
Phân Tích và Thiết Kế Hướng Đối Tượng Sử dụng UML

cuu duong than cong . com

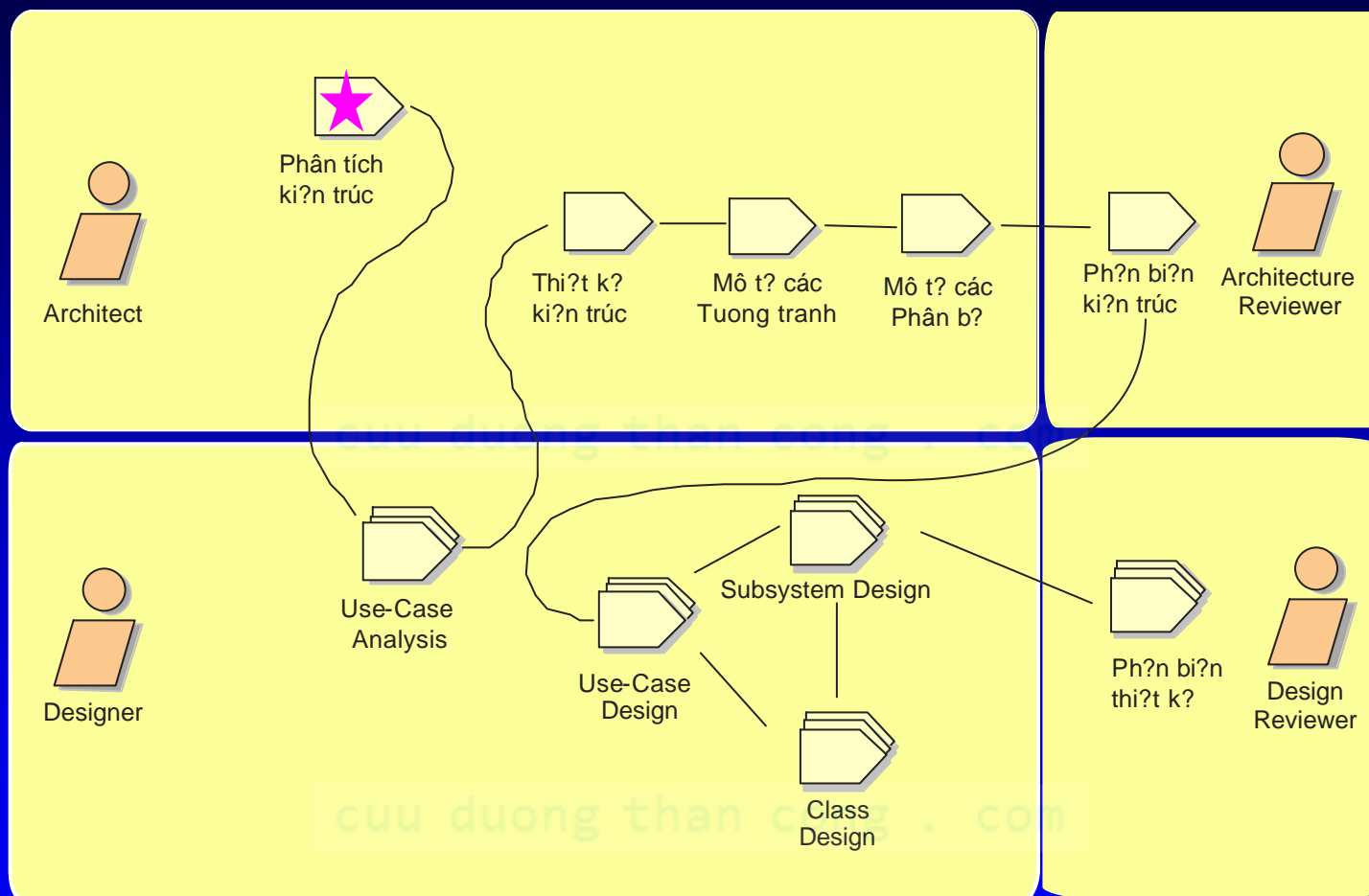
Phân tích Kiến trúc (Architectural Analysis)

