

---

# Phân tích và Thiết kế Hướng đối tượng dùng UML

cuu duong than cong . com

## Module 11: Thiết kế Use-Case

cuu duong than cong . com

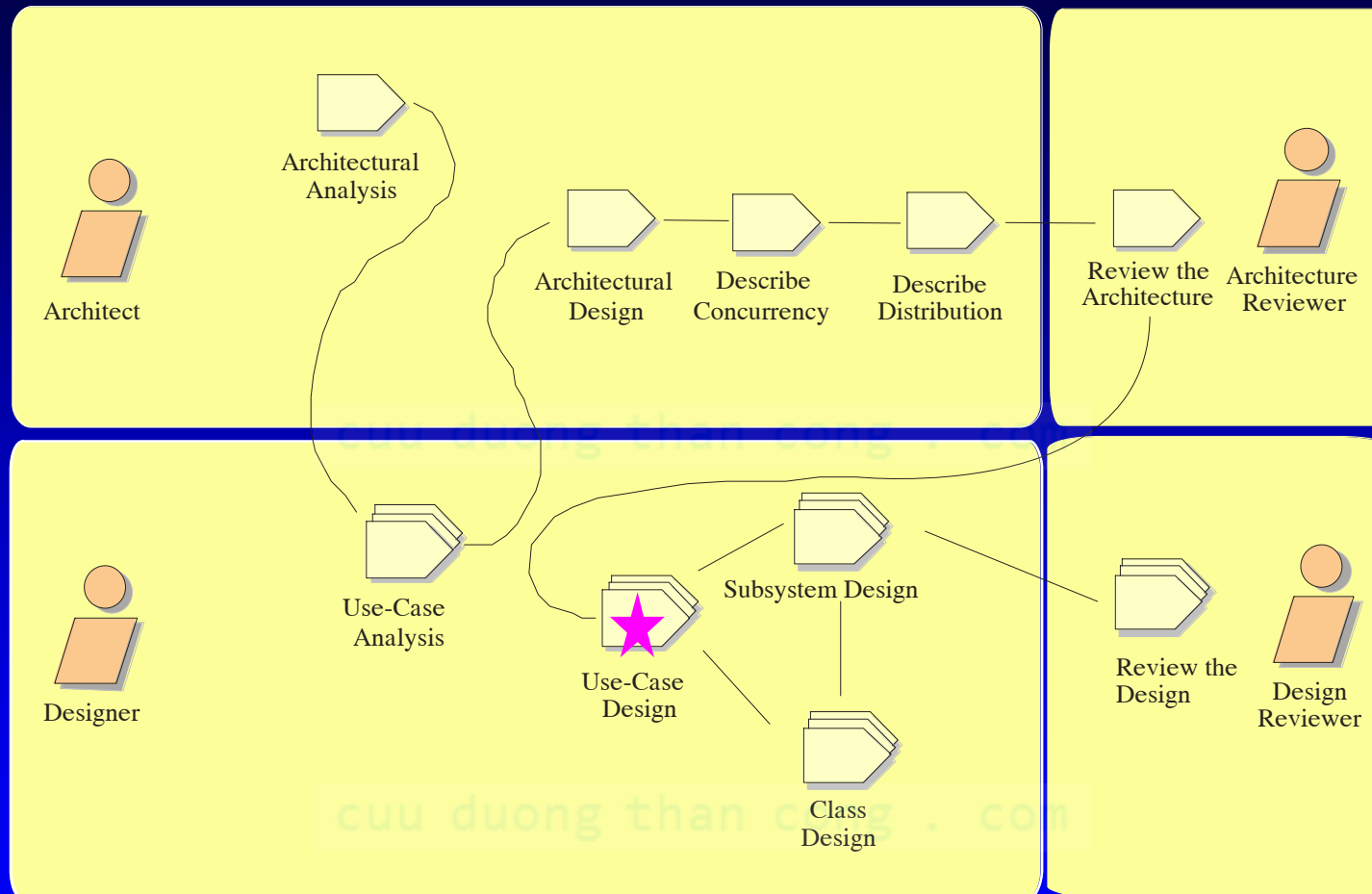
# Mục tiêu

---

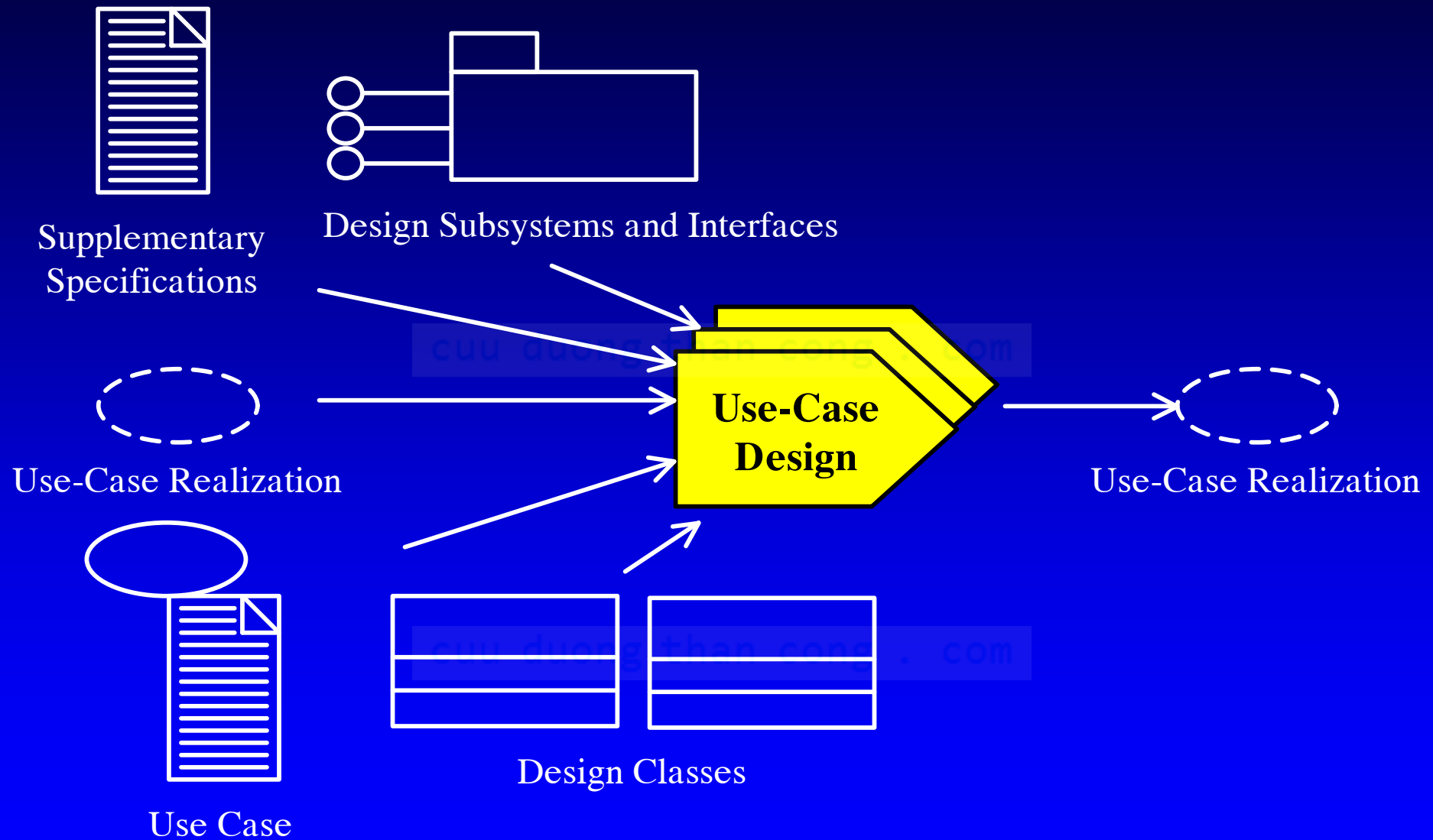
- ◆ Tìm hiểu mục đích của bước thiết kế Use-Case và thời điểm thực hiện công đoạn này
- ◆ Kiểm định tính nhất quán trong cài đặt use-case
- ◆ Tinh chỉnh use-case realizations có được từ bước phân tích Use-Case dựa trên các phần tử thiết kế đã được xây dựng

cuu duong than cong . com

# Vị trí của Thiết kế Use-Case



# Tổng quan về Thiết kế Use-Case



# Các bước thiết kế Use-Case

---

- ◆ Mô tả tương tác giữa các Design Object
- ◆ Đơn giản hóa các Interaction Diagram nhờ vào các Subsystem (optional)
- ◆ Mô tác các hành vi liên quan đến tính Persistence
- ◆ Tinh chỉnh mô tả về các Flow of Events
