Vất vả hơn khi phải xây dựng lại từ đầu.
* Xử lý những bài toán từ đơn giản đến phức tạp, theo mọi quy trình, yêu cầu cụ thể.

**Software as a Service CMS**

* Được build sẵn và đóng gói bởi bên cung cấp.
* Được hỗ trợ vận hành, sửa lỗi, nâng cấp
* Đòi hỏi phí bản quyền.

## Câu 3 : Kết luận



# The LMAX Architectures

## Câu 1 : The LMAX Architecture là gì ?

* **TLA** là một nền tảng giao dịch tài chính bán lẻ mới, với khả năng xử lý nhiều giao dịch với độ trễ thấp.
* TLA gồm 3 phần chính :

+ Bộ xử lý logic

+ Bộ ngắt đầu vào

+ Bộ ngắt đầu ra

## Câu 2 : Ưu điểm và nhược điểm TLA :

* **Ưu điểm** : Hoạt động hoàn toàn trong bộ nhớ, không sử dụng cơ sở dữ liệu hoặc kho lưu trữ nào khác nên:
* Nhanh vì không tốn thời gian truy cập, lấy và ghi dữ liệu vào cơ sở dữ liệu.
* Đơn giản hóa việc lập trình vì không cần thiết lập object maps / quan hệ với cơ sở dữ liệu.
  + Tốc độ xử lý có thể đạt 10K -> 100K TPS
* **Nhược điểm :**
* Vì xử lý tuần tự nên hệ thống có thể bị treo (kẹt). Để giải quyết vấn đề này, có thể xử lý bằng việc đẩy ra một output event và đợi một input event khác để tiếp tục gọi lại (VD)
* Khi có lỗi xảy ra, hệ thống không có cơ chế tự động hoàn trả như CSDL. Do đó phải đảm bảo tuyệt đối sự kiện đầu vào phải hợp lệ trước khi lưu trữ vào bộ nhớ.

## Câu 3 : LMAX Disruptor là gì ?

* LMAX-Disruptor là một bộ thư viện giúp cho việc phát triển các ứng dụng với độ tải lớn (high-performance) cho phép xử lý đồng thời (concurrency) một số lượng rất lớn message mà không cần Lock (lock-free).
* Nếu bạn làm việc với Java thì thực tế đây là bộ thư viện về Concurrency tốt nhất và nhanh nhất hiện nay.
* Các thành phần cơ bản :

+ Data(Dữ liệu cần xử lý)

+ Ringbuffer (Hàng đợi dạng vòng chứa dữ liệu)

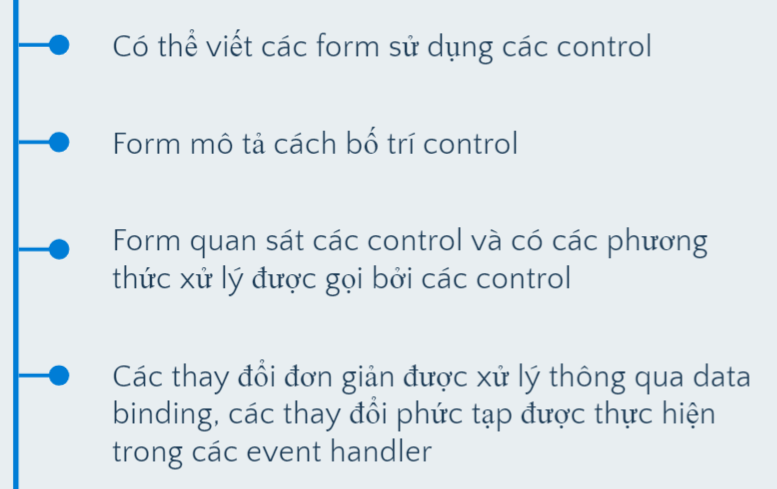
+ Producers (Đưa dữ liệu vào vòng)

+ Consumers (Đọc dữ liệu từ vòng)

# GUI Architectures

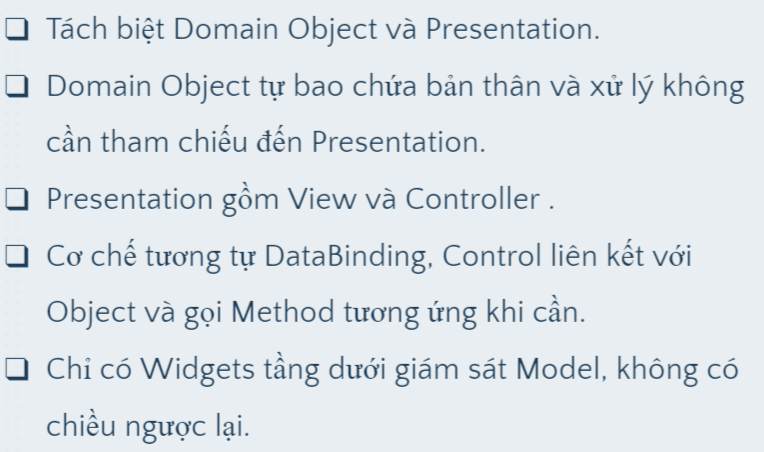
**(Kiến trúc giao diện người dùng)**

## Câu 1 : Form và Control

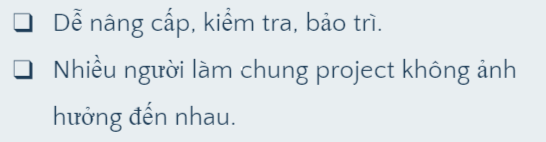


## Câu 2 : Mô hình Model – View – Controller

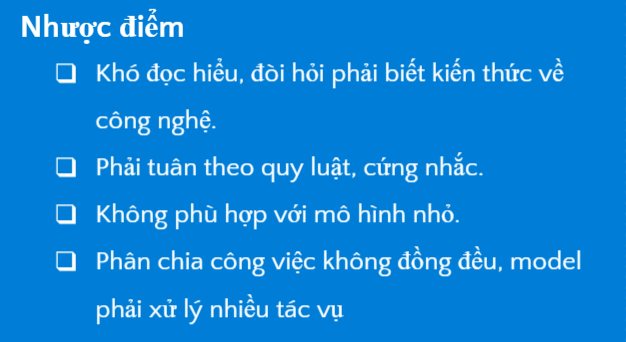
* **Khái niệm:**



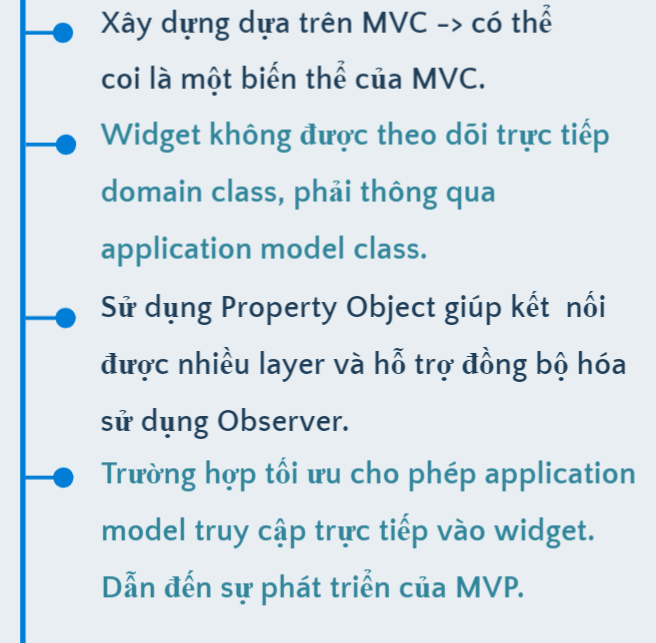
* **Ưu điểm :**



* **Nhược điểm :**



## Câu 3 : Mô hình VisualWorks Application Model :



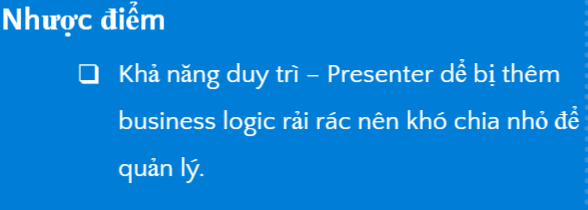
## Câu 4 : Mô hình Model – View – Presenter (MWP)

