基于面向对象的类的设计

Yang YI, Ph.D

issyy@mail.sysu.edu.cn

软件工程

方法

工具

做软件的步骤和制品

软件项目管理

软件测试

面向对象方法论

结构化方法论

基于面向对象的步骤和制品

基于结构化的步骤和制品
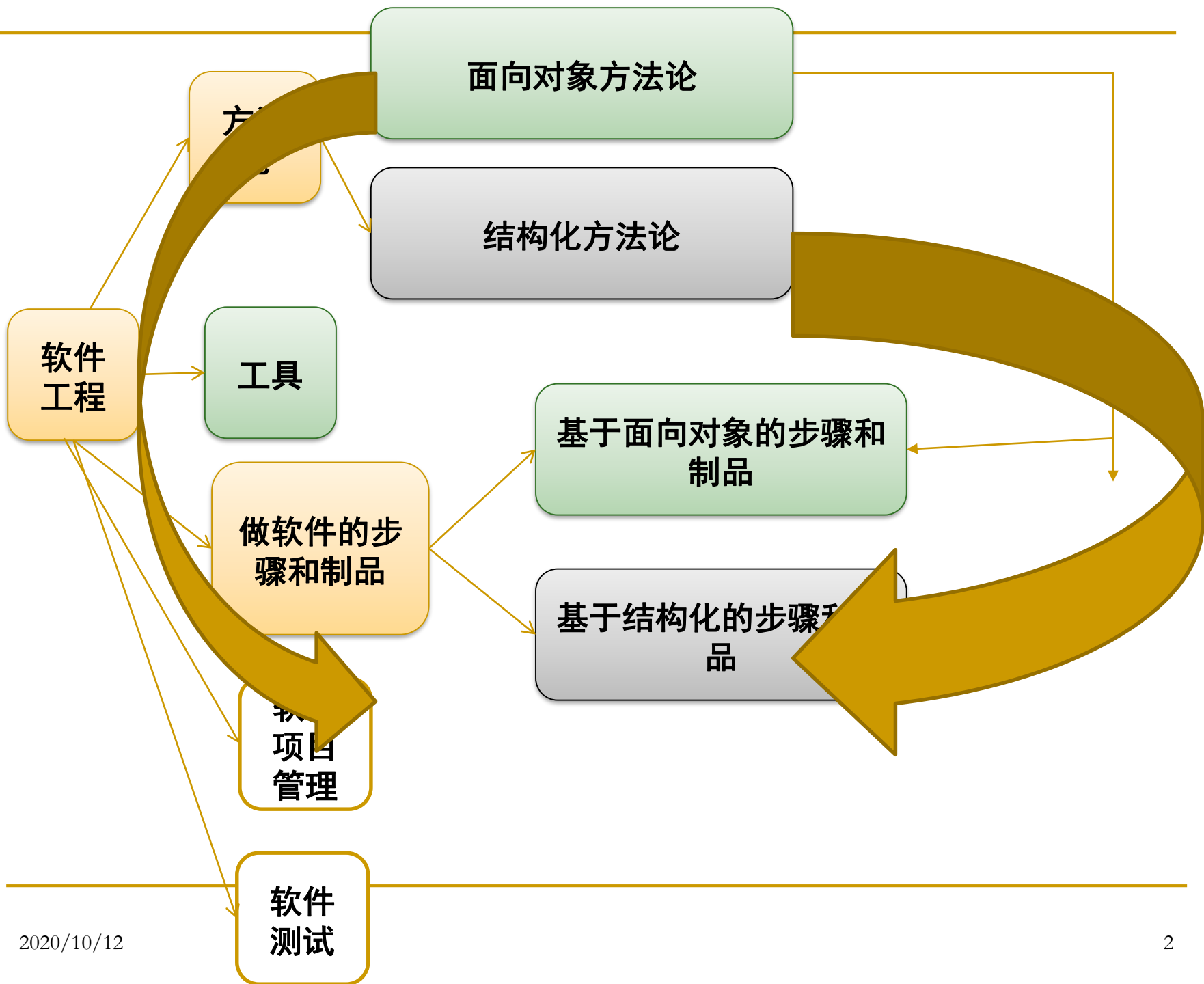
基于面向对象的类的设计

（用例分析）

主要内容

- 用例分析总述

- 补充用例规约

- 查找类

- 将用例行为分配给类

- 描述类

- 描述分析机制

- 合并类

类（用例分析）的设计的步骤

1. 补充用例规约

2. 查找类

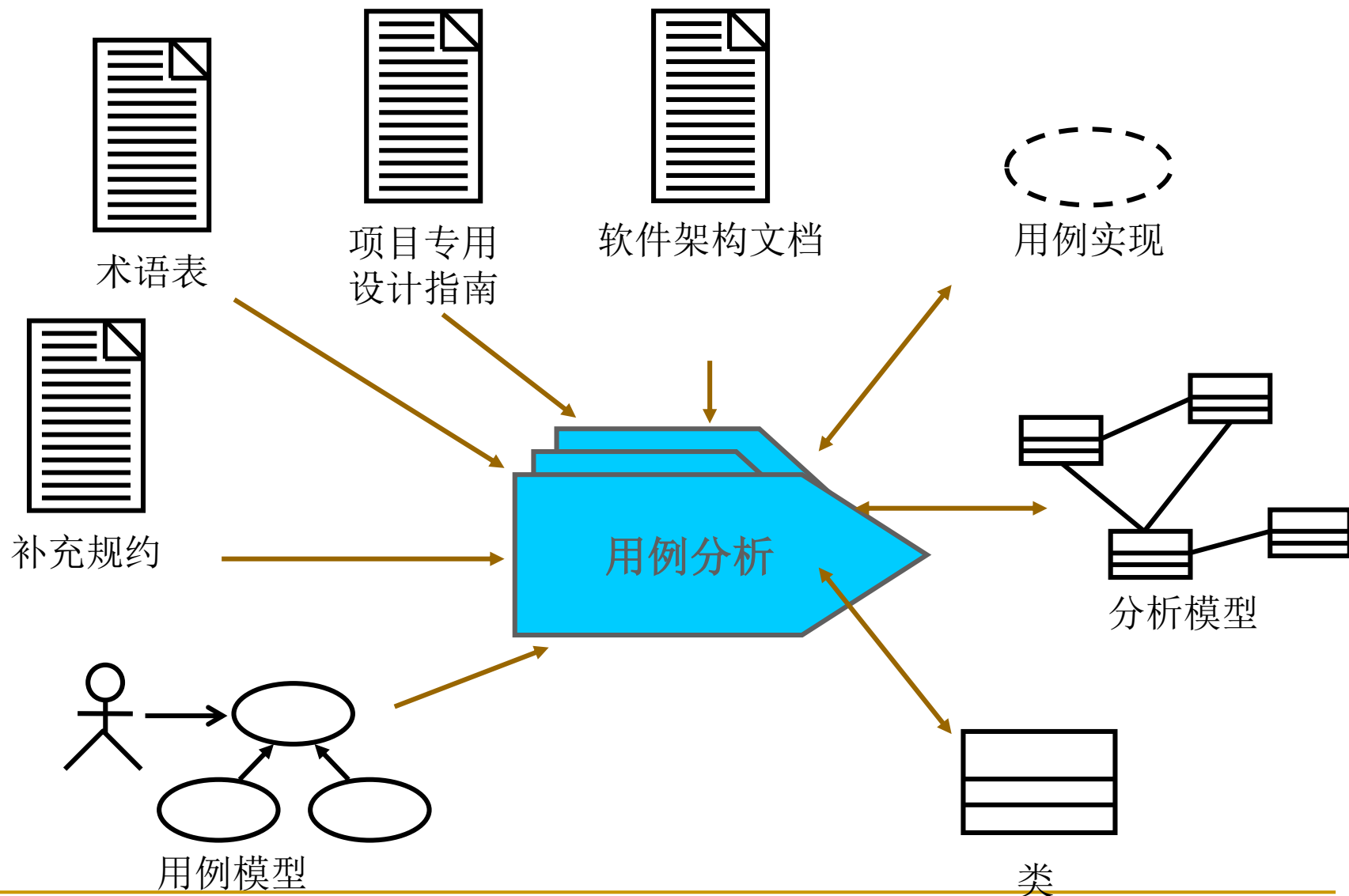3. 将用例行为分配给类

4. 描述类

5. 描述分析机制

6. 合并类

Objectives: Use-Case Analysis

- Explain the purpose of Use-Case

- Analysis and where in the lifecycle it is performed

- Identify the classes which perform a usecase flow of events

- Distribute the use-case behavior to those classes, identifying responsibilities of the classes

- Develop Use-Case Realizations that model the collaborations between instances of the identified classes