- ◆ Hợp nhất các Class và các Subsystem
- ◆ Checkpoints

# Nhắc lại: Use-Case Realization

*Use-Case Model*

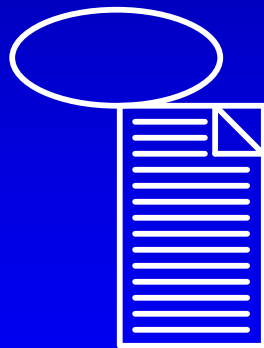


Use Case

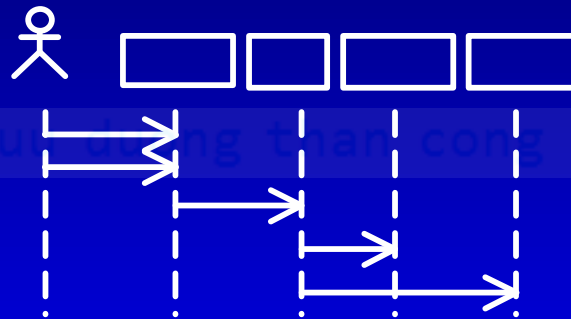
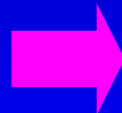
*Design Model*



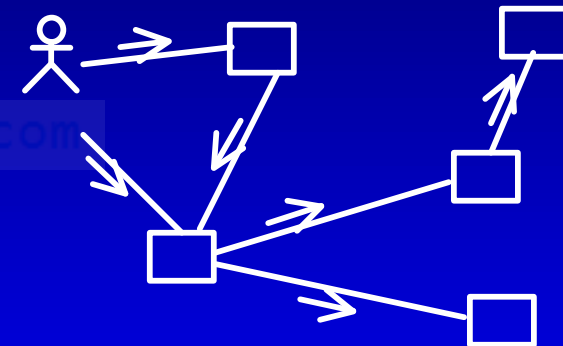
Use-Case Realization



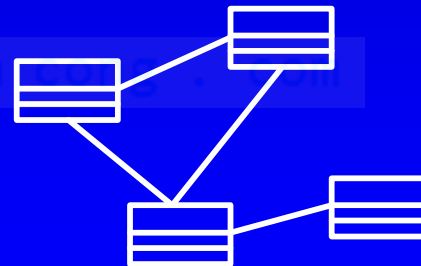
Use Case



Sequence Diagrams



Collaboration Diagrams



Class Diagrams

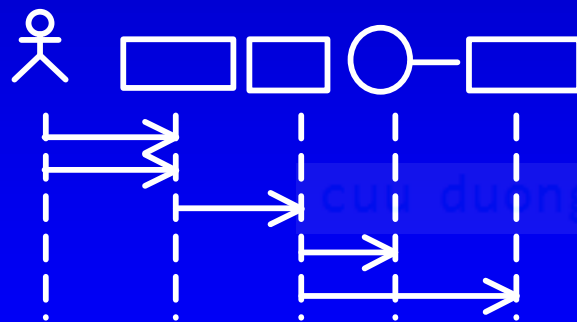
# Các bước thiết kế Use-Case

---

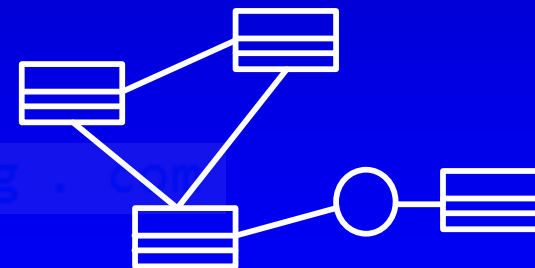
- ★ ♦ Mô tả tương tác giữa các Design Object
- ♦ Đơn giản hóa các Interaction Diagram nhờ vào các Subsystem (optional)
- ♦ Mô tác các hành vi liên quan đến tính Persistence
- ♦ Tinh chỉnh mô tả về các Flow of Events
- ♦ Hợp nhất các Class và các Subsystem
- ♦ Checkpoints