cuu duong than cong . com

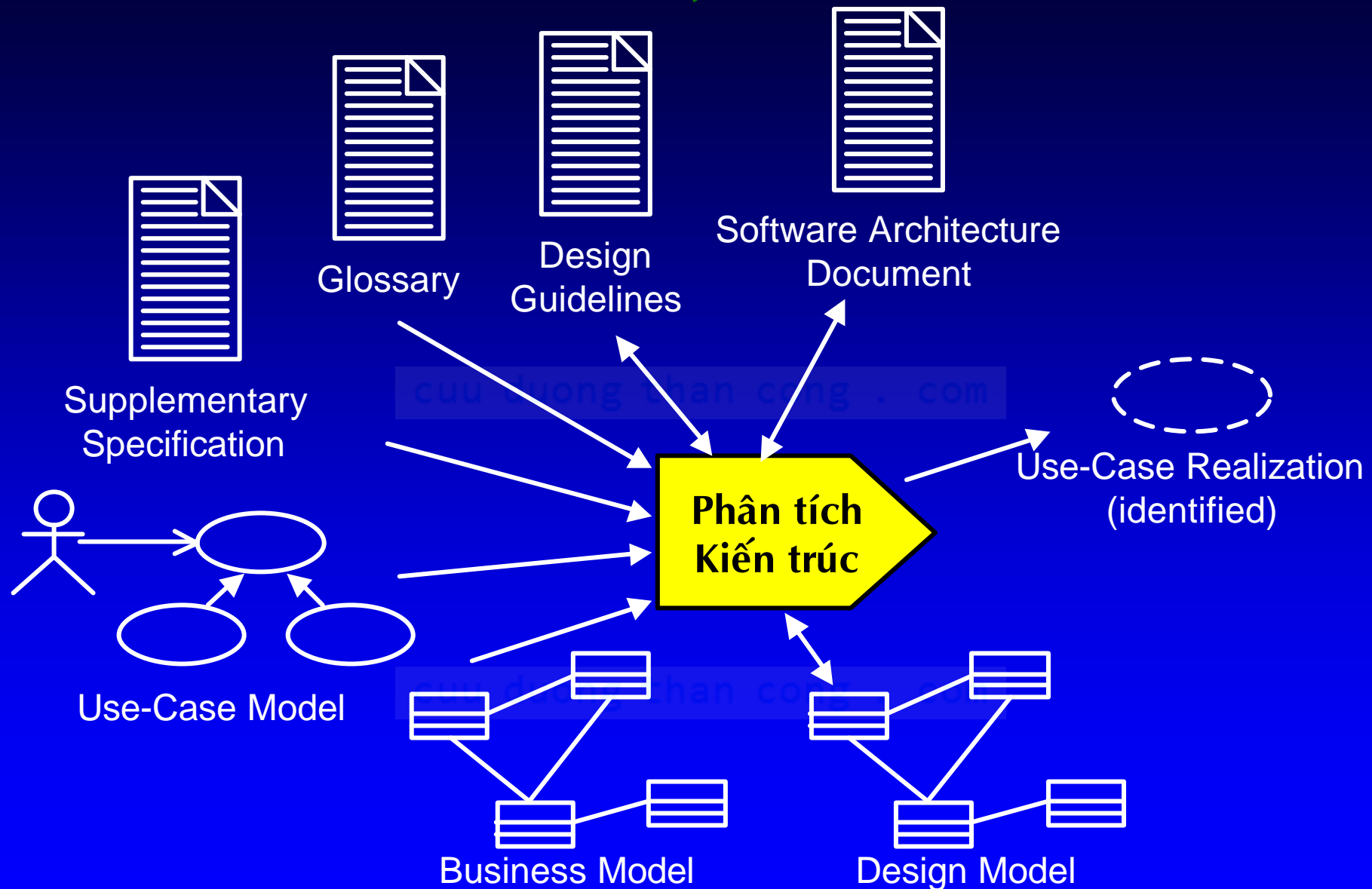
Mục tiêu:

- ✍ Tìm hiểu mục đích của Phân tích Kiến trúc và nơi thực hiện công việc này trong chu kỳ sống của hệ thống
- ✍ Mô tả một mẫu biểu diễn kiến trúc và một tập hợp các cơ chế phân tích cùng với ảnh hưởng của chúng đến kiến trúc
- ✍ Tìm hiểu nguồn gốc căn bản và các khảo sát hợp lý nhằm hỗ trợ cho các quyết định liên quan đến kiến trúc (hệ thống)
- ✍ Tìm hiểu cách đọc và diễn dịch các kết quả của Phân tích Kiến trúc
 - ✍ Các tầng kiến trúc và quan hệ giữa chúng
 - ✍ Các trừu tượng hóa chính
 - ✍ Các cơ chế phân tích