总揽

基于面向对象的类的设计（用例分析）

术语表

项目专用
设计指南

软件架构文档

用例实现

补充规约

用例分析

分析模型

用例模型

类

第八章 用例分析

主要内容

用例分析总述

补充用例规约

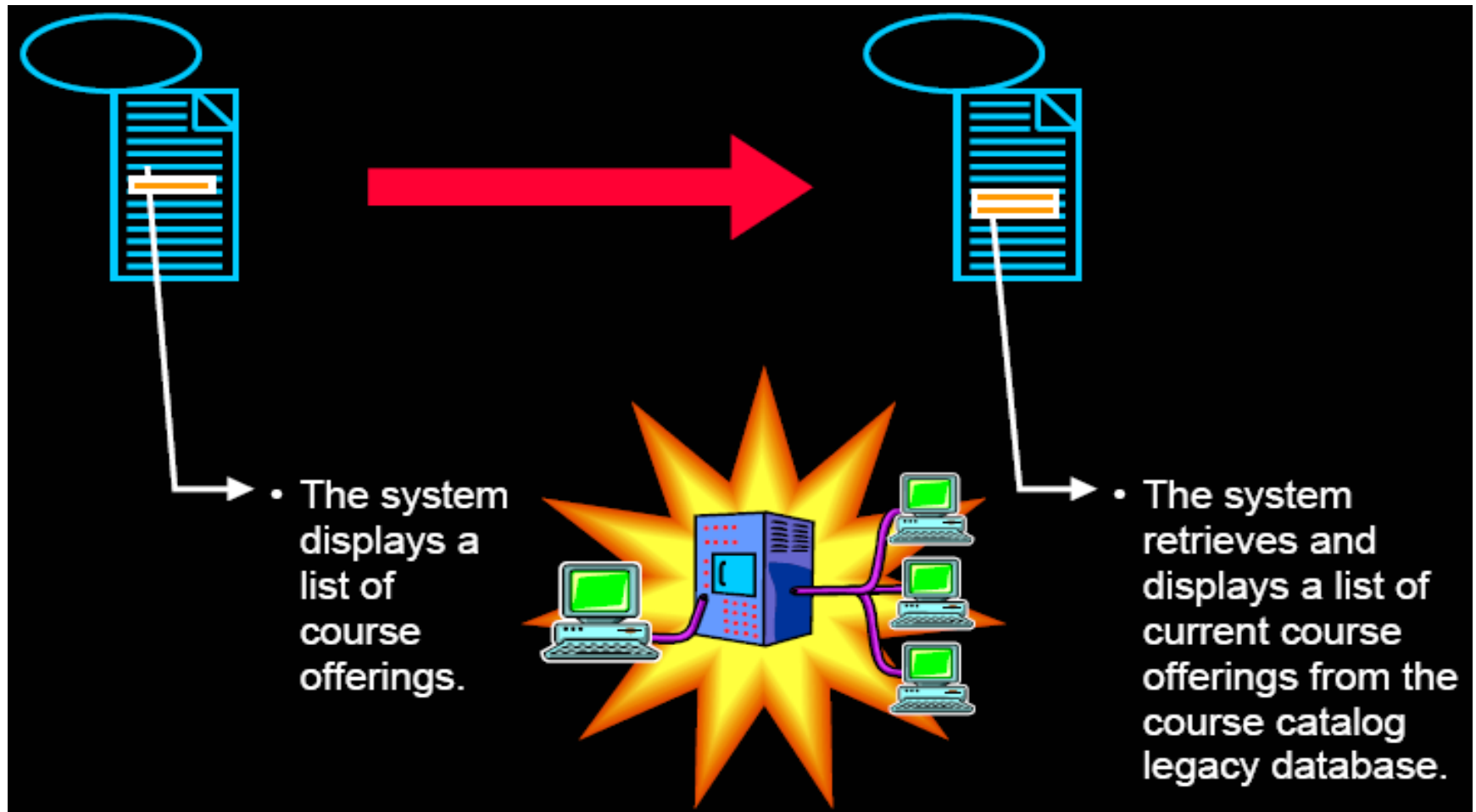查找类
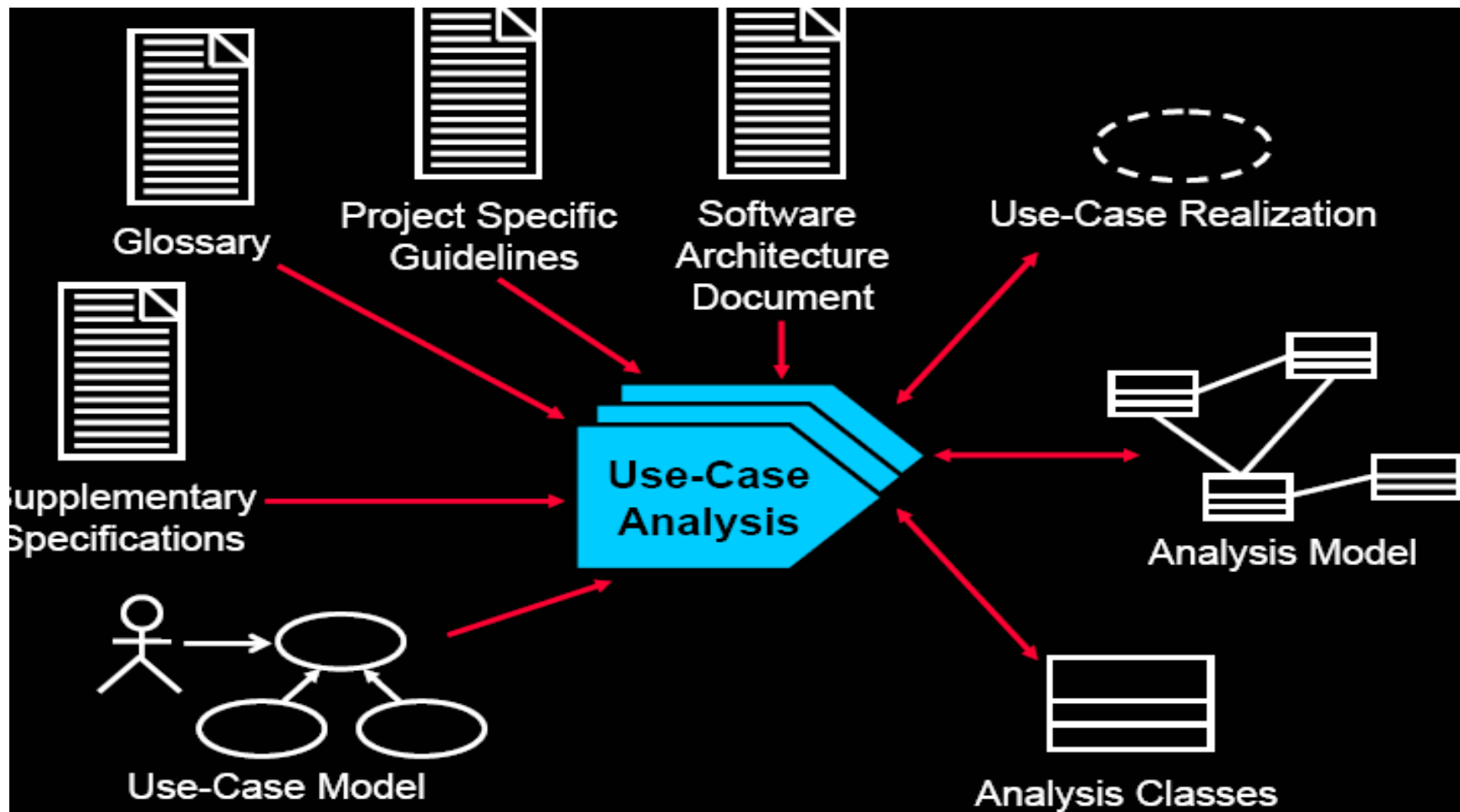
将用例行为分配给类

描述类

描述分析机制

合并类
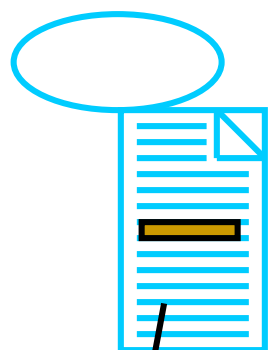
案例实践

Use-Case Analysis Steps

- Supplement the Use-Case Description
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify classes
- Checkpoints

Supplement the Use-Case Description

- The system displays a list of course offerings.