# Tinh chỉnh Use-Case Realization

- ◆ Xác định các object có tham gia vào Use-Case
- ◆ Phân công trách nhiệm cho các object
- ◆ Mô hình hóa các thông điệp giữa các object
- ◆ Mô tả các kết quả xử lý từ các thông điệp
- ◆ Mô hình hóa quan hệ giữa các class liên quan



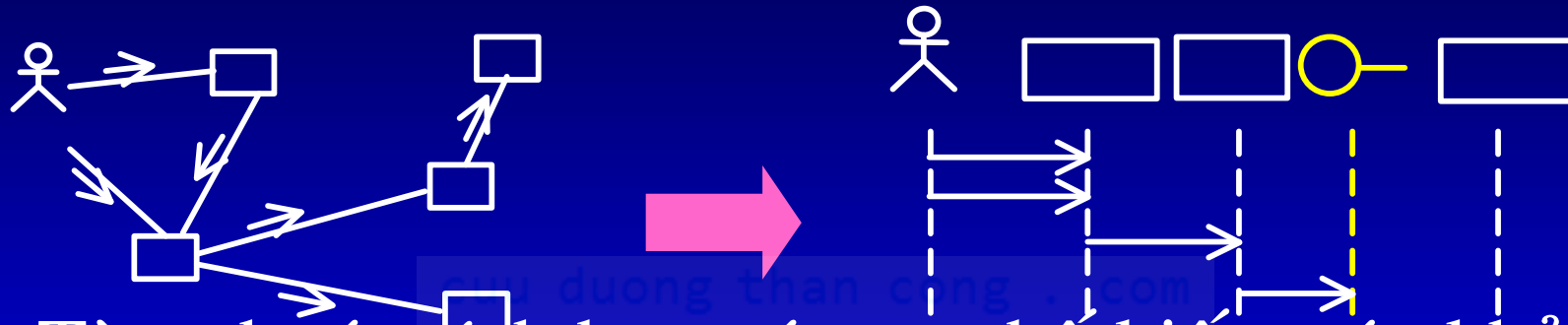
Sequence Diagrams



Class Diagrams

# Các bước tinh chỉnh Use-Case Realization

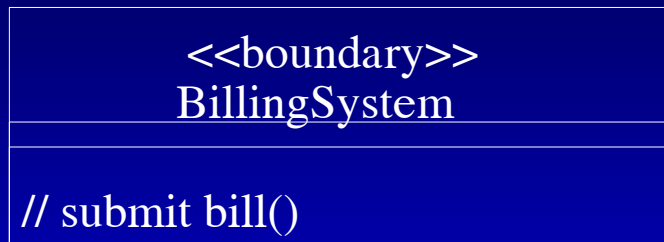
- ◆ Thay thế các class khả dụng bằng các subsystem interface kết hợp với chúng



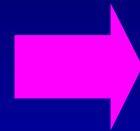
- ◆ Từng bước tích hợp các cơ chế kiến trúc khả dụng
- ◆ Hiệu chỉnh use-case realization
  - Các Interaction diagram
  - View of participating classes (VOPC) class diagram(s)

# Ví dụ: Tích hợp Subsystem Interfaces

## Analysis Classes

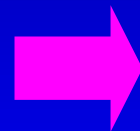
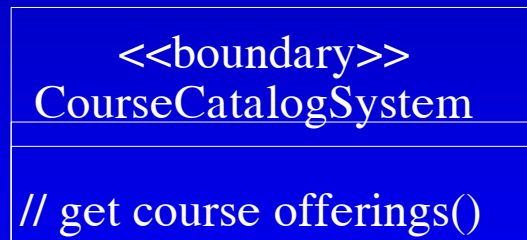
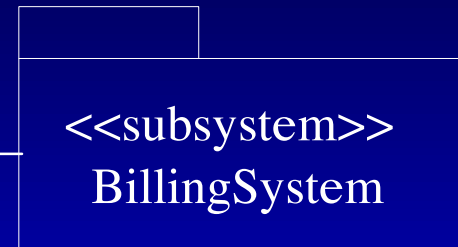


## Design Elements



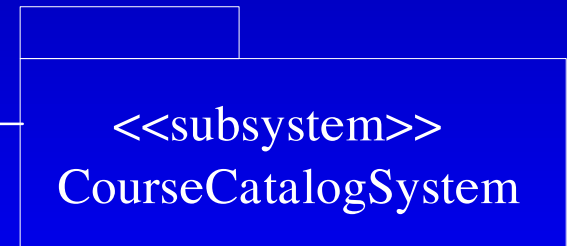
IBillingSystem

submitBill(forTuition : Double, forStudent : Student)