Phân tích kiến trúc trong ngữ cảnh



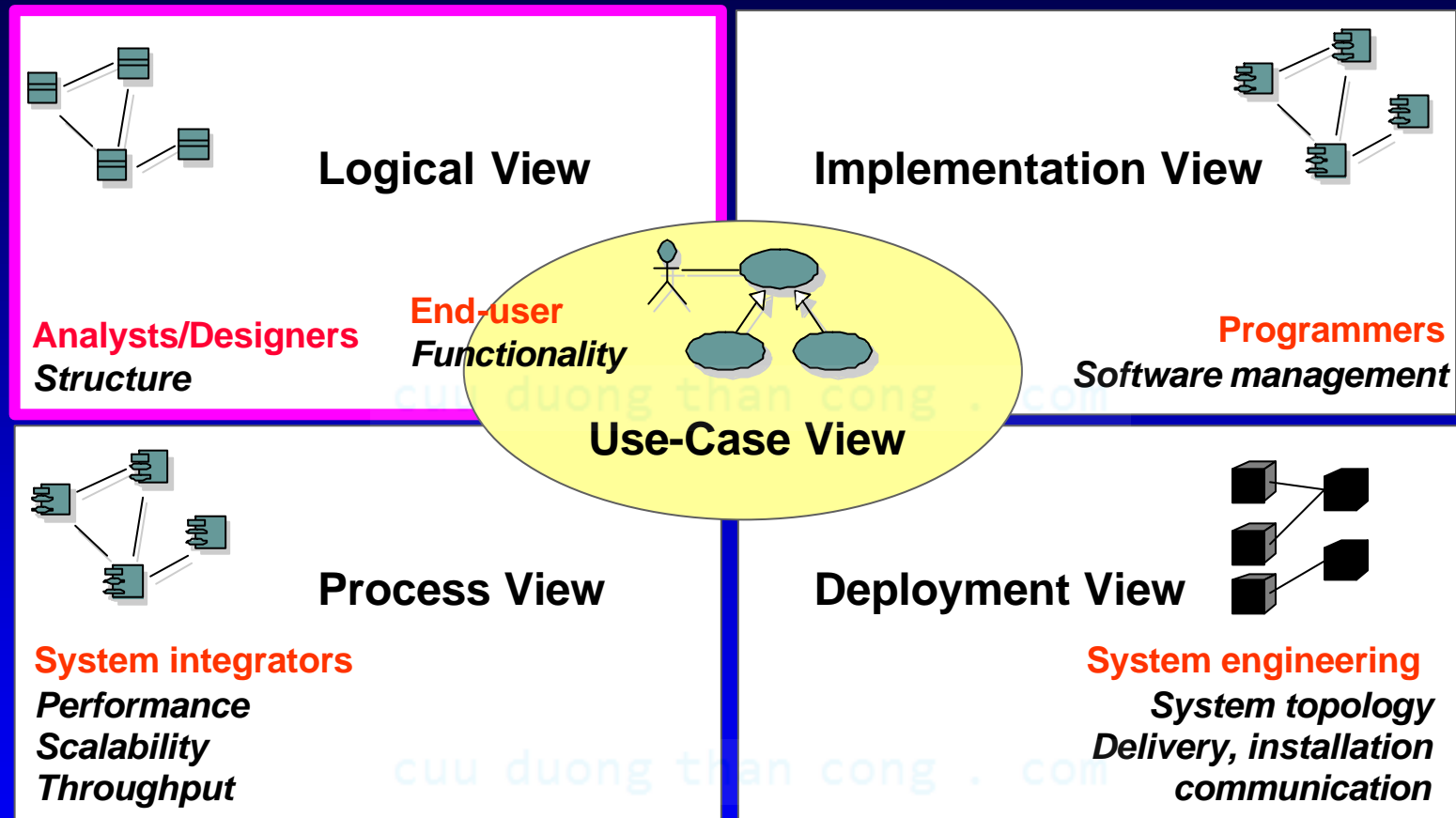
Tổng quan về phân tích kiến trúc



Các chủ đề:

- ★ ✎ Các khái niệm then chốt
- ✎ Các qui ước trong mô hình hóa
- ✎ Các cơ chế phân tích
- ✎ Các trừu tượng hóa chính
- ✎ Các tầng kiến trúc ban đầu
- ✎ Checkpoints

Kiến trúc là gì: Mô hình “4+1 View”



Nhắc lại: Package là gì ?

- ✍ Package là một cơ chế để tổ chức các phần tử thành nhóm
- ✍ Là một phần tử của mô hình có thể chứa các phần tử khác

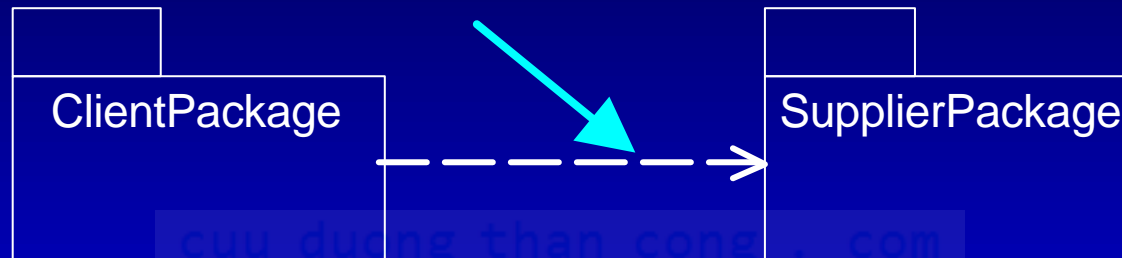


- ✍ Dùng để
 - ✍ Tổ chức một mô hình đang trong q/t phát triển
 - ✍ Làm một đơn vị trong quản trị cấu hình