* Xuất hiện đầu tiên vào năm 1990
* Là sự kết hợp giữa mô hình Forms-Controls và mô hình MVC
* Mô hình dựa trên nghiên cứu của Martin Fowler 2006
* **Các thành phần**
* **Model** chứa dữ liệu và các tính toán xử lý logic để giải quyết vấn đều mà phần mềm hướng tới.
* **View** tương tự như các Controls của Form nhưng không chứa bất kì việc xử lí logic nghiệp vụ nào. Chúng là thành phần tương tác trực tiếp với người dùng.
* **Presenter** đảm nhiệm vai trò trình bày, xử lý thao tác người dùng. Các thao tác xử lý cơ bản vẫn nằm trong Widgets nhưng do Presenter điều khiển.

**Câu 5 : So sánh Form and control, MVC, Application Model**



## Câu 6 : Ưu điểm, nhược điểm MWP

# CQRS

## Câu 1 : CQRS là gì ?

* Command and Query Responsibility Segregation (CQRS) là pattern chia tách việc ghi và đọc dữ liệu.

## Câu 2 : Vai trò của CQRS :

+ Tối ưu hóa hiệu suất

+ Tăng khả năng mở rộng

+ Tăng tính bảo mật

+ Ngăn chặn các lệnh gây ra xung đột

## Câu 3 : Ưu điểm và nhược điểm CQRS

* **Ưu điểm:**

+ Mở rộng quy mô độc lập

+ Bảo mật và bảo trì

+ Lược đồ dữ liệu được tối ưu hóa

+ Truy vấn đơn giản hơn

* **Nhược điểm:**

+ Tính phức tạp

+ Tính nhất quán

+ Truyền thông tin

# Ôn tập các câu hỏi của phương pháp mô hình hóa

## Mô hình hóa là gì? Ý nghĩa của mô hình hóa?

**Định nghĩa:**

* Mô hình hoá là **hướng nền tảng** và **định lượng** để **hiểu các hệ thống phức tạp** và các hiện tượng (tự nhiên, trong cuộc sống…).
* Mô hình hoá là sự **bổ sung** tới các phương pháp nghiên cứu cổ điển về **lý thuyết và thực nghiệm**.
* Mô hình hoá là quá trình sản sinh ra các **mô hình trừu tượng và khái niệm**.

**Ý nghĩa của mô hình hóa:**

* Tất cả các **phương pháp, kỹ thuật và lý thuyết** đều có thể được mô hình hoá các dạng khác nhau.
* Bản thân lý thuyết mô hình hoá được tạo bởi **triết học, lý thuyết hệ thống** và cả những **sự mường tượng** thông minh.
* Mô hình hoá là sự cần thiết và không thể tách rời trong hoạt động khoa học.
* Mô hình hoá sẽ tạo các kỹ năng và kỹ thuật nhằm tạo ra các **kết quả** sâu sắc, đáng tin cậy, và dễ sử dụng (đây cũng là một nguyên nhân và mục đích của mô hình hoá).
* Mô hình hoá cần phải được xác nhận nếu như mô hình **phản ánh thực tế**, và nhận dạng sự thoả thuận khác biệt giữa **lý thuyết và dữ liệu**.

## Mô hình hướng dịch vụ (SOA) là gì? Kiến trúc và lợi ích của SOA?

***SOA - Service Oriented Architecture.***

**Định nghĩa:**

* Là **một cách tiếp cận** để **tổ chức các tài nguyên công nghệ thông tin** mà ở đó dữ liệu, logic và nguồn lực hạ tầng được truy cập thông qua các **giao diện** (interface) và **thông điệp** (Message).
* SOA là **cấp độ cao hơn của sự phát triển ứng dụng**, chú trọng đến quy trình nghiệp vụ và dùng **giao diện chuẩn** để **che dấu sự phức tạp** kỹ thuật bên dưới

**Kiến trúc của SOA:**

****

**Lợi ích của SOA**:

* Hướng sự tập trung vào xây dựng các **tính năng** nghiệp vụ (trong quá trình phát triển phần mềm).
* Tính **tái sử dụng** cao, không đòi hỏi phải thay đổi toàn bộ quy trình
* Giảm thiểu **yêu cầu về đào tạo và kỹ năng**
* Giảm thiểu **chi phí**, chi phí bảo trì thấp
* Chu trình phát triển phần mềm **nhanh chóng** hơn
* Các bên không cần khai báo **tường minh** về công nghệ, nền tảng hệ thống, vị trí hoặc môi trường

## Web service (WS) là gì? Các thành phần của WS?

**Định nghĩa:**

* Web service là một công nghệ triệu gọi từ xa có tính khả chuyển cao: mang tính độc lập nền, độc lập ngôn ngữ.
* Với web service, các chương trình viết bằng các ngôn ngữ lập trình khác nhau, chạy trên các nền tảng (phần cứng & OS) khác nhau đều có thể trao đổi với nhau
* Sử dụng công nghệ được xây dựng trên nền tảng Web
* Như là những hệ thống phần mềm được thiết kế nhằm hỗ trợ tính tương tác liên thông trên hệ thống mạng

**Thành phần:**

* Nhóm các chuẩn dựa trên XML
* Ngôn ngữ định danh dịch vụ Web (Web Services Definition Language – WSDL) có nhiệm vụ cung cấp một cách thức cơ bản để mô tả các thành phần của phần mềm
* Giao thức truy cập đối tượng đơn giản (Simple Object Access Protocol – SOAP)
* Chuẩn dịch vụ Tích hợp, Khám phá và Mô tả thống nhất (Universal Description, Discovery, and Integration – UDDI)

Ngoài ra trong kiến trúc WS còn các thành phần:

Web services provider

* Chứa các WS và hiện thực các business logic
* Quản lý kiểm soát các truy cập đế. Vd như Microsoft, Amazon, Facebook...

Web services requestor

* Yêu cầu một dịch vụ cụ thể
* Tìm và gọi các dịch vụ. Vd: client, server hoặc một dịch vụ khác.