- The system retrieves and displays a list of current course offerings from the course catalog legacy database.
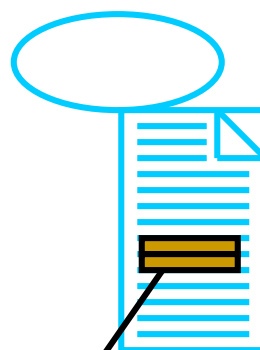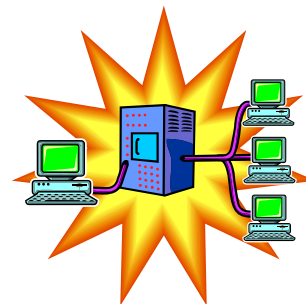
补充用例规约

步骤1：修改完善需求分析中的用例规约

- 在选课系统中，学生点击"创建课程表"
- 系统显示课程列表

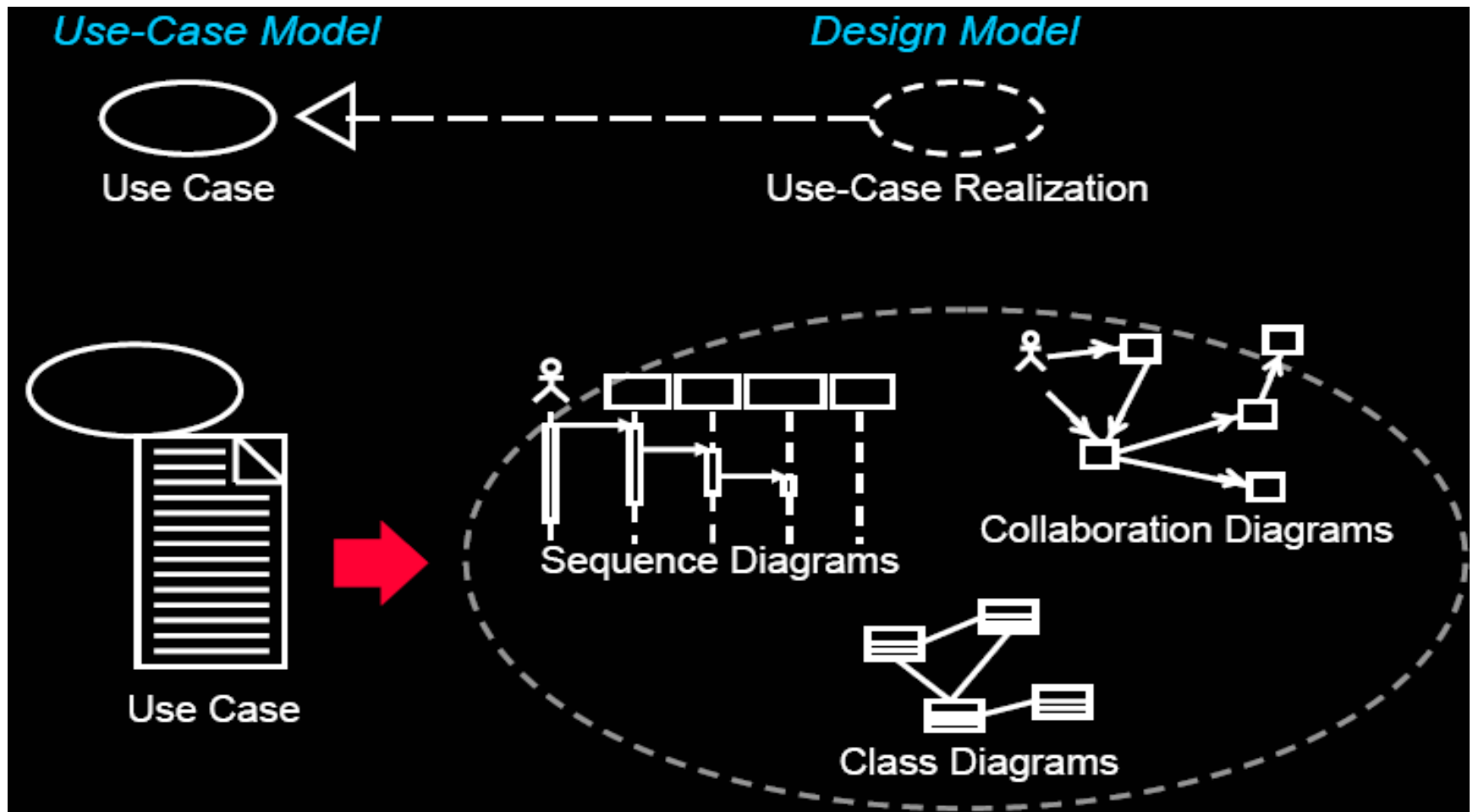- 应该：系统找到并列出课程目录数据库（遗留系统）提供的课程列表，供学生选择

上面这样写，可以吗？
不可以！

补充用例规约，到底是啥意思

- 需要打开前面的需求分析，对**所有用例规约**进行审核

- 看看，是否有某些输入、输出，没有写全