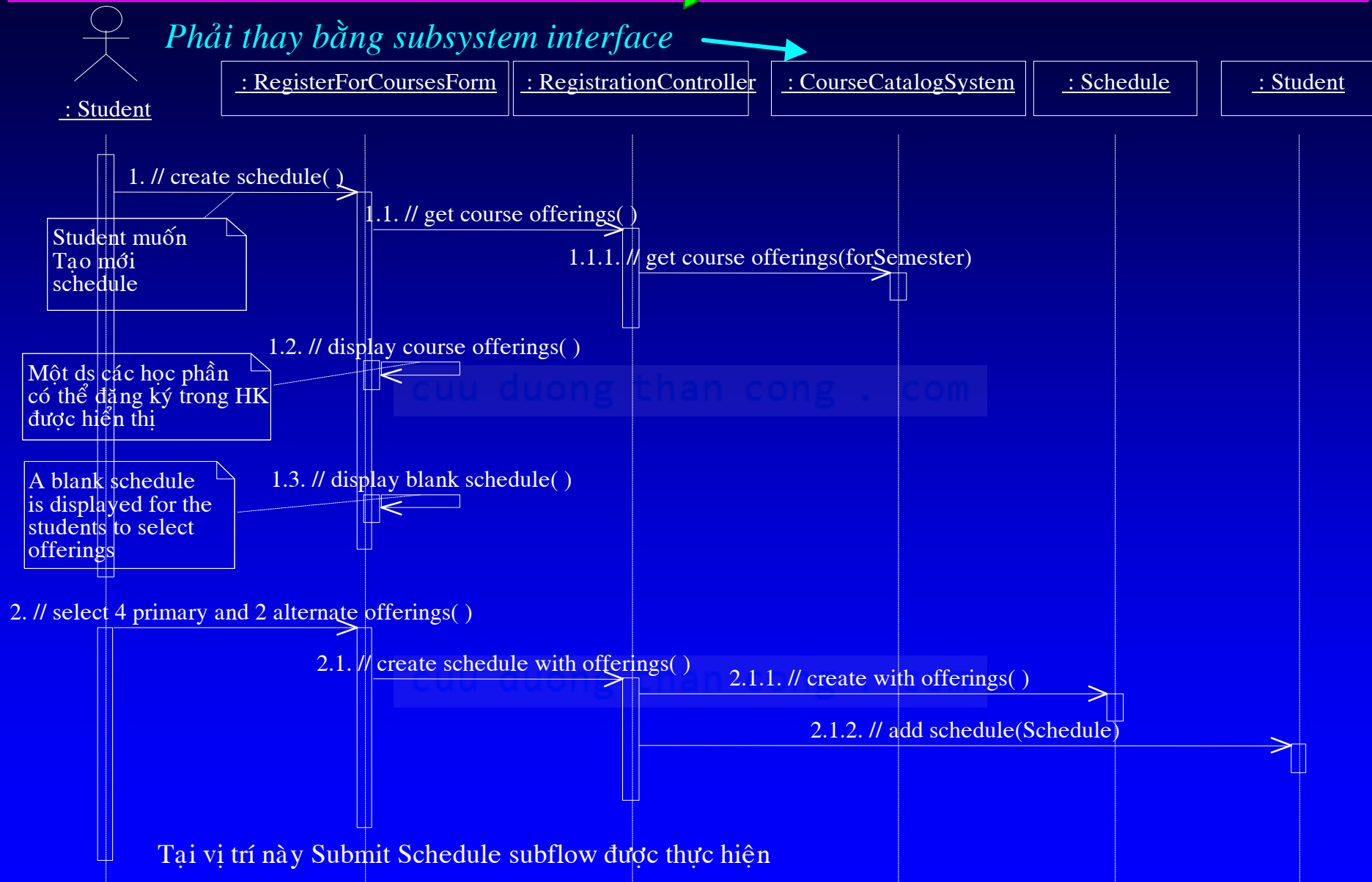
ICourseCatalogSystem

getCourseOfferings(forSemester : Semester) : CourseOfferingList



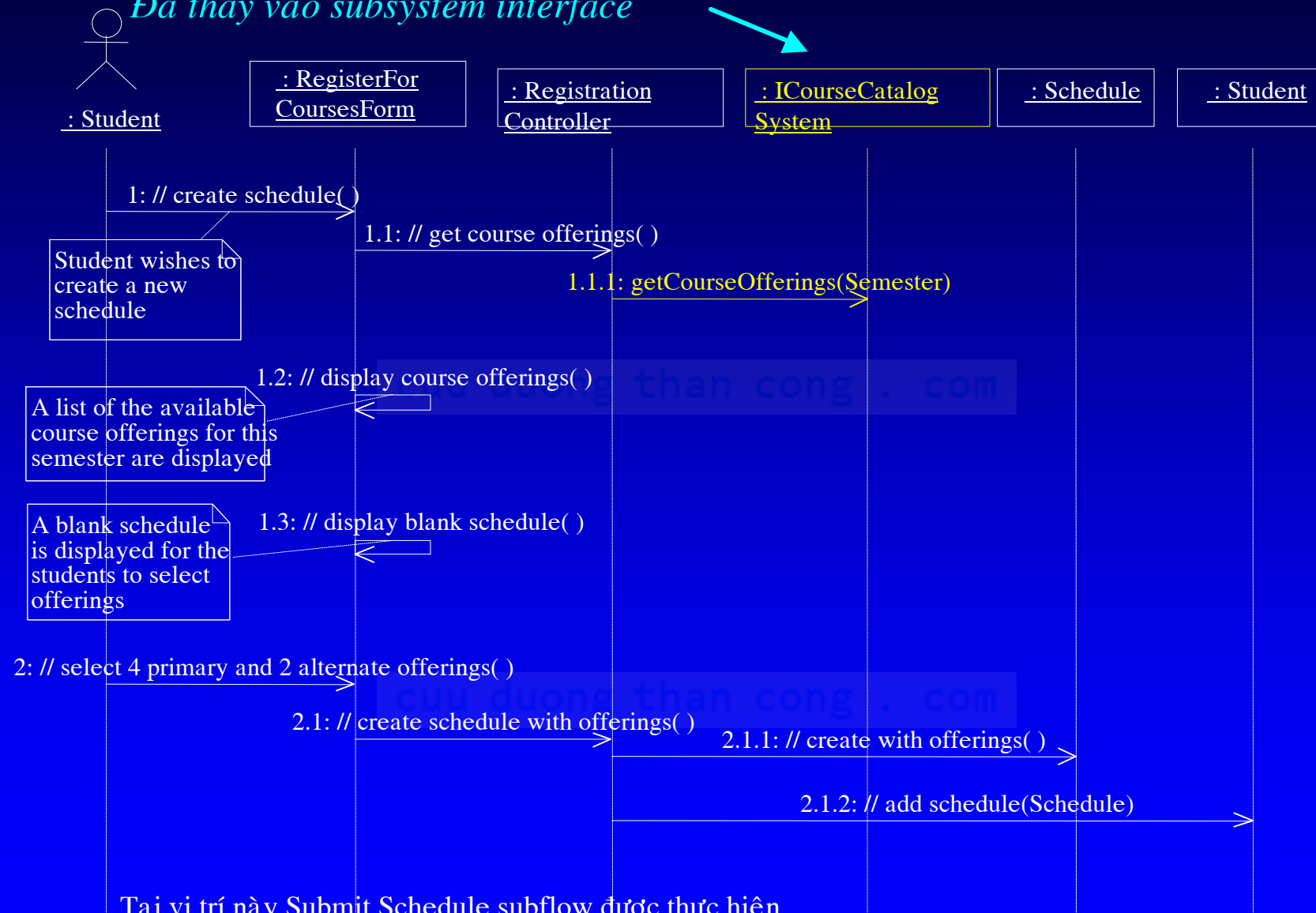
Tất cả các analysis class khác được ánh xạ thành các design class