Web services registry

* Nơi hỗ trợ tra cứu các dịch vụ cùng các mô tả liên quan về dịch vụ
* Vd: UDDI registry

## SDLC là gì? Các giai đoạn trong SDLC? Trình bày mục đích của từng giai đoạn?

**Khái niệm:**

* SDLC viết tắt của System Development Life Cycle - Vòng đời phát triển hệ thống là một quy trình được sử dụng bởi ngành công nghiệp phần mềm để thiết kế.
* Với mục đích: sản xuất một phần mềm chất lượng cao, đáp ứng hoặc vượt quá mong đợi của khách hàng, hoàn thành trong thời gian và ước tính chi phí
* SDLC là phương pháp hệ thống giải quyết vấn đề nghiệp vụ

**Các giai đoạn trong SDLC - mục đích của từng giai đoạn:**

**7 giai đoạn**

**Giai đoạn 1: Lập kế hoạch**

**Mục đích:** Xác định phạm vi, yêu cầu liên quan đến việc nghiên cứu hệ thống hiện có và thu thập thông tin chi tiết để tìm hiểu các yêu cầu là gì, cách thức hoạt động và nơi cần cải thiện,

**Giai đoạn 2: Xác định yêu cầu**

**Mục đích:** Người sử dụng cần gì để thực hiện công việc đó (phỏng vấn, tập hợp hệ thống, quản lý hệ thống, sử dụng câu hỏi, form…)

Cuối cùng có được tài liệu SRS (Đặc tả yêu cầu phần mềm-Software Requirements Specification) bao gồm tất cả các yêu cầu sản phẩm được thiết kế và phát triển trong vòng đời dự án

**Giai đoạn 3: Phân tích**

**Mục đích:** Mô hình hệ thống mà tổng quát tìm kiếm cái gì, cung cấp giá thành/lợi ích trong quá trình chọn lựa và đưa ra những đề nghị sẽ làm gì trong mô hình (tạo sơ đồ dòng dữ liệu, dẫn chứng tài liệu lập luận logic…)

**Giai đoạn 4: Thiết kế**

**Mục đích:** Thiết kế bên trong của tất cả các mô-đun của hệ thống (Design Document

Specification-DDS) như thiết kế giao diện sử dụng, thiết kế kiểm tra CSDL, thiết kế file, CSDL…

**Giai đoạn 5: Xây dựng hệ thống**

**Mục đích:** Chọn ngôn ngữ và công nghệ liên quan đến loại phần mềm, viết mã cho các thiết kế DDS, phải tuân theo các nguyên tắc mã hóa được xác định bởi các công cụ tổ chức và lập trình của họ như trình biên dịch, trình thông dịch, trình gỡ lỗi,...

**Giai đoạn 6: Thử nghiệm và bảo trì hệ thống:**

**Mục đích:** Tiến hành thử nghiệm bởi các đội lập trình viên, phân tích hệ thống, đội kiểm tra chất lượng, thử nghiệm trên dữ liệu. Bảo trì hệ thống khi chưa đáp yêu cầu. Thử nghiệm và bảo trì đến khi sản phẩm đạt chất lượng được xác định trong SRS.

**Giai đoạn 7: Vận hành và đánh giá hệ thống**

**Mục đích:** Đặt kế hoạch chuyển đổi, huấn luyện người sử dụng, cài đặt và đưa ra đánh giá cho hệ thống.

## Có những cách tiếp cận nào trong hiện thực SDLC? Nêu ưu/nhược điểm của từng cách và cho ví dụ về phương pháp trong từng cách tiếp cận.

**2 Cách tiếp cận trong thực hiện SDLC:**

**Cách tiếp cận tuyến tính:** các bước được thực hiện theo trình tự và được hoàn thành trước khi chuyển sang.

**Cách tiếp cận tiến hóa:** giải pháp được phân phối xuất hiện trong các giai đoạn phát triển phần mềm lặp lại

* **Cách tiếp cận tuyến tính:**

**Ưu điểm:**

* Mỗi bước hoàn thành ở một thời điểm nhất định.
* Phù hợp với những dự án nhỏ, có yêu cầu xác định rõ ràng
* Các bước (pha) biệt lập, rõ ràng
* Dễ dàng sắp xếp kế hoạch và công việc
* Quy trình và kết quả được văn bản hóa rõ ràng.
* Dễ hiểu và có thể kiểm tra đánh giá từng bước.

**Nhược điểm:**

* Chỉ có thể tiếp cận sản phẩm ở giai đoạn cuối.
* Tính rủi ro và bất định cao
* Khó xử lý đồng thời các sự kiện
* Khó điều chỉnh thay đổi với sự yêu cầu
* Không phù hợp với những hệ thống phức tạp và phương pháp hướng đối tượng

**VD: mô hình thác nước và mô hình V...**

* **Cách tiếp cận tiến hóa:**

**Ưu điểm:**

* Một số chức năng được thực hiện một cách nhanh chóng
* Có thể thực hiện song song các chức năng
* Dễ dàng kiểm tra, gỡ lỗi
* Quá trình thực hiện có thể đo lường hiệu quả dễ dàng
* Phân tích, phát hiện và quản lý các rủi ro dễ dàng
* Phù hợp với những dự án lớn và cấp bách
* Tận dụng được phản hồi của khách hàng trong suốt quá trình phát triển
* Người dùng có thể thấy được “hình hài” của sản phẩm sớm

**Nhược điểm:**

* Công tác quản lý và quy trình khá phức tạp
* Cần nhiều tài nguyên
* Phác thảo kiến trúc và thiết kế kiến trúc liên tục
* Cần nhiều kinh nghiệm khi tiến hành
* Cần nhiều công sức hơn cho quản lý sự phức tạp của hệ thống
* Kết thúc dự án có thể chưa phát hiện được các rủi ro tiềm ẩn

**VD: mô hình lặp, xoắn ốc…**

## SDLC - Vòng đời mô hình hóa – Software Development Life Cycle

1. **Các mô hình: Waterfall, mô hình chữ V, lặp, xoắn ốc, RAD, Agile**

### - Mô hình Waterfall:

+ 1 giai đoạn chỉ được bắt đầu khi **giai đoạn trước nó đã kết thúc và xác nhận**

+ Trong quá trình hoạt động nếu có vấn đề thì phải lập tức **review**

+ Dự án có thể **kéo dài rất lâu** vì các giai đoạn có ảnh hưởng liên đới và phụ thuộc nhau

+ Không thích hợp với sự thay đổi thường xuyên

*+ Trường hợp sử dụng:* y/c rõ rang, ko thay đổi. C/nghệ và sp cố định. Các th/tin, t/nguyên đc cg cấp sắn. d.a nhỏ

*+ Ưu điểm:*đ/giản và dễ a/dụng.Mỗi pha h/thành ở 1 t/điểm nh/định. P/hợp vs d.a nhỏ và y/c rõ rg. Các pha biệt lập, rõ rg. Các cột mốc t/g đc ph/định rõ, ko trùng lắp. Dễ s/xếp k/hoạch và c/việc. Q/trình và kq đc văn bản hóa dễ dàng.

*+ Khuyết điểm:* chỉ t/cận sp ở g/đ cuối. Rủi ro và bất định cao. Ko p/hợp với h/thống ph/tạp và pp hg đ/tượng. Khó điều tiết, hiệu chỉnh với sự th/đổi y/c. Hiệu chỉnh phạm vi có thể dẫn đến k/thúc dự án.