Các mối quan hệ giữa Packages: Dependency

✎ Các Package có thể liên hệ với nhau thông qua mối quan hệ dependency

Dependency relationship

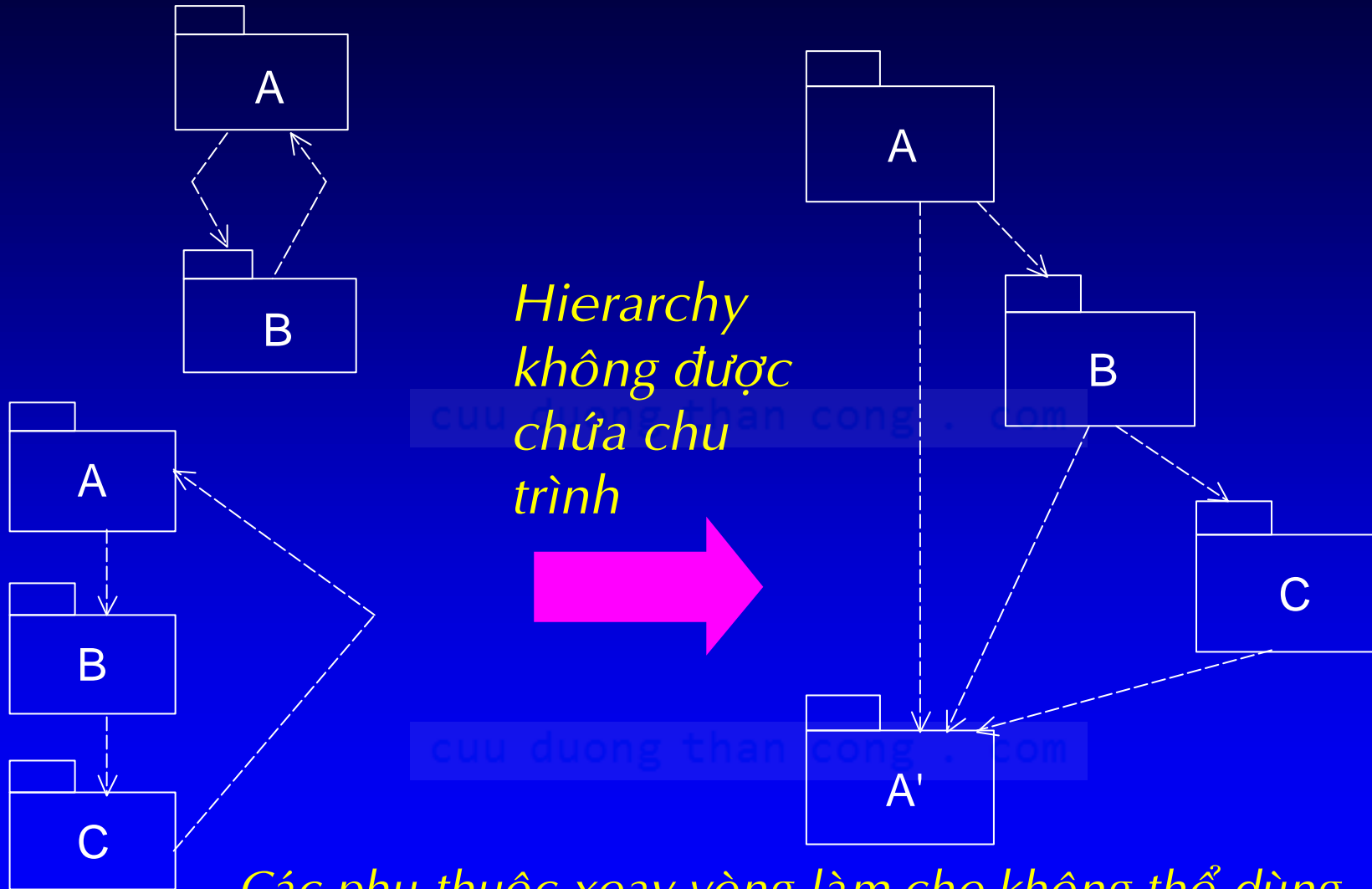


✎ Dependency hàm nghĩa

Các thay đổi ở Supplier package có thể ảnh hưởng đến Client package

Client package không thể được dùng lại một cách độc lập vì nó phụ thuộc vào Supplier package

Loại bỏ các phụ thuộc xoay vòng



Các phụ thuộc xoay vòng làm cho không thể dùng lại một package khi không có các package khác

Các chủ đề:

 Các khái niệm then chốt

★  Các quy ước trong mô hình hóa

 Các cơ chế phân tích

 Các trừu tượng hóa chính

 Các tầng kiến trúc ban đầu

 Checkpoints

Các qui ước trong mô hình hóa

✍ Chúng là những gì?

- ✍ Dùng những diagram và phần tử mô hình nào
- ✍ Các luật để sử dụng các phần tử mô hình và diagram
- ✍ Qui ước về đặt tên

✍ Các ví dụ



- ✍ Các modeling construct không được dùng
- ✍ Các diagram phải hiện diện
- ✍ Phải dùng các diagram để mô hình hóa các architectural view
- ✍ Cách trình bày mô hình (Model layout)

Ví dụ: (Modeling Conventions)

Use-Case View

-  Dùng các câu ngắn ở thể chủ động để đặt tên cho các Use Case, ví dụ Submit Grades, Vô điểm

Logical View

-  Một Use-Case Realization package chứa:
 - Ít nhất một realization cho mỗi use case
 - Một View Of Participating Classes diagram thể hiện tất cả các class trong realization và các quan hệ cần thiết của chúng
-  Dùng các danh từ để đặt tên cho các Class. Tên càng phù hợp với ý nghĩa ứng dụng càng tốt

Các chủ đề:

 Các khái niệm then chốt

 Các qui ước trong mô hình hóa

★  Các cơ chế phân tích

 Các trừu tượng hóa chính

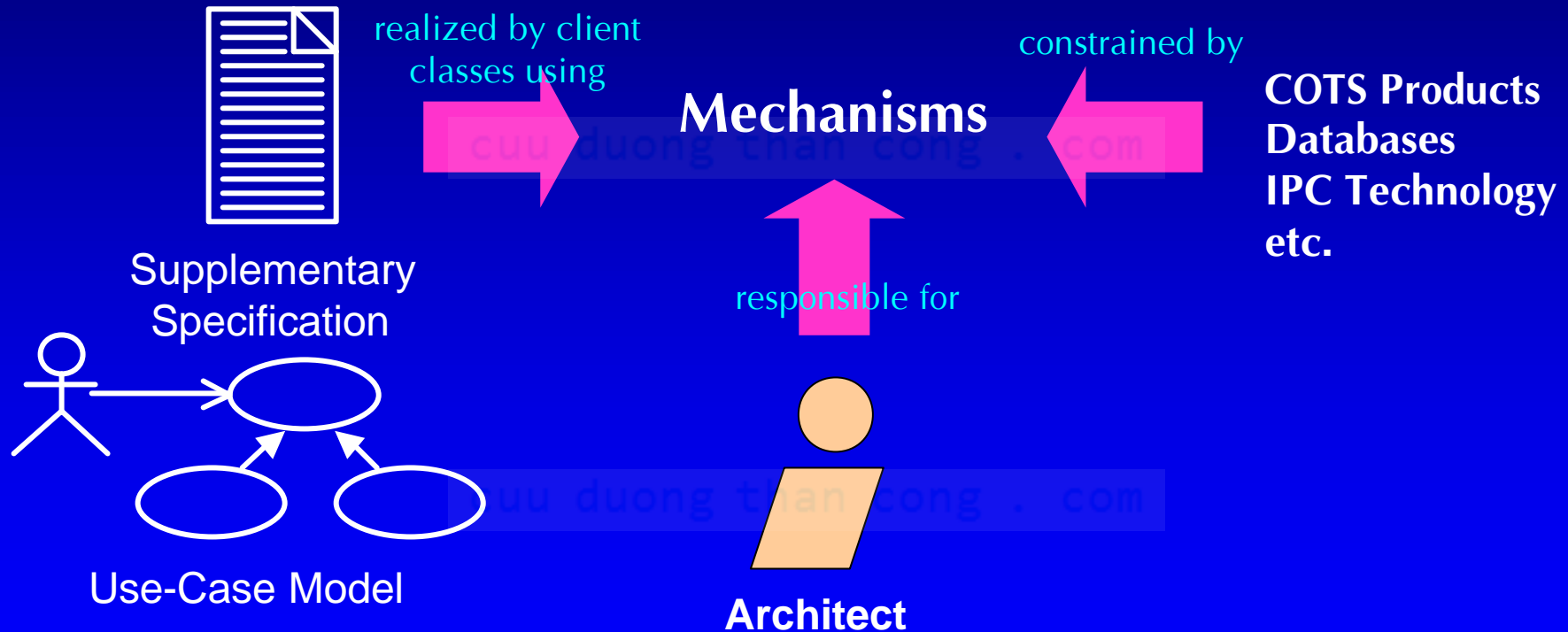
 Các tầng kiến trúc ban đầu

 Checkpoints

Các cơ chế kiến trúc là gì?

**Required
Functionality**

**Implementation
Environment**



Ba loại cơ chế kiến trúc

✍ Các loại cơ chế kiến trúc

✍ Các cơ chế phân tích (conceptual)

✍ Các cơ chế thiết kế (concrete)

✍ Các cơ chế cài đặt (actual)

cuu duong than cong . com

cuu duong than cong . com

Các Analysis Mechanism mẫu

- ✍ Persistency
- ✍ Communication (IPC and RPC)
- ✍ Message routing
- ✍ Distribution
- ✍ Transaction management
- ✍ Process control and synchronization (resource contention)
- ✍ Information exchange, format conversion
- ✍ Security
- ✍ Error detection / handling / reporting
- ✍ Redundancy
- ✍ Legacy Interface

Các đặc trưng của Analysis Mechanism

Persistency

 Granularity

 Volume

 Duration

 Access mechanism

 Access frequency (creation/deletion, update, read)

 Reliability

Communication

 Latency

 Synchronicity

 Message Size

 Protocol

Các đặc trưng của Analysis Mechanism (tt)

- ✎ Legacy interface
 - ✎ Latency
 - ✎ Duration
 - ✎ Access mechanism
 - ✎ Access frequency
- ✎ Security
 - ✎ Data granularity
 - ✎ User granularity
 - ✎ Security rules
 - ✎ Privilege types
- ✎ etc.