# Ví dụ: Trước khi tích hợp SubSystem Interfaces



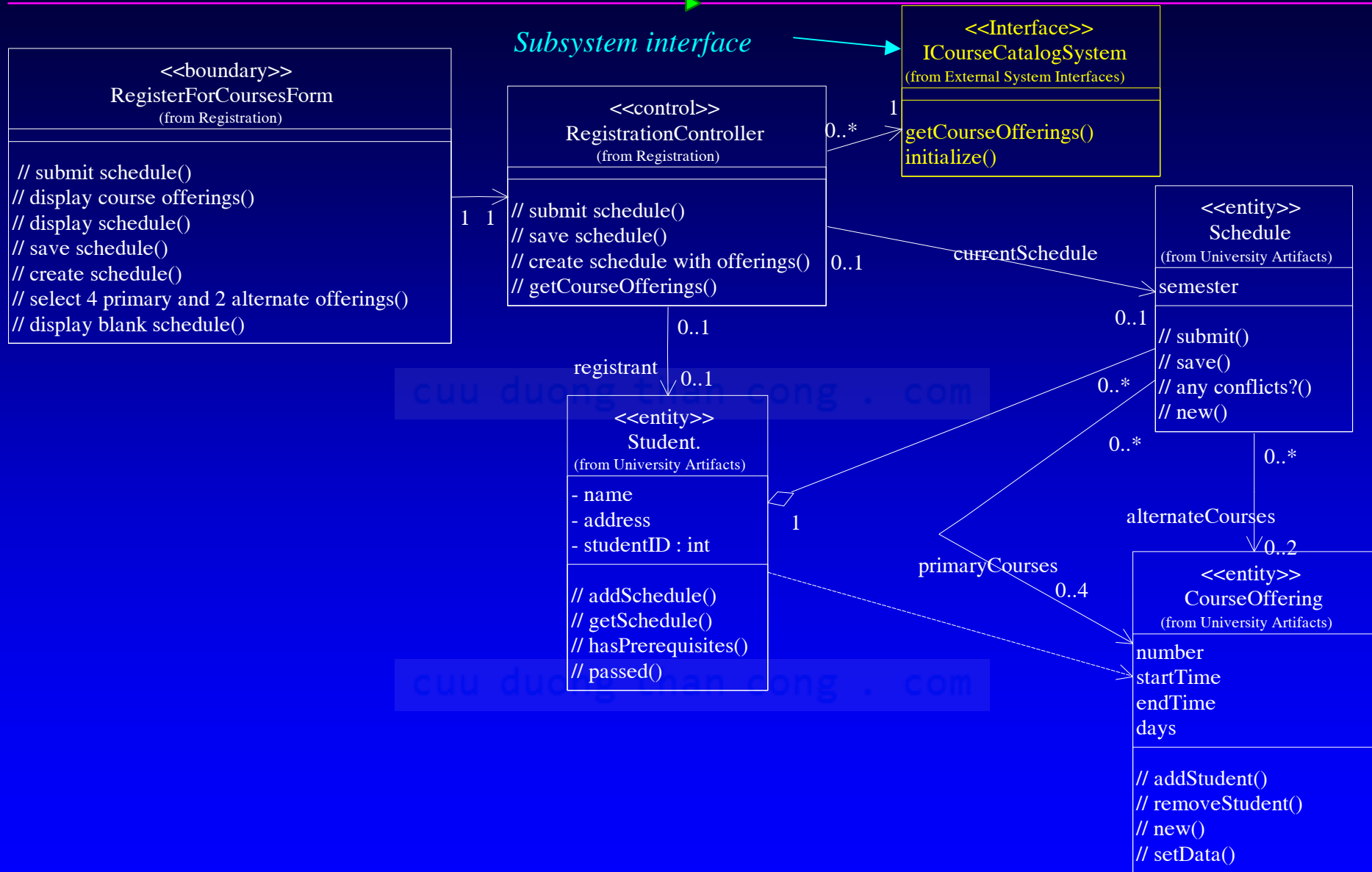
# Ví dụ: Sau khi tích hợp Subsystem Interface

Đã thay vào subsystem interface



Tại vị trí này Submit Schedule subflow được thực hiện

# Ví dụ: Tích hợp Subsystem Interfaces (VOPC)



# Tích hợp các cơ chế kiến trúc: Security

- ◆ Bảng ánh xạ các Analysis-Class với các cơ chế kiến trúc có từ bước phân tích Use-Case

Analysis Class	Các cơ chế
Student	Persistency, <i>Security</i>
Schedule	Persistency, <i>Security</i>
CourseOffering	Persistency, Legacy Interface
Course	Persistency, Legacy Interface
RegistrationController	Distribution

# Tích hợp các cơ chế kiến trúc: Distribution

- ◆ Bảng ánh xạ các Analysis-Class với các cơ chế kiến trúc có từ bước phân tích Use-Case

Analysis Class	Các cơ chế
Student	Persistency, Security
Schedule	Persistency, Security
CourseOffering	Persistency, Legacy Interface
Course	Persistency, Legacy Interface
RegistrationController	<i>Distribution</i>