### - Mô hình chữ V:

+Tương tự như m/hình thác nước nhg mỗi g/đ đều k/hợp với h/đg kiểm tra.

+Cạnh bên trái : Liên quan đến phát triển sản phẩm

+Cạnh bên phải: Liên quan đến kiểm thử đảm bảo chất lượng

+Ý nghĩa chấm nối hai cạnh: Thể hiện 2 việc làm *song song*

### - Mô hình Lặp:

+ Hướng tiếp cận nhanh để nhanh tạo ra sản phẩm => nhanh chỉnh sửa => phát triển lặp

Cho khách hàng sử dụng bản prototype => **lặp và cải tiến**

+ Tiến hành cùng lúc 3 module trước => Xem cái nào khách hàng oke thì check, chưa thì tiến hành lặp lại, mở rộng thì (+) bổ sung thêm

### - Mô hình Xoắn ốc:

+ Kết hợp **tính lặp** của mô hình lặp và tính **hệ thống, dễ kiểm soát** của mô hình thác nước

+ Phát hành các **sản phẩm tăng dần** một cách tuần tự theo các bước lặp

+ Còn với mô hình lặp sẽ xđ các chức năng chính r thực hiện song các bước, còn xoắn ốc thì tuần tự theo các bước lặp

**+ Ý kiến phản hồi** từ khách hàng/ người dùng là cơ sở tiến hành các bước lặp tiếp theo

+Để cải tiến, bổ sung hoàn chỉnh từ các yêu cầu của khách hàng

+Là mô hình được **sử dụng rộng rãi** trong phát triển phần mềm

* bước được lặp đi lặp lại một cách thường xuyên:
* Identifications: Xác định
* Design; Thiết kế
* Construct or Build: Xây dựng hoặc cấu tạo hệ thống dựa trên kết quả của việc thiết kế
* Evaluation and Risk Analysis: Đánh giá và phân tích rủi ro.

### - Mô hình RAD:

+ Mô hình RAD (Phát triển ứng dụng nhanh) dựa trên nguyên mẫu và phát triển lặp mà không có kế hoạch cụ thể liên quan

+ Tập trung vào việc thu thập các yêu cầu của khách hàng thông qua các hội thảo hoặc các nhóm tập trung

+ Thử nghiệm sớm các nguyên mẫu của khách hàng bằng cách sử dụng khái niệm lặp, tái sử dụng các nguyên mẫu (thành phần) hiện có, tích hợp liên tục và giao hàng nhanh

+ Là một phương pháp phát triển phần mềm đề cao việc **ít lập kế hoạch và tạo mẫu nhanh**

+ Các **mô đun** chức năng được **phát triển song song** dưới dạng nguyên mẫu và được tích hợp để tạo ra sản phẩm hoàn chỉnh để phân phối sản phẩm nhanh hơn

+ Vì không có kế hoạch trước chi tiết, nên việc kết hợp các thay đổi trong quá trình phát triển sẽ **linh hoạt** hơn

+ Các nhóm nhỏ bao gồm các nhà phát triển, chuyên gia hệ thống, đại diện khách hàng và các tài nguyên CNTT khác làm việc liên tục trên thành phần hoặc nguyên mẫu

+ Đảm bảo rằng các nguyên mẫu được phát triển có thể **tái sử dụng**

**Ưu:**+Thích nghi với việc thay đổi yêu cầu

+ Tiến độ có thể được đo lường

+Thời gian lặp thể ngắn

+Hiệu quả trong khi cần ít nhân lực trong thời gian ngắn

+ Giảm thời gian phát triển

+Tăng khả năng tái sử dụng của các thành phần

+Đánh giá ban đầu được diễn ra nhanh chóng

+Khuyến khích phản hồi của khách hàng

+Giải quyết rất nhiều vấn đề tích hợp

**Nhược điểm**

+ Phụ thuộc vào đội ngũ mạnh về kỹ thuật xác định yêu cầu

+ Chỉ dùng cho hệ thống có thể được mô đun hóa

+ Yêu cầu các nhà phát triển / thiết kế có tay nghề cao

  + Nếu không tốt => Để lại ấn tượng xấu cho khách hàng trong những bản mẫu đầu tiên

+ Phụ thuộc cao vào kỹ năng xây dựng mẫu

**+ Không thể áp dụng** cho các dự án có **chi phí thấp**

+ Quản lý phức tạp hơn

    + Thích hợp cho các hệ thống dựa trên thành phần và có thể mở rộng

+ Yêu cầu sự tham gia xuyên suốt của người dùng

**+ Thích hợp** cho dự án đòi hỏi **thời gian phát triển ngắn** hơn

### - Mô hình Agile:

+ Kết hợp giữa tính **lặp** và **tăng dần**

+Tập trung vào khả năng **thích ứng quy trình** và sự **hài lòng của khách hàng** bằng + +Cách cung cấp nhanh chóng mẫu sản phẩm

+Chia sản phẩm thành các **bản dựng** nhỏ tăng dần

+Mỗi lần lặp thường kéo dài từ khoảng 1-3 tuần

  +Khác với mô hình xoắn ốc :Thời gian lặp k xác định, k cố định

+Mỗi lần lặp lại liên quan đến các nhóm chức năng thuộc các pha khác nhau như: Planning, Requirements, Analysis, Design, Coding, Unit Testing, Acceptance Testing

**+ Các tuyên ngôn của Agile(thiên hướng & trọng tâm):**

**Các cá nhân và tương tác:** Trong phát triển Agile, tự tổ chức và tạo động lực rất quan trọng, cũng như các tương tác như cùng vị trí hay lập trình cặp

**Phần mềm làm việc:** Phần mềm hỗ trợ làm việc được coi là phương tiện giao tiếp tốt nhất với khách hàng để hiểu các yêu cầu của họ, thay vì chỉ phụ thuộc vào tài liệu

**Hợp tác với khách hàng:** Tương tác khách hàng liên tục là rất quan trọng để có được yêu cầu sản phẩm phù hợp

**Ứng phó với thay đổi:** Phát triển Agile tập trung vào các phản ứng nhanh với thay đổi và phát triển liên tục

**Ưu điểm:**

* Tiếp cận rất thực tế để phát triển phần mềm
* Thúc đẩy tinh thần đồng đội và đào tạo chéo
* Chức năng có thể được phát triển nhanh chóng
* Yêu cầu tài nguyên là *tối thiểu*
* Thích hợp cho các yêu cầu cố định hoặc thay đổi
* Hỗ trợ xây dựng giải pháp cục bộ
* Thích ứng tốt với môi trường thay đổi liên tục
* Sử dụng ít các quy định, tài liệu
* Cho phép phát triển đồng thời và phân phối trong bối cảnh kế hoạch tổng thể
* Ít hoặc không cần lập kế hoạch
* Dễ quản lý
* Cung cấp sự linh hoạt cho người lập trình

**Nhược:**

* Không phù hợp để xử lý các phụ thuộc phức tạp
* Nhiều rủi ro về tính bền vững, khả năng duy trì và mở rộng
* Cần có một kế hoạch tổng thể, một nhà lãnh đạo nhanh nhẹn và kinh nghiệm
* **Quản lý phân phối nghiêm ngặt** chỉ ra phạm vi, **chức năng sẽ được phân phối** và điều chỉnh để đáp ứng thời hạn
* Phụ thuộc nhiều vào tương tác của khách hàng
* Sự phụ thuộc cá nhân rất cao
* Chuyển giao công nghệ cho các thành viên nhóm mới có thể khá khó khăn do thiếu tài liệu

1. **Agile là gì? Các nguyên tắc chính của Agile?**

**Khái niệm:**

**Agile** là một phương pháp phát triển phần mềm linh hoạt, là một hướng tiếp cận cụ thể cho việc quản lý dự án phần mềm. Nó gồm một quá trình làm việc tương tác và tích hợp để có thể đưa sản phẩm đến tay người dùng càng nhanh càng tốt.