Ví dụ: Các cơ chế phân tích trong “ĐKý HP”

- ✗ Persistence
- ✗ Distribution
- ✗ Security
- ✗ Legacy Interface

cuu duong than cong . com

cuu duong than cong . com

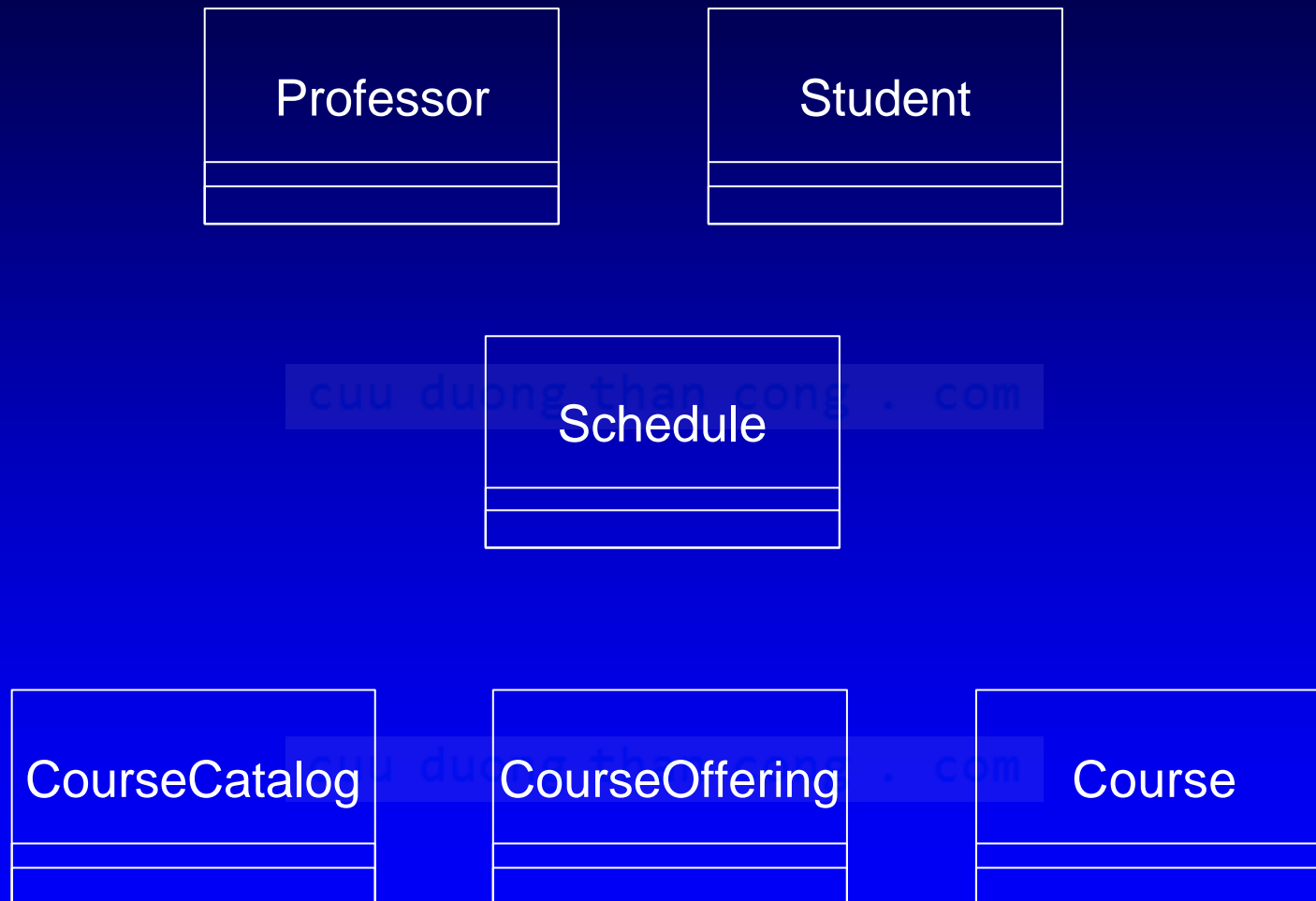
Các chủ đề:

- ✍ Các khái niệm then chốt
- ✍ Các qui ước trong mô hình hóa
- ✍ Các cơ chế phân tích
- ★ ✍ Các trừu tượng hóa chính
- ✍ Các tầng kiến trúc ban đầu
- ✍ Checkpoints

Xác định các trừu tượng hóa chính

- ✍ Định nghĩa sơ bộ các class (sources) từ:
 - ✍ Tri thức về miền ứng dụng
 - ✍ Các y/c đặt ra cho hệ thống (Requirements)
 - ✍ Bảng chú giải (Glossary)
 - ✍ Domain Model, hoặc Mô hình nghiệp vụ (nếu có)
- ✍ Định nghĩa analysis class relationships
- ✍ Mô hình hóa các analysis class và các quan hệ của chúng trên Class Diagram
 - ✍ Đính kèm mô tả ngắn gọn của analysis class
- ✍ Ánh xạ các analysis class với các analysis mechanism cần thiết

Ví dụ: Key Abstractions




Các chủ đề:

- ✍ Các khái niệm then chốt
- ✍ Các qui ước trong mô hình hóa
- ✍ Các cơ chế phân tích
- ✍ Các trừu tượng hóa chính
- ★ ✍ Các tầng kiến trúc ban đầu
- ✍ Checkpoints

Patterns va Frameworks

Pattern (Khuôn mẫu)