- 是否，有某些操作，是"拐弯抹角"的，需要详细写出流程

- 以便**程序员能够理解并且写出完整的代码**
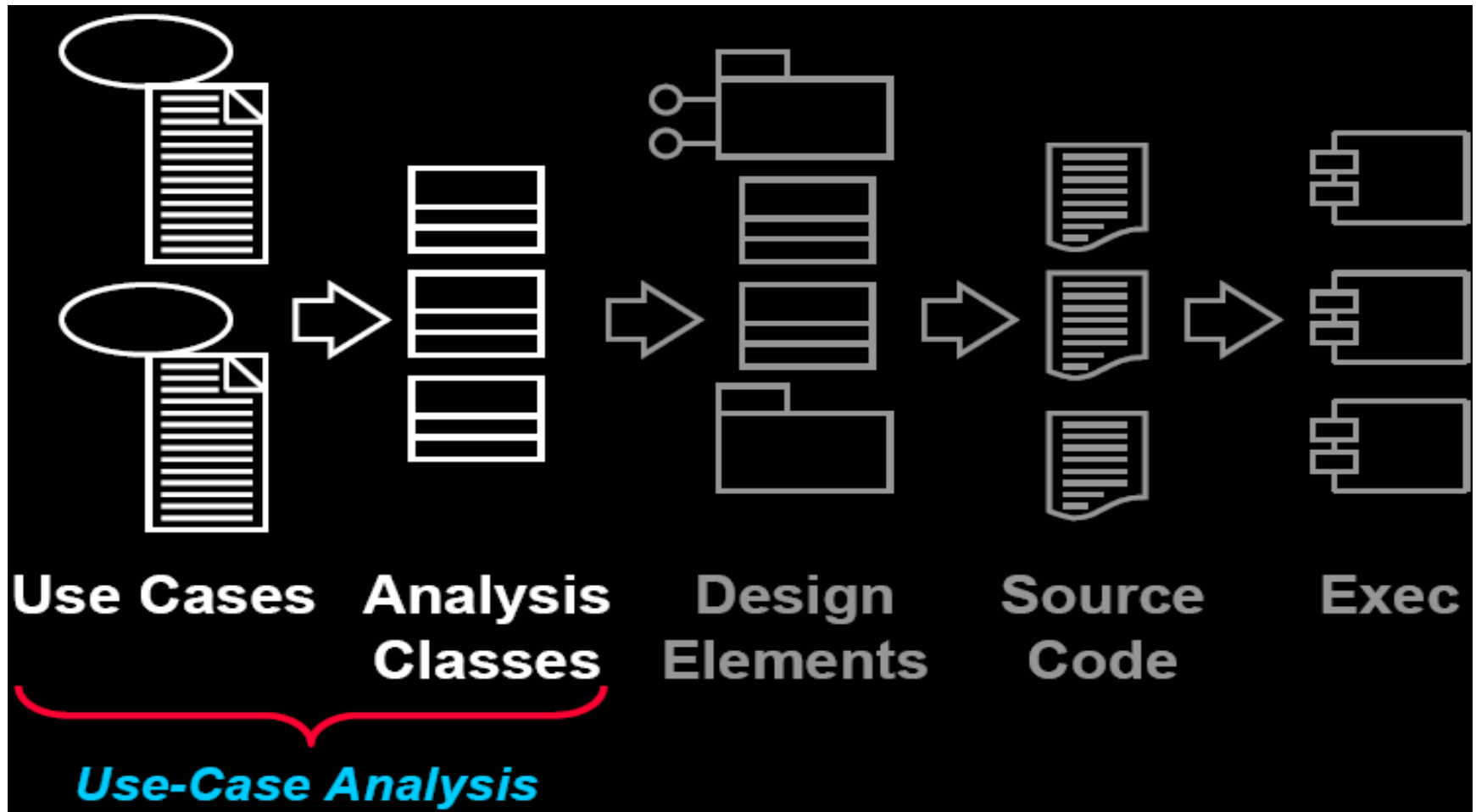
- 语言描述要简洁、清晰

Use-Case Analysis Steps

- Supplement the Use-Case Description
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify classes
- Checkpoints

Review: Use-Case Realization

classes: A First Step Toward Executables

Use Cases → **Analysis Classes** → **Design Elements** → **Source Code** → **Exec**