**Các nguyên tắc (tuyên bố) của Agile:**

* Customer Satisfaction
* Welcome Change
* Deliver a Working Software
* Collaboration
* Motivation
* Face-to-face Conversation
* Measure the Progress as per the Working Software
* Maintain Constant Pace
* Monitoring
* Simplicity
* Self-organized Teams
* Review the Work Regularly

1. **So sánh Agile và Waterfall**

- Agile: Tách v/đời p/triển d.a thành chạy nước rút. Theo 1 cách t/cận gia tăng. Pp nhanh đc biết đến với tính l/hoạt của nó. Agile c/thể đc coi là 1 bst của nhiều d.a kh/nhau. Agile là một pp khá l/hoạt cho phép th/đổi đc th/hiện trg các y/c p/triển d.a ngay cả khi k/hoạch bđ đã đc h/thành. Pp nhanh, theo 1 cách tiếp cận ph/triển lặp lại vì quy hoạch, ph/triển, tạo mẫu và các g/đ ph/triển phần mềm khác có thể xuất hiện nhiều lần. Kế hoạch kiểm tra được xem xét sau mỗi lần chạy nước rút. Phát triển nhanh là một quá trình trong đó các yêu cầu được dự kiến ​​sẽ thay đổi và phát triển. Trong phương pháp Agile, thử nghiệm được thực hiện đồng thời với phát triển phần mềm. Agile giới thiệu tư duy sản phẩm, nơi sản phẩm phần mềm đáp ứng nhu cầu của khách hàng cuối cùng và thay đổi chính nó theo nhu cầu của khách hàng. Agat methodology hoạt động đặc biệt tốt với Time & Materials hoặc tài trợ không cố định. Nó có thể làm tăng căng thẳng trong các kịch bản giá cố định. Thích các nhóm nhỏ nhưng chuyên dụng với mức độ phối hợp và đồng bộ hóa cao.  Chủ sở hữu sản phẩm với nhóm chuẩn bị các yêu cầu chỉ là về mỗi ngày trong một dự án. Đội kiểm tra có thể tham gia vào các yêu cầu thay đổi mà không có vấn đề gì. Mô tả chi tiết dự án có thể được thay đổi bất cứ lúc nào trong quá trình SDLC. Các thành viên của Nhóm Agile có thể hoán đổi cho nhau, do đó, chúng hoạt động nhanh hơn. Cũng không cần thiết cho các nhà quản lý dự án vì các dự án được quản lý bởi toàn bộ nhóm

- Waterfall: Quá trình phát triển phần mềm được chia thành các giai đoạn riêng biệt. Phương pháp thác nước là một quá trình thiết kế tuần tự. Thác là một phương pháp phát triểphần mềm có cấu trúc nên hầu hết thời gian nó có thể khá cứng nhắc. Phát triển phần mềm sẽ được hoàn thành như một dự án duy nhất. Không có phạm vi thay đổi các yêu cầu khi phát triển dự án bắt đầu. Tất cả các giai đoạn phát triển dự án như thiết kế, phát triển, thử nghiệm, vv được hoàn thành một lần trong mô hình Thác. Kế hoạch kiểm tra hiếm khi được thảo luận trong giai đoạn thử nghiệm. Phương pháp này là lý tưởng cho các dự án có yêu cầu nhất định và thay đổi không được mong đợi. Trong phương pháp này, giai đoạn "Thử nghiệm" xuất hiện sau giai đoạn "Xây dựng". Mô hình này cho thấy một tư duy dự án và đặt trọng tâm của nó hoàn toàn vào việc hoàn thành dự án. Giảm rủi ro trong các hợp đồng giá cố định của công ty bằng cách nhận được thỏa thuận rủi ro vào đầu quá trình. Phối hợp / đồng bộ hóa nhóm rất hạn chế. Phân tích kinh doanh chuẩn bị các yêu cầu trước khi bắt đầu dự án. Thật khó để thử nghiệm bắt đầu bất kỳ thay đổi nào về yêu cầu. Mô tả chi tiết cần thực hiện phương pháp tiếp cận phát triển phần mềm thác nước. Trong phương pháp thác nước, quy trình luôn đơn giản như vậy, người quản lý dự án đóng một vai trò thiết yếu trong mọi giai đoạn của SDLC.

1. **Scrum và Extreme Programming: là gì? so sánh 2 phương pháp**

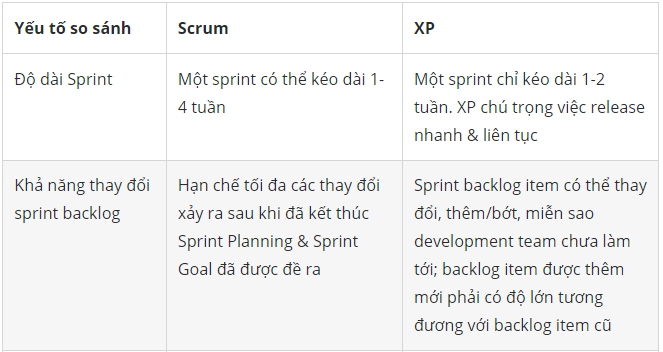
**Scrum là gì?**

**Scrum** là một “bộ khung làm việc” cơ bản để tiếp cận những công việc phức tạp. Dựa trên bộ khung này, nhóm làm việc có thể áp dụng những quy trình, kỹ thuật khác nhau cho công việc của mình… Nó là một thành viên của họ [Agile](https://topdev.vn/blog/agile-la-gi/).

**Extreme Programming là ?**

**XP (Extreme Programming)** là một phương pháp phát triển phần mềm hướng đến việc nâng cao chất lượng phần mềm và khả năng đáp ứng với thay đổi yêu cầu người dùng.