 Là một lời giải chung cho một bài toán trong ngữ cảnh hiện hành

Analysis/Design Pattern

 Lời giải cho một bài toán kỹ thuật hẹp

 Một đoạn của lời giải, một mảnh của puzzle

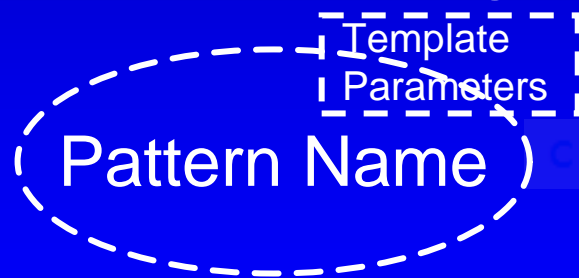
Framework

 Định nghĩa hướng tiếp cận tổng quát để giải quyết bài toán

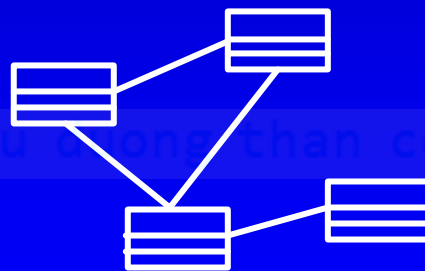
 Sườn của lời giải, mà chi tiết của nó có thể là các analysis/design pattern

Design Patterns

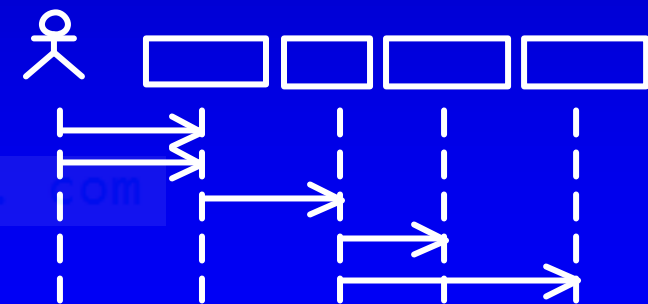
- ✍ Design pattern là lời giải chung của một design problem
 - ✍ Mô tả design problem chung
 - ✍ Mô tả lời giải của bài toán
 - ✍ Thảo luận về các kết quả và cân nhắc việc sử dụng hiệu quả pattern
- ✍ Design pattern cung cấp khả năng tái sử dụng thành công các thiết kế



*Parameterized
collaboration*



Structural Aspect



Behavioral Aspect

Architectural Patterns

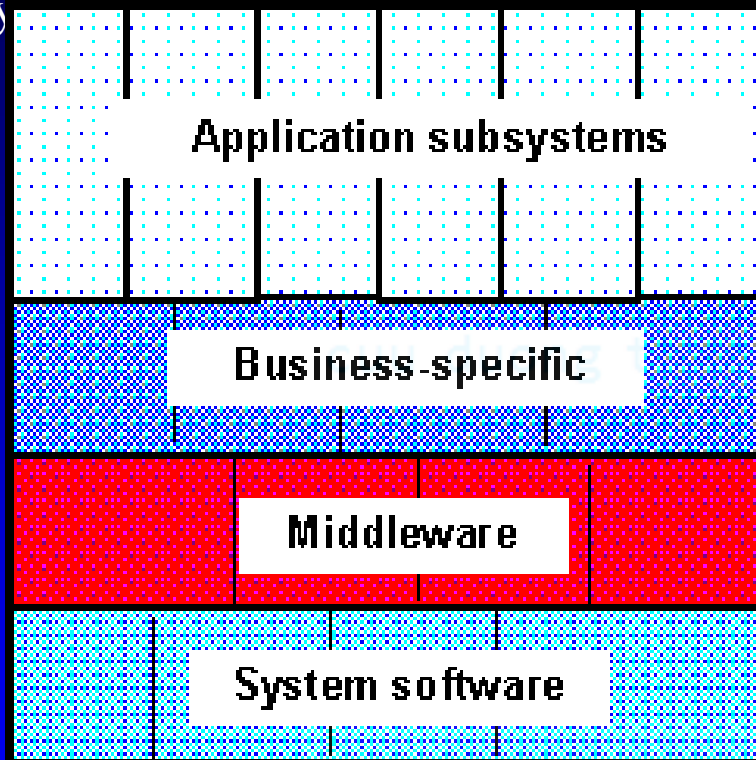
- ✍ Các tầng (Layers)
- ✍ Model-view-controller (M-V-C)
- ✍ Pipes và filters
- ✍ Blackboard

cuu duong than cong . com

cuu duong than cong . com

Hướng tiếp cận phân lớp truyền thống

Specific
functionality



Lớp các application subsystem khác nhau, mà tất cả đều thành một ứng dụng, chia các giá trị bổ sung cho phần mềm đơn lẻ đang phát triển

Lớp nghiệp vụ chuyên nghiệp (Business-specific) chia một số các subsystem có thể ứng dụng cho các nghiệp vụ

Lớp Middleware đưa ra các subsystem chia các class tiện ích và các dịch vụ để lớp platform hỗ trợ cho các ứng dụng chạy trên các môi trường không thuần nhất.

Lớp phần mềm hỗ trợ ứng dụng chia phần mềm dành cho kiến trúc hệ thống như các hệ điều hành, các giao tiếp với phần cứng, trình điều khiển thiết bị ...