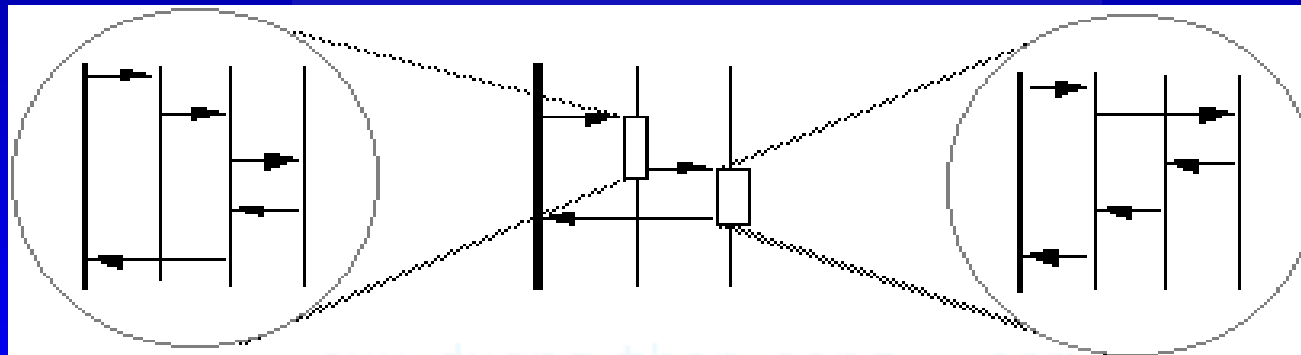
# Các bước thiết kế Use-Case

---

- ◆ Mô tả tương tác giữa các Design Object
- ★◆ Đơn giản hóa các Interaction Diagram nhờ vào các Subsystem (optional)
- ◆ Mô tác các hành vi liên quan đến tính Persistence
- ◆ Tinh chỉnh mô tả về các Flow of Events
- ◆ Hợp nhất các Class và các Subsystem
- ◆ Checkpoints

# Đóng gói các Subsystem Interaction

- ◆ Có thể mô tả các tương tác dưới nhiều mức độ khác nhau
- ◆ Tương tác giữa các Subsystem có thể mô tả bởi các interaction diagram của chúng



*Tăng mức độ trừu tượng*

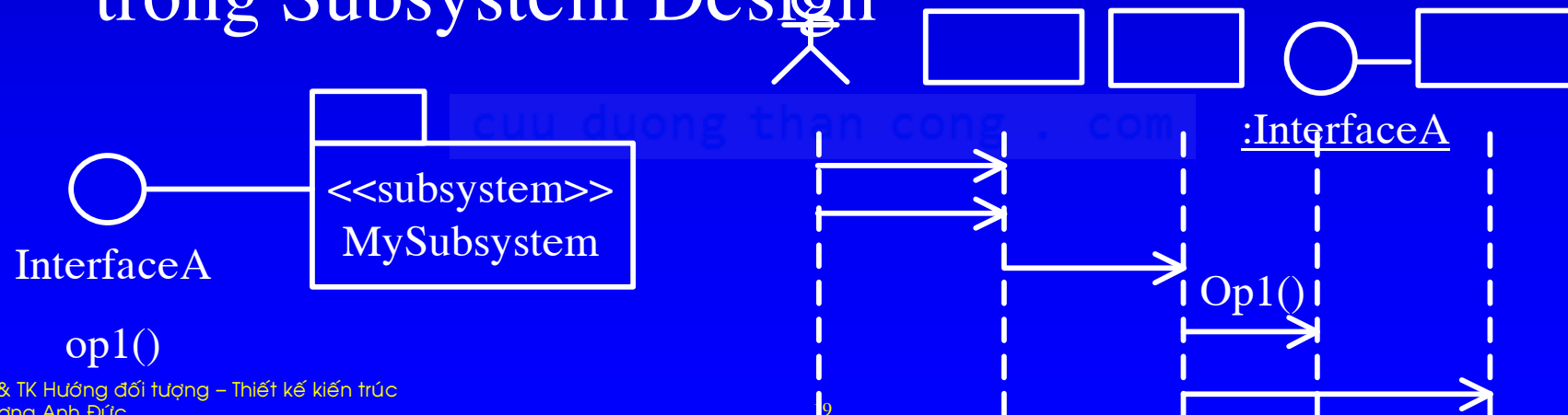
# Khi nào đóng gói Sub-Flows trong Subsystem

- ◆ Sub-flow xuất hiện trong nhiều use-case realizations
- ◆ Sub-flow có tiềm năng tái sử dụng
- ◆ Sub-flow phức tạp và dễ dàng đóng gói
- ◆ Sub-flow do 1 người/đội đảm nhiệm
- ◆ Sub-flow tạo ra một kết quả xác định tốt
- ◆ Sub-flow được gói gọn trong một component trong mô hình cài đặt

[cuuduongthancong.com](http://cuuduongthancong.com)

# Guidelines: Đóng gói Subsystem Interactions

- ◆ Các Subsystem phải được biểu diễn với các interface của chúng trong interaction diagrams
- ◆ Các thông điệp đến subsystems được mô hình như các thông điệp đến subsystem interface
- ◆ Các thông điệp đến subsystems tương ứng với các operation của subsystem interface
- ◆ Các tương tác trong subsystems được mô hình trong Subsystem Design



# Lợi ích của việc đóng gói Subsystem Interaction

- ◆ Use-case realization bớt hỗn độn
- ◆ Use-case realization có thể được tạo trước khi xây dựng thiết kế bên trong của subsystems (parallel development)
- ◆ Use-case realizations **generic** hơn và dễ dàng thay đổi (subsystems có thể được thay thế)

cuu duong than cong . com

# Parallel Subsystem Development

- ◆ Chú ý vào các y/c ảnh hưởng đến subsystem interfaces
- ◆ Phác thảo các interface cần thiết
- ◆ Mô hình hóa các thông điệp băng qua ranh giới các subsystem
- ◆ Vẽ interaction diagrams dùng các subsystem interfaces cho mỗi use case
- ◆ Tinh chỉnh các interface cần để cung cấp các thông điệp
- ◆ Phát triển song song các subsystem  
*Dùng các subsystem interface như điểm đồng bộ hóa*

# Các bước thiết kế Use-Case

---

- ◆ Mô tả tương tác giữa các Design Object
- ◆ Đơn giản hóa các Interaction Diagram nhờ vào các Subsystem (optional)
- ★ ◆ Mô tác các hành vi liên quan đến tính Persistence
- ◆ Tinh chỉnh mô tả về các Flow of Events
- ◆ Hợp nhất các Class và các Subsystem
- ◆ Checkpoints