**So sánh 2 phương pháp:**



## 7. Thiết kế hướng cấu trúc (SOD) và thiết kế hướng đối tượng (OOD) là gì? So sánh 2 phương pháp.

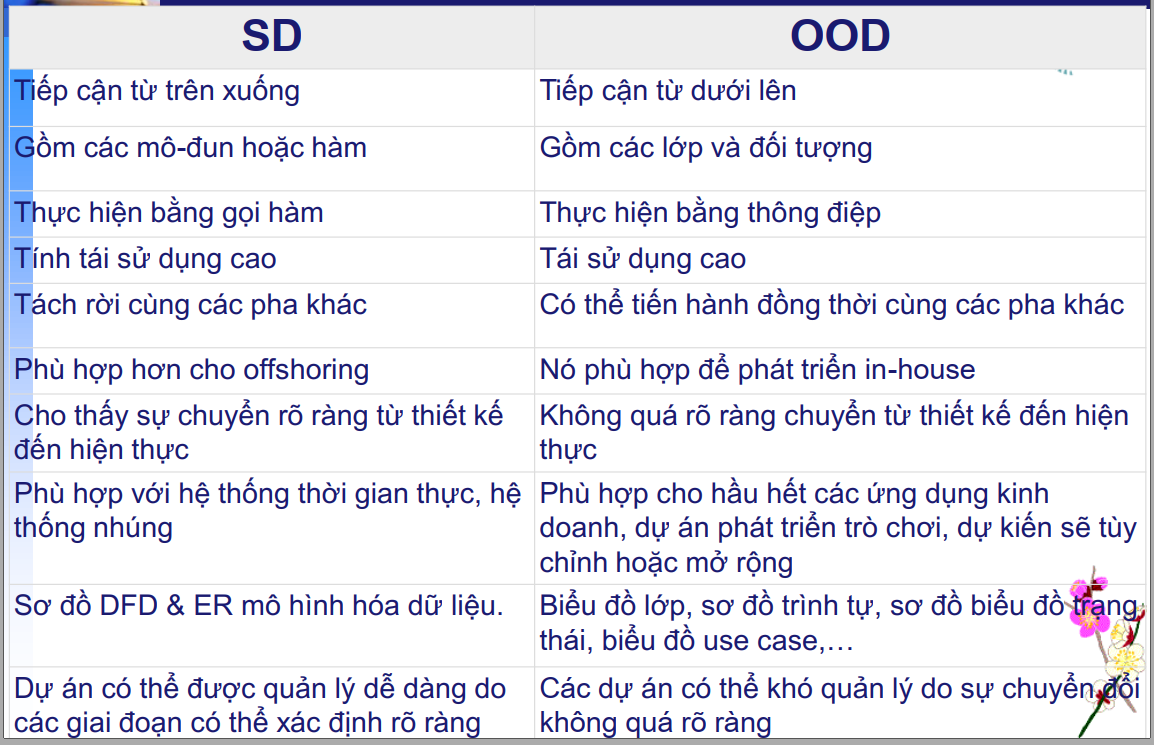
**SOD** là một phương pháp dựa trên luồng dữ liệu giúp xác định đầu vào và đầu ra của hệ thống đang phát triển.

Mục tiêu chính là để giảm thiểu sự phức tạp và tăng tính mô-đun của một chương trình bằng cách thiết kế một hệ thống mô-đun top-down

**ODD** Nắm bắt cấu trúc và hành vi của các hệ thống thông tin thành các mô-đun nhỏ kết hợp cả dữ liệu và quy trình

Mục đích chính là cải thiện chất lượng và năng suất của phân tích và thiết kế hệ thống bằng cách làm cho nó trở nên tiện dụng hơn, mô phỏng hệ thống tốt hơn

**So sánh:**



## 8. ERD (mô hình mối quan hệ thực thể) là gì? Mục đích? Thành phần? Các bước xây dựng?

**Khái niệm:**

Mô hình ERD là viết tắt của **Entity Relationship Diagram** được hiểu là mô hình thực thể kết hợp hay còn gọi là thực thể liên kết.

**Mục đích:**

Mục tiêu của mô hình E-R trong quá trình thiết kế cơ sở dữ liệu đó là phân tích dữ liệu, xác định các đơn vị thông tin cơ bản cần thiết của tổ chức, mô tả cấu trúc và mối liên hệ giữa chúng

E-R là mô hình trung gian để chuyển những yêu cầu quản lý dữ liệu trong thế giới thực thành mô hình cơ sở dữ liệu quan hệ

**Thành phần:**

* **Entity** (thực thể) nó chỉ định các mục trong thế giới thực vd: nhà cung cấp, sản phẩm, sinh viên…
* **RELATIONSHIP** (mối kết hợp) là những phụ thuộc có ý nghĩa giữa các thực thể VD: nhà cung cấp vật phẩm, giáo viên dạy các khóa học…
* **ATTRIBUTES (thuộc tính)** Các thông tin của các thực thể/mối kết hợp VD: mã nhà cung cấp, tên sinh viên,…

**Các bước xây dựng:**

* Liệt kê, chọn lọc thông tin
* Xác định tập thực thể
* Xác định mối quan hệ
* Xác định thuộc tính và gán thuộc tính cho tập thực thể và mối quan hệ
* Quyết định thuộc tính khóa
* Vẽ biểu đồ mô hình thực thể E-R
* Chuẩn hóa biểu đồ

**8.1. Dữ liệu quan hệ (là gì? Mục đích? Thành phần? Các bước xây dựng)**

**Khái niệm:** **Mô hình quan hệ** đề xuất năm 1970, được xây dựng dựa trên cấu trúc toán học tự nhiên và đơn giản: Quan hệ (relation) hay Bảng (table).

Mô hình quan hệ mô tả dữ liệu trong bảng dưới dạng Miền, Thuộc tính, Bộ dữ liệu.

**Mục đích:** Tương tự như ERD - tạo nên mối quan hệ giữa các bảng sẽ ít dễ dàng nhìn hơn.

**Thành phần:**

**ATTRIBUTES (thuộc tính)** Là các cột của một quan hệ được đặt tên cụ thể.

Các thuộc tính xuất hiện ở trên cùng của một cột và mô tả ý nghĩa của cột đó. Thứ tự của các cột trong một quan hệ không quan trọng, miễn sao cung cấp đầy đủ thông tin và tránh dư thừa dữ liệu là được.

Đặc trưng: Các mối quan hệ phải có tên riêng biệt để phân biệt với nhau, Các bộ được phân biệt với nhau và không quan trọng thứ tự, Mỗi thuộc tính cũng cần có tên phân biệt và không quan trọng thứ tự

**Các bước xây dựng**: tương tự ERD

**8.2 Chuyển đổi từ ERD sang mô hình dữ liệu quan hệ**

***Bước 1: chuyển mỗi loại thực thể thành một loại quan hệ tương ứng***

* **Chuyển các mối kết hợp 1-1 gom 2 thực thể thành một thực thể**
* **Các mối kết hợp 1-N lấy khóa bên thực thể nhiều chuyển thành khóa ngoại**
* **Các mối quan hệ N-N hình thành một loại quan hệ mới**