Use-Case Analysis

类：可执行系统的第一步

用例   类   设计元素   源代码   可执行系统
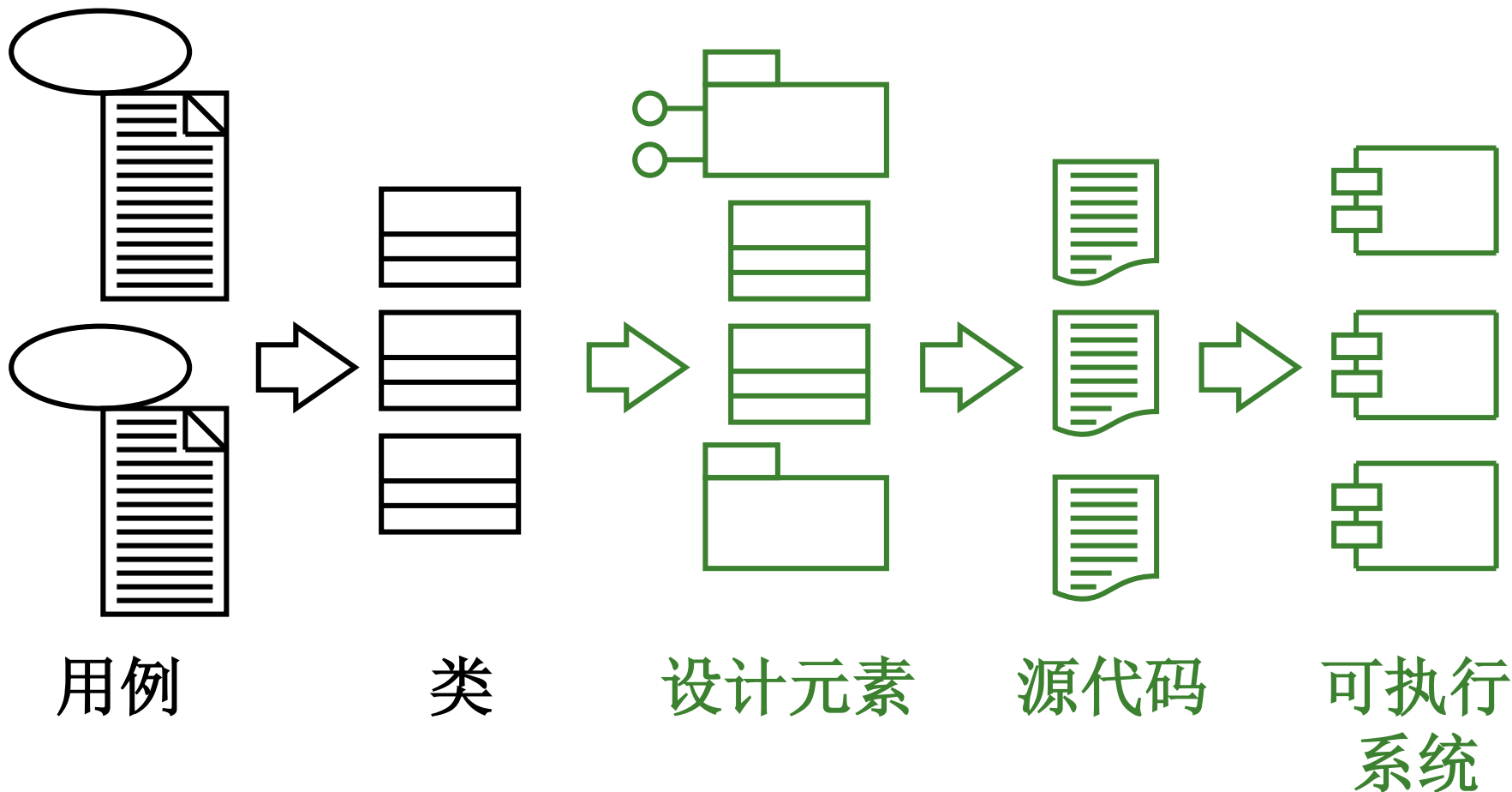
用例分析

类的构造型（sterotype）

- UML中，类的符号是一至三个矩形构成

- 在MVC框架下，为了进一步区分View、Controller、Model里面的类，一目了然，分别给它们命名并且赋予了新的图形符号

Find Classes from Use-Case Behavior

The complete behavior of a use case has to be distributed to classes

从用例行为中查找类

边界类

系统边界

实体类

系统信息

控制类

协调用例行为

```
┌─────────────────────────────┐