General
functionality

Làm thế nào để tìm thấy các Layer?




Mức trừu tượng

-  Nhóm các phần tử cùng chung mức độ trừu tượng

Phân tách các thành phần liên quan

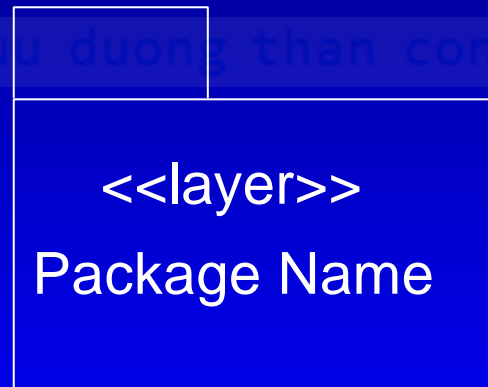
-  Nhóm những gì giống nhau lại chung
-  Phân biệt những gì khác biệt nhau
-  Application vs. Domain model elements

Sự co giãn (Resiliency)

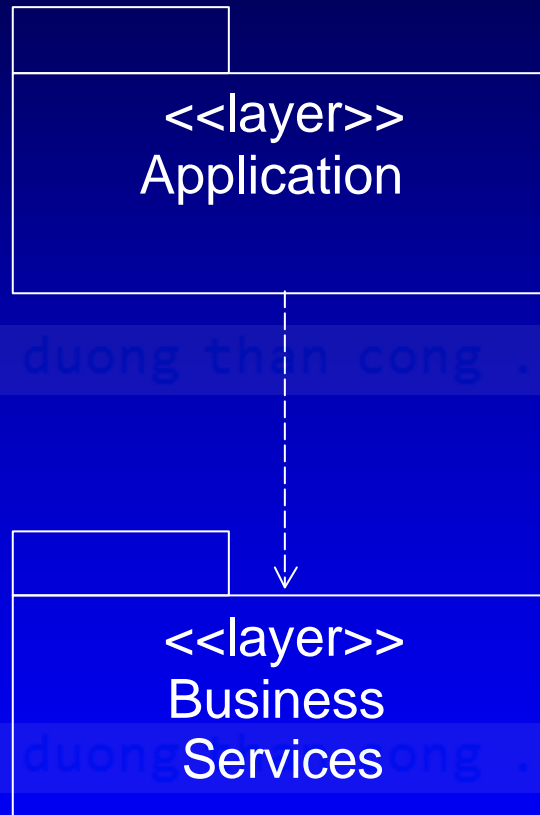
-  Sự kết hợp lỏng lẻo
-  Chú trọng đến các thay đổi (encapsulating)
-  User interface, business rules, và dữ liệu có khả năng thay đổi cao

Modeling Architectural Layers

- ✍ Architectural layers can be modeled using stereotyped packages
- ✍ <<layer>> stereotype



Ví dụ: Tổ chức cấp cao của Model





Các chủ đề:

- ✍ Các khái niệm then chốt
- ✍ Các qui ước trong mô hình hóa
- ✍ Các cơ chế phân tích
- ✍ Các trừu tượng hóa chính
- ✍ Các tầng kiến trúc ban đầu


★ ✍ Checkpoints

Checkpoints

Tổng thể

-  Việc phân chia các package (partitioning và layering) có được thực hiện một cách chắc chắn và hợp lý không ?
-  Các analysis mechanisms cần thiết đã được xác định đầy đủ ?

Packages

-  Chúng ta đã cung cấp một hình ảnh toàn diện (comprehensive) về các dịch vụ của packages trong các upper-level layer chưa ?

Checkpoints (cont.)

✍ Các Class

- ✍ Các key entity class và những mối quan hệ của chúng đã được xác định và mô hình một cách chính xác đúng đắn chưa?
- ✍ Tên của mỗi class có thể hiển rõ ràng vai trò của chúng không ?
- ✍ Các key abstraction/class và những mối quan hệ của chúng có nhất quán với business model, domain model, requirements, glossary, ... không ?

Review: Architectural Analysis

- ✍ Mục tiêu của Architectural Analysis là gì?
- ✍ Modeling Conventions là gì và tại sao phải cần đến chúng ? Cho ví dụ.
- ✍ Package là gì ?
- ✍ Analysis Mechanisms là gì ? Cho ví dụ.
- ✍ Những key abstractions nào được xác định trong Architectural Analysis? Tại sao chúng lại được xác định ở đây?
- ✍ Kiến trúc phân lớp là gì ? Cho ví dụ về các layer truyền thống.

Bài tập:

✎ Làm các công việc sau:

✎ Cho một số kết quả của luồng công việc đặc tả y/c người dùng:

Phát biểu bài toán

Use-Case Model main diagram

Glossary

✎ Cho một số quyết định về kiến trúc hệ thống:

Các upper-level architectural layer và các mối phụ thuộc của chúng (bằng văn bản)

Bài tập: (tt)

✍ Xác định:

✍ Các key abstraction

✍ Xây dựng:

✍ Class diagram chứa các key abstraction

✍ Class diagram chứa các upper-level architectural layer và các mối phụ thuộc của chúng

cuu duong than cong . com