***Bước 2: kiểm tra lại dạng chuẩn của các mối quan hệ.***

## 9. UML – Ngôn ngữ mô hình hóa thống nhất – Unified Modeling Language

**Khái niệm:**

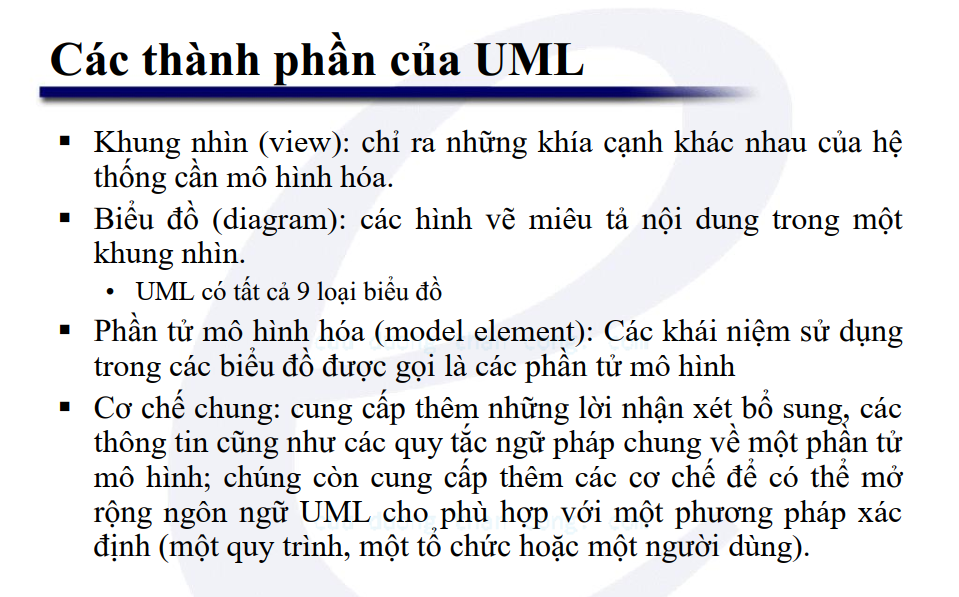
UML là ngôn ngữ trực quan cho phép mô hình hóa các quy trình, phần mềm và hệ thống theo phương pháp hướng đối tượng để thể hiện thiết kế kiến ​​trúc hệ thống

Cho phép các kiến ​​trúc sư kỹ thuật giao tiếp với nhà phát triển dễ dàng

**Lịch sử:**



**Thành phần:** sơ đồ và các đối tượng



**Các sơ đồ trong UML**

1. **Use case**

**Khái niệm:** Use case diagram là một sơ đồ thể hiện cách để những user trong system có thể tương tác với nhau bằng những tính năng gì

**Thành phần:**

**Actor:** Actor được dùng để chỉ người sử dụng hoặc một đối tượng nào đó bên ngoài tương tác với hệ thống chúng ta đang xem xét

**Use case:** Use Case là chức năng mà các Actor sẽ sử dụng.

**Communication link:** Kết nối giữa các Actor và Use case, cho biết những Actor đó có tương tác như thế nào với hệ thống

**Boundary of System:** Phạm vị của sự tương tác. Có thể là trong một hệ thống một module, hoặc 1 tình trạng bất kì

**Relationship:** Mối quan hệ của các use case với nhau

* **include:** Include là quan hệ giữa các Use Case với nhau, nó mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại.
* **extend:** Extend dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.
* **Generalization:** được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau.
* **Association:** thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau.

**2. Sơ đồ tuần tự (sequence diagram)**

**Khái niệm:** dùng để xác định trình tự diễn ra các sự kiện của một nhóm đối tượng nào đó. Nó miêu tả chi tiết các thông điệp gửi và nhận giữa các đối tượng đồng thời cũng chú trọng đến việc trình tự về mặt thời gian gửi và nhận thông điệp đó.

**Thành phần:**

**Object** mô tả một đối tượng trong hệ thống.  Để phân biệt với Class, Object có dấu “:” phía trước tên của nó.

**Đường đời đối tượng (Lifelines)**: biểu diễn bằng các đường gạch rời thẳng đứng bên dưới các đối tượng

**Thông điệp (Message):** biểu diễn bằng các đường mũi tên.

**Xử lý bên trong đối tượng** (biểu diễn bằng các đoạn hình chữ nhật rỗng nối với các đường đời đối tượng).

**3. Sơ đồ hoạt động (Activity diagram)**

**Khái niệm:** Activity Diagram là bản vẽ tập trung vào mô tả các hoạt động, luồng xử lý bên trong hệ thống. Nó có thể được sử dụng để mô tả các quy trình nghiệp vụ trong hệ thống, các luồng của một chức năng hoặc các hoạt động của một đối tượng.

**Thành phần:**

* **Activity:** thể hiện những hành động
* **Action:** thể hiện 1 công việc
* **Control/Object flow:** thể hiện sự chuyển đổi từ hoạt động này sang hoạt động khác
* **Initial node:** thể hiện sự bắt đầu quy trình làm việc trong sơ đồ hoạt động
* **Activity final node:** kết thúc tất cả các luồng điều khiển và luồng đối tượng
* **Object node:** biểu diễn một đối tượng được kết nối với các dòng đối tượng
* **Decision node:** thể hiện điều kiện kiểm tra và luôn có ít nhất 2 luồng trả về
* **Merge node:** Tập hợp lại các quyết định khác nhau đã được tạo từ decision node
* **Fork node:** chia hành vi thành một tập hợp các luồng hoạt động song song đồng thời.
* **Join node:** tập hợp lại các các luồng hoạt động song song hoặc đồng thời
* **Swimlane and Partition(optional):** dùng để nhóm các hoạt động có liên quan với nhau thành một cột (phân luồng dữ liệu)

**4. Sơ đồ lớp (class diagram)**

**Khái niệm:** Class Diagram là một loại sơ đồ cấu trúc tĩnh mô tả cấu trúc của hệ thống bằng cách hiển thị các lớp của hệ thống, thuộc tính, hoạt động của chúng và mối quan hệ giữa các đối tượng. Sơ đồ lớp là khối xây dựng chính của mô hình hướng đối tượng.

**Thành phần:**

**Classes (Các lớp):** Class là thành phần chính của bản vẽ Class Diagram. Class mô tả về một nhóm đối tượng có cùng tính chất, hành động trong hệ thống. Class được mô tả gồm tên Class, thuộc tính và phương thức.

* Class Name: là tên của lớp.
* Attributes (thuộc tính): mô tả tính chất của các đối tượng.
* Method (Phương thức): chỉ các hành động mà đối tượng này có thể thực hiện trong hệ thống. Nó thể hiện hành vi của các đối tượng do lớp này tạo ra.

**Relationship (quan hệ):**

Liên kết (Association)

Kế thừa (Inheritance)

Hiện thực hóa (Realization)

Phụ thuộc (Dependency)

Tổng hợp (Aggregation)

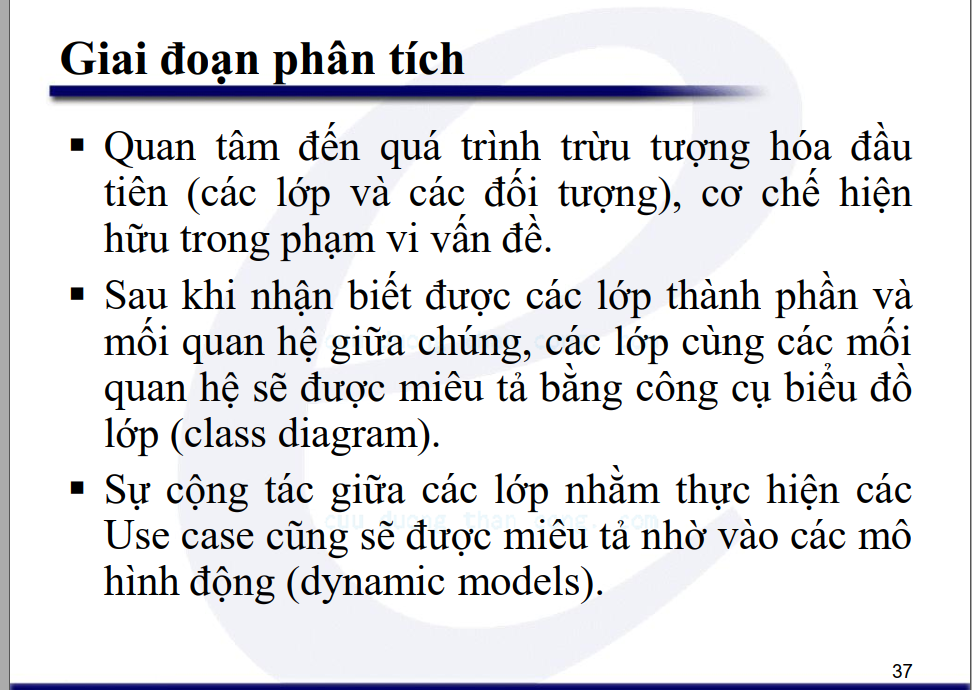
Hợp thành (Composition)