│              ` │            │
│           View              │
│                             │
│                             │
│                             │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│                 │           │
│        Controller           │
│                             │
│                             │
│                             │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│              │              │
│        Model                │
│                             │
│                             │
│                             │
│                             │
└─────────────────────────────┘
```
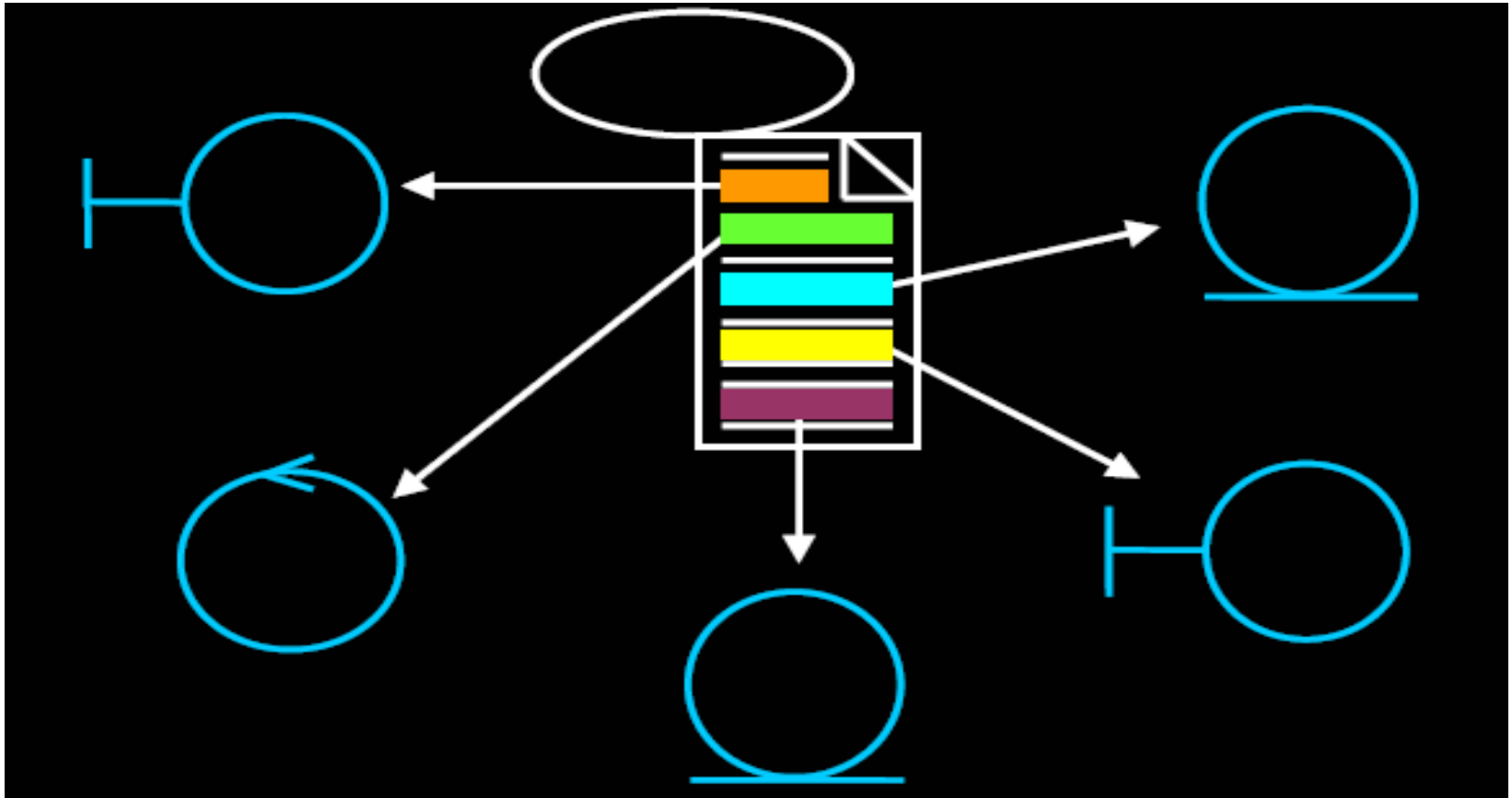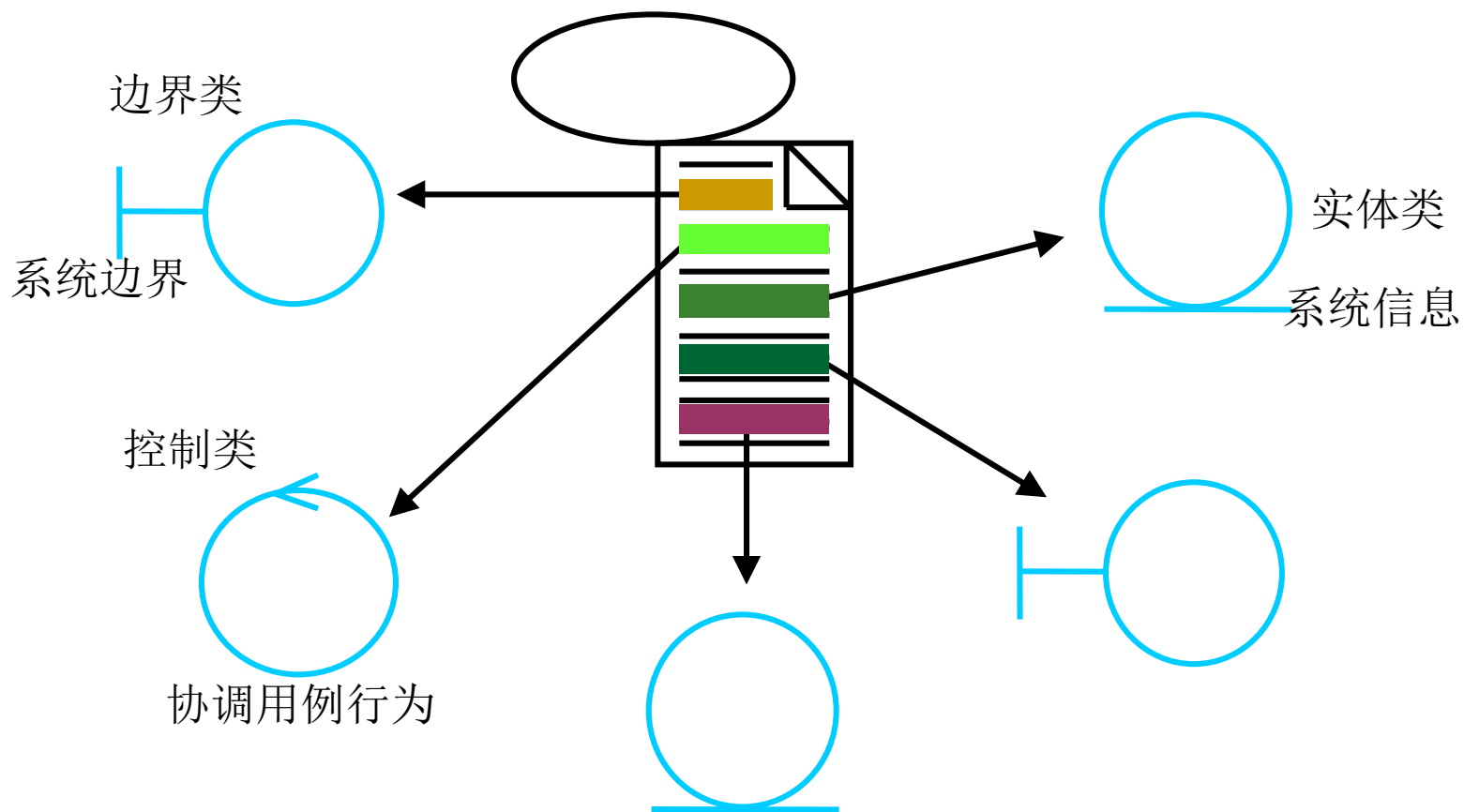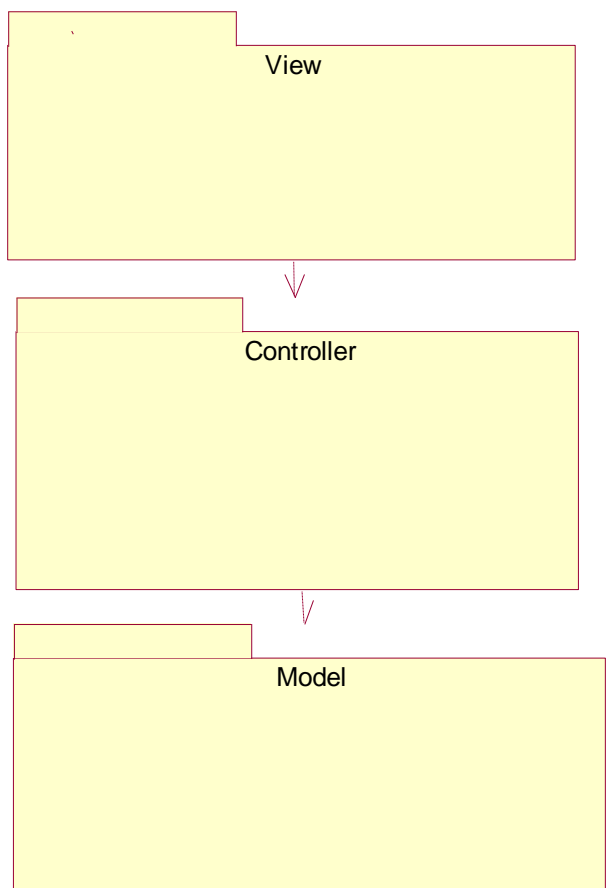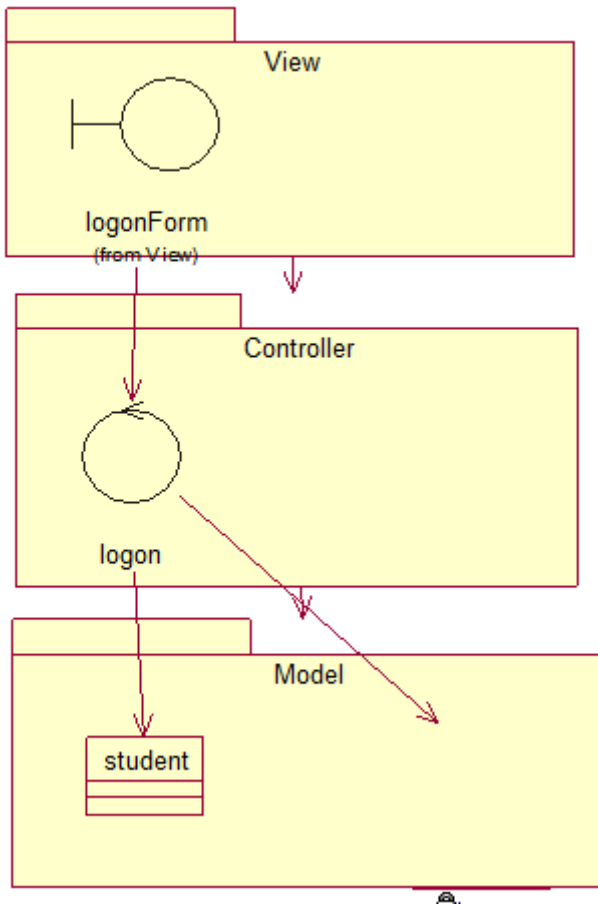
> **架构设计是一个迭代然后才能完成的工作；**