# Mô tả các hành vi liên quan đến cơ chế Persistence

---

- ◆ Mô tả các hành vi liên quan đến cơ chế Persistence
  - Mô hình hóa các Transaction
  - Lưu (ghi) các Persistent Object
  - Đọc các Persistent Object
  - Hủy các Persistent Object

# Mô hình hóa các Transaction

- ♦ Transaction là gì?
  - Lỗi gọi đến các Atomic operation
  - “Tất cả hoặc không operation nào”
  - Cung cấp tính bền vững
- ♦ Modeling Options
  - Văn bản (scripts)
  - Các thông điệp hiện
- ♦ Error conditions
  - Có thể đòi hỏi các interaction diagrams riêng biệt
  - Rollback
  - Failure modes

# Tích hợp các cơ chế kiến trúc: Persistency

- ◆ Bảng ánh xạ các Analysis-Class với các cơ chế kiến trúc có từ bước phân tích Use-Case

Analysis Class	Analysis Mechanism(s)	
Student	<i>Persistency</i> , Security	<i>OODBMS</i> <i>Persistency</i>
Schedule	<i>Persistency</i> , Security	
CourseOffering	Persistency, Legacy Interface	<i>RDBMS</i> <i>Persistency</i>
Course	Persistency, Legacy Interface	
RegistrationController	Distribution	

*Legacy Persistency (RDBMS )  
deferred to Subsystem Design*

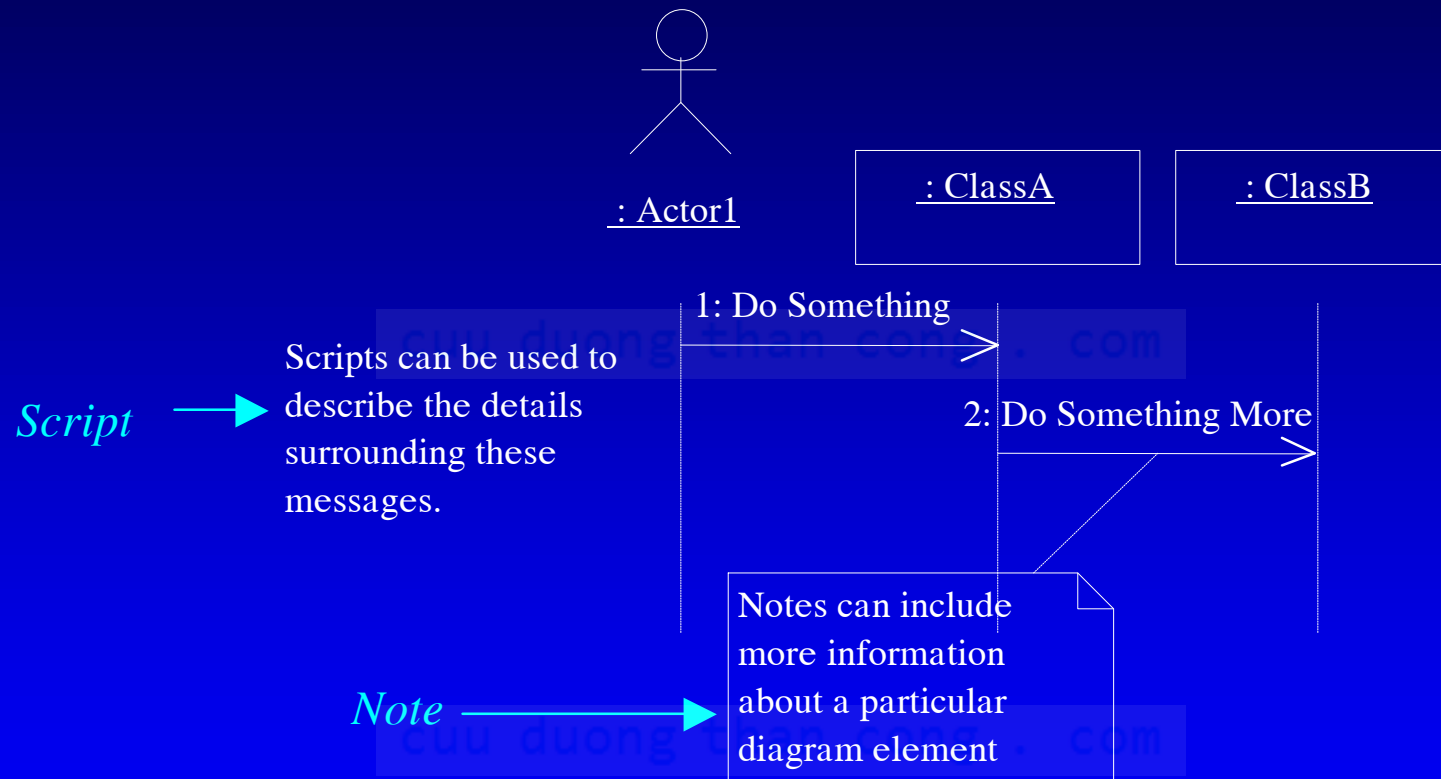
# Các bước thiết kế Use-Case

---

- ◆ Mô tả tương tác giữa các Design Object
- ◆ Đơn giản hóa các Interaction Diagram nhờ vào các Subsystem (optional)
- ◆ Mô tác các hành vi liên quan đến tính Persistence
- ★ ◆ Tinh chỉnh mô tả về các Flow of Events
- ◆ Hợp nhất các Class và các Subsystem
- ◆ Checkpoints