**Các bước trong phân tích và thiết kế hướng đối tượng, nội dung từng bước**

**Phân tích hướng đối tượng:**

* Xác định các yêu cầu, xây dựng mô hình cần sử dụng
* Xác định các lớp và mối quan hệ của chúng với các lớp khác trong phạm vi liên quan

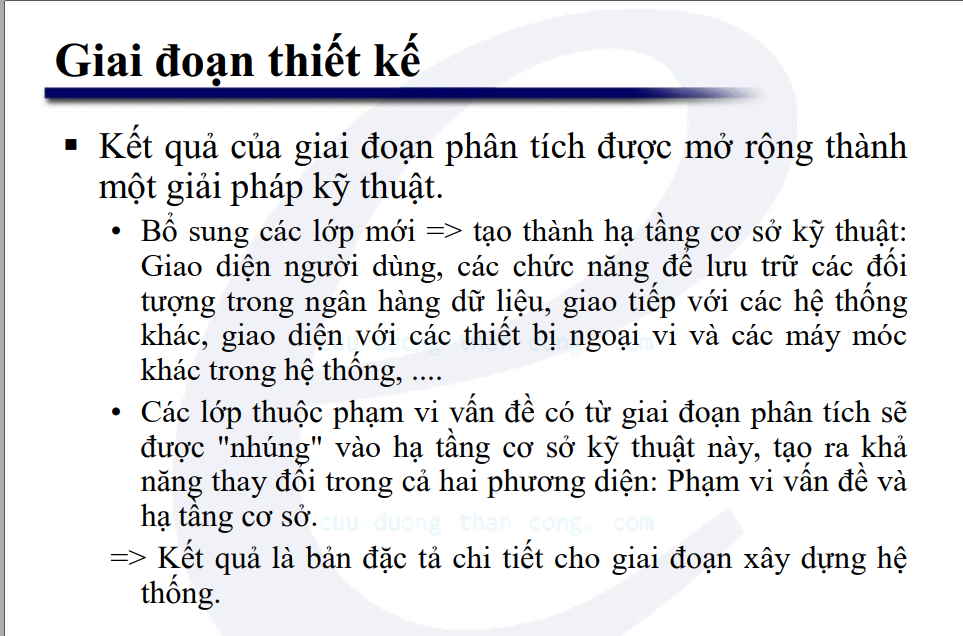
**Nội dung chi tiết:**



**Thiết kế hướng đối tượng:**

* Thiết kế và tinh chỉnh các lớp, thuộc tính, phương thức và cấu trúc, giao diện người dùng và truy cập dữ liệu
* Xác định và định nghĩa các lớp hoặc đối tượng bổ sung hỗ trợ thực hiện yêu cầu

**Nội dung chi tiết:**



## 10. Mô hình tier và 3 layer

**Mô hình 3 tier:**

**Khái niệm:** 3-tiers là một kiến trúc kiểu [client/server](https://topdev.vn/blog/cach-gui-thong-bao-bang-fcm/) mà trong đó giao diện người dùng (UI-user interface), các quy tắc xử lý (BR-business rule hay BL-business logic), và việc lưu trữ dữ liệu được phát triển như những module độc lập, và hầu hết là được duy trì trên các nền tảng độc lập, và mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.” 3-tiers là một kiến trúc kiểu [client/server](https://topdev.vn/blog/cach-gui-thong-bao-bang-fcm/) mà trong đó giao diện người dùng (UI-user interface), các quy tắc xử lý (BR-business rule hay BL-business logic), và việc lưu trữ dữ liệu được phát triển như những module độc lập, và hầu hết là được duy trì trên các nền tảng độc lập, và mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.”

**Thành phần:**

Đây là kiến trúc triển khai ứng dụng ở mức vật lý. Kiến trúc gồm 3 module chính và riêng biệt :

* **Tầng Presentation:** hiển thị các thành phần giao diện để tương tác với người dùng như tiếp nhận thông tin, thông báo lỗi, …
* **Tầng Business Logic**: thực hiện các hành động nghiệp vụ của phần mềm như tính toán, đánh giá tính hợp lệ của thông tin, … Tầng này còn di chuyển, xử lí thông tin giữa 2 tầng trên dưới.
* **Tầng Data:** nơi lưu trữ và trích xuất dữ liệu từ các hệ quản trị CSDL hay các file trong hệ thống. Cho phép tầng Business logic thực hiện các truy vấn dữ liệu .

**Mô hình 3 layer:**

**Khái niệm:** Mô hình 3 lớp hay còn được gọi là mô hình Three Layer(3-Layer), mô hình này ra đời nhằm phân chia các thành phần trong hệ thống, các thành phần cùng chức năng sẽ được nhóm lại với nhau và phân chia công việc cho từng nhóm để dữ liệu không bị chồng chéo và chạy lộn xộn.

**Thành phần:**

* **Graphic User Interface (GUI):** Thành phần giao diện, là các form của chương trình tương tác với người sử dụng.
* **Business Logic Layer (BLL):** Xử lý các nghiệp vụ của chương trình như tính toán, xử lý hợp lệ và toàn vẹn về mặt dữ liệu.
* **Data Access Layer (DAL):** Tầng giao tiếp với các hệ quản trị CSDL

**So sánh điểm khác nhau:**

* Không như 3-Tiers có tính vật lý, 3-Layers có tính logic (mỗi layer có 1 công việc) và là 1 thành phần của 3-Tiers
* Khi dùng từ layer, chúng ta nói tới việc phân chia ứng dụng thành các thành phần một cách logic theo chức năng hoặc theo vai trò, điều này giúp phần mềm của bạn có cấu trúc sáng sủa, dễ dùng lại, từ đó giúp việc phát triển và bảo trì dễ dàng hơn. Các layer khác nhau khi được thực thi vẫn có thể nằm trong cùng một vùng bộ nhớ của một process, và hiển nhiên việc giao tiếp giữa 2 layer có thể không phải là giao tiếp giữa 2 process, đồng nghĩa với việc chúng không liên quan tới mô hình client/server.
* Trái lại, tier liên quan đến cách phân chia một cách vật lý các thành phần trên các máy tính khác nhau. Điều làm nhiều người nhầm lẫn giữa layer và tier là chúng có cùng cách phân chia (presentation, business, data), tuy nhiên trên thực tế chúng khác nhau. Vì cách phân chia như trên nền 1 tier có thể chứa nhiều hơn 1 layer.