# Detailed Flow of Events Description Options

## ◆ Annotate the interaction diagrams



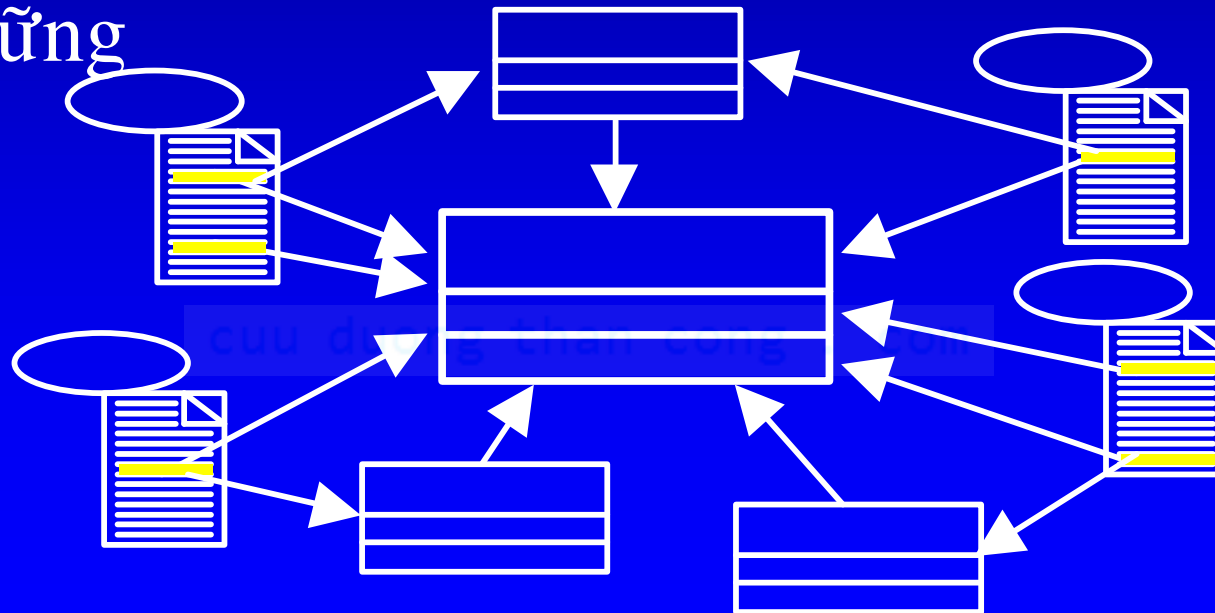
# Các bước thiết kế Use-Case

---

- ◆ Mô tả tương tác giữa các Design Object
- ◆ Đơn giản hóa các Interaction Diagram nhờ vào các Subsystem (optional)
- ◆ Mô tác các hành vi liên quan đến tính Persistence
- ◆ Tinh chỉnh mô tả về các Flow of Events
- ★ ◆ Hợp nhất các Class và các Subsystem
- ◆ Checkpoints

# Design Model Unification Considerations

- ◆ Tên của các phần tử mô hình phải diễn tả được chức năng của chúng
- ◆ Trộn các phần tử giống nhau
- ◆ Dùng phép kế thừa với các phần tử trừu tượng
- ◆ Giữ cho model elements và flows of events bền vững



# Các bước thiết kế Use-Case

---

- ◆ Mô tả tương tác giữa các Design Object
- ◆ Đơn giản hóa các Interaction Diagram nhờ vào các Subsystem (optional)
- ◆ Mô tác các hành vi liên quan đến tính Persistence
- ◆ Tinh chỉnh mô tả về các Flow of Events
- ◆ Hợp nhất các Class và các Subsystem
- ★ ◆ Checkpoints

# Checkpoints: Design Model

---

- ◆ Việc chia thành package/subsystem có hợp lý và bền vững?
- ◆ Tên của các packages/subsystems có gợi nhớ?
- ◆ Các public package class and các subsystem interface có cung cấp một tập các dịch vụ duy nhất và bền vững hợp lý?
- ◆ Các phụ thuộc giữa các package/subsystem có tương ứng với quan hệ giữa các class chứa bên trong không?
- ◆ Các class chứa trong package có phù hợp với tiêu chí phân chia thành package?
- ◆ Có thể tách package/subsystem thành hai?
- ◆ Tỷ lệ các packages/subsystems và số lượng các class có hợp lý không?

# Checkpoints: Use-Case Realizations

---

- ◆ Tất cả các luồng chính và sub-flows trong vòng lặp này đã xử lý chưa?
- ◆ Tất cả các hành vi đã phân bổ cho các phần tử thiết kế chưa?
- ◆ Việc phân bổ này có chính xác không?
- ◆ Nếu có vài interaction diagrams dành cho use-case realization, việc xác định collaboration diagrams nào liên quan đến flow of events nào có dễ dàng không?

# Nhắc lại: Use-Case Design

---

- ◆ Mục tiêu của Use-Case Design là gì?
- ◆ Việc đóng gói các subsystem interaction có ý nghĩa gì ? Tại sao đây là việc hữu ích?

cuu duong than cong . com

cuu duong than cong . com

# Bài tập: Use-Case Design, Part 1

- ◆ Thực hiện các việc sau:
  - Analysis use-case realizations (VOPCs and interaction diagrams)
  - The analysis-class-to-design-element map
  - *The analysis-class-to-analysis-mechanism map*
  - *Analysis-to-design-mechanism map*
  - *Patterns of use for the architectural mechanisms*

cuu duong than cong . com

# Bài tập: Use-Case Design, Part 1 (cont.)

- ◆ Identify the following for a particular use case:
  - The design elements that replaced the analysis classes in the analysis use-case realizations
  - *The architectural mechanisms that affect the use-case realizations*
  - The design element collaborations needed to implement the use case
  - The relationships between the design elements needed to support the collaborations

*(continued)*

# Bài tập: Use-Case Design, Part 1 (cont.)

- ◆ Produce the following for a particular use case:
  - Design use-case realization
    - Interaction diagram(s) per use-case flow of events that describes the DESIGN ELEMENT collaborations required to implement the use case
    - Class diagram (VOPC) that includes the DESIGN ELEMENTS that must collaborate to perform the use case, and their relationships

# Bài tập: Use-Case Design, Part 2 (optional)

- ◆ Given the following:
  - The architectural layers, their packages, and their dependencies
  - All design use-case realization VOPCs (design elements, their packages, and their relationships)

*(continued)*

# Bài tập: Use-Case Design, Part 2(optional) (tt.)

- ◆ Identify the following:
  - Any updates to the package relationships needed to support the class relationships
- ◆ Produce the following diagrams:
  - Refined class diagram that contains all packages and their dependencies (organized by layer)

cuu duong than cong . com