> **至少包含两次迭代；**

> **架构设计的第一次迭代被称为 Architecture Analysis，第二次以及后续的修改完善被称为 Architecture Design**

> **左图是Architecture Analysis 的结果**

> **左图显示了类设计之后，将架构设计进一步完善的情况**
> **此时的架构设计可以称为 Architecture Design**
> **Thus,架构设计的初始情况是染过层由空的包构成的结构，各个包里面是空的，因为还没有进行类的设计**

Boundary Class

- **Inter mediates between the interface and something outside the system**

- **Several Types**
    - **User interface classes**
    - **System interface classes**
    - **Device interface classes**

- *One boundary class per actor/use-case pair*

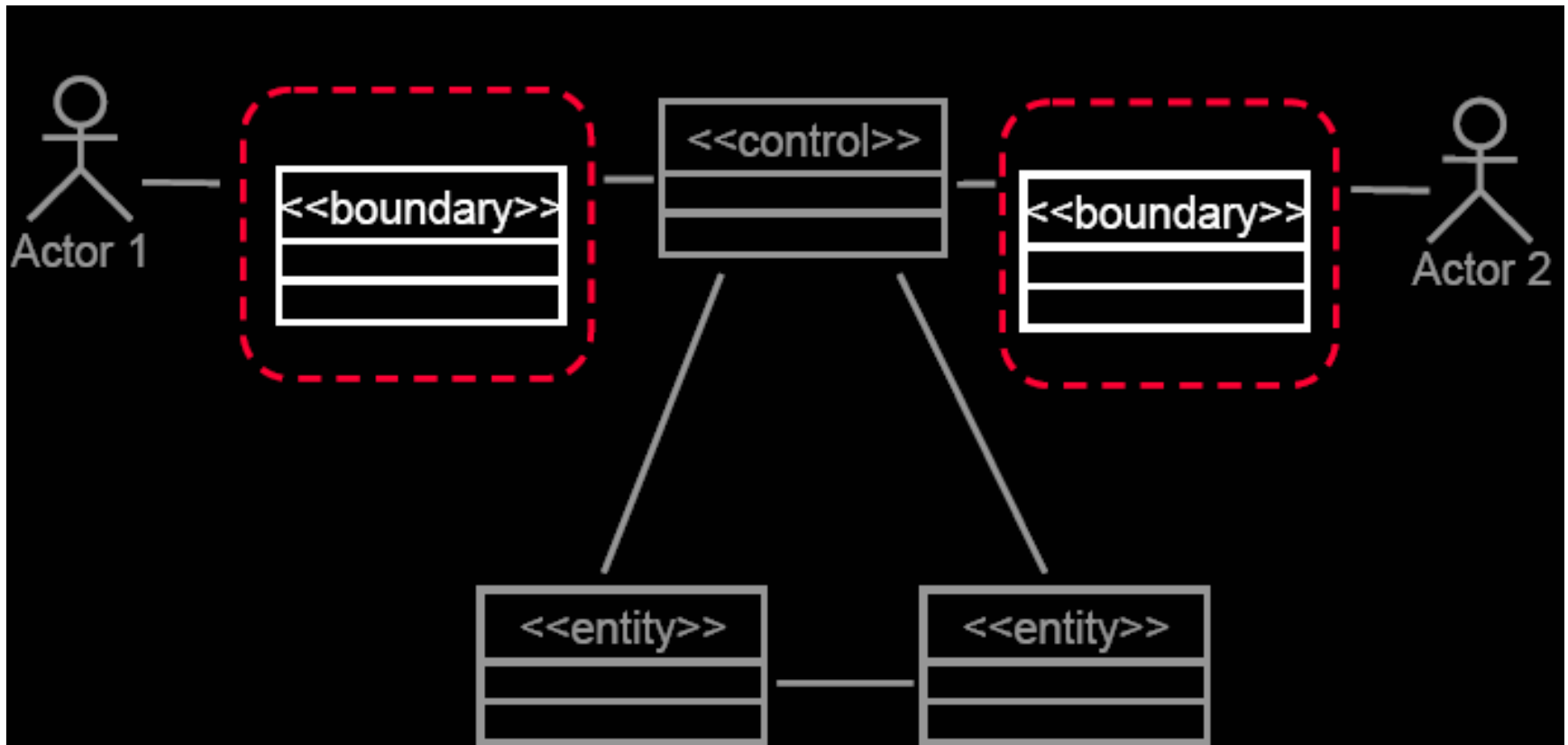*Analysis class stereotype* →

Environment Dependent

边界类

- **提供了对参与者或外部系统交互协议的接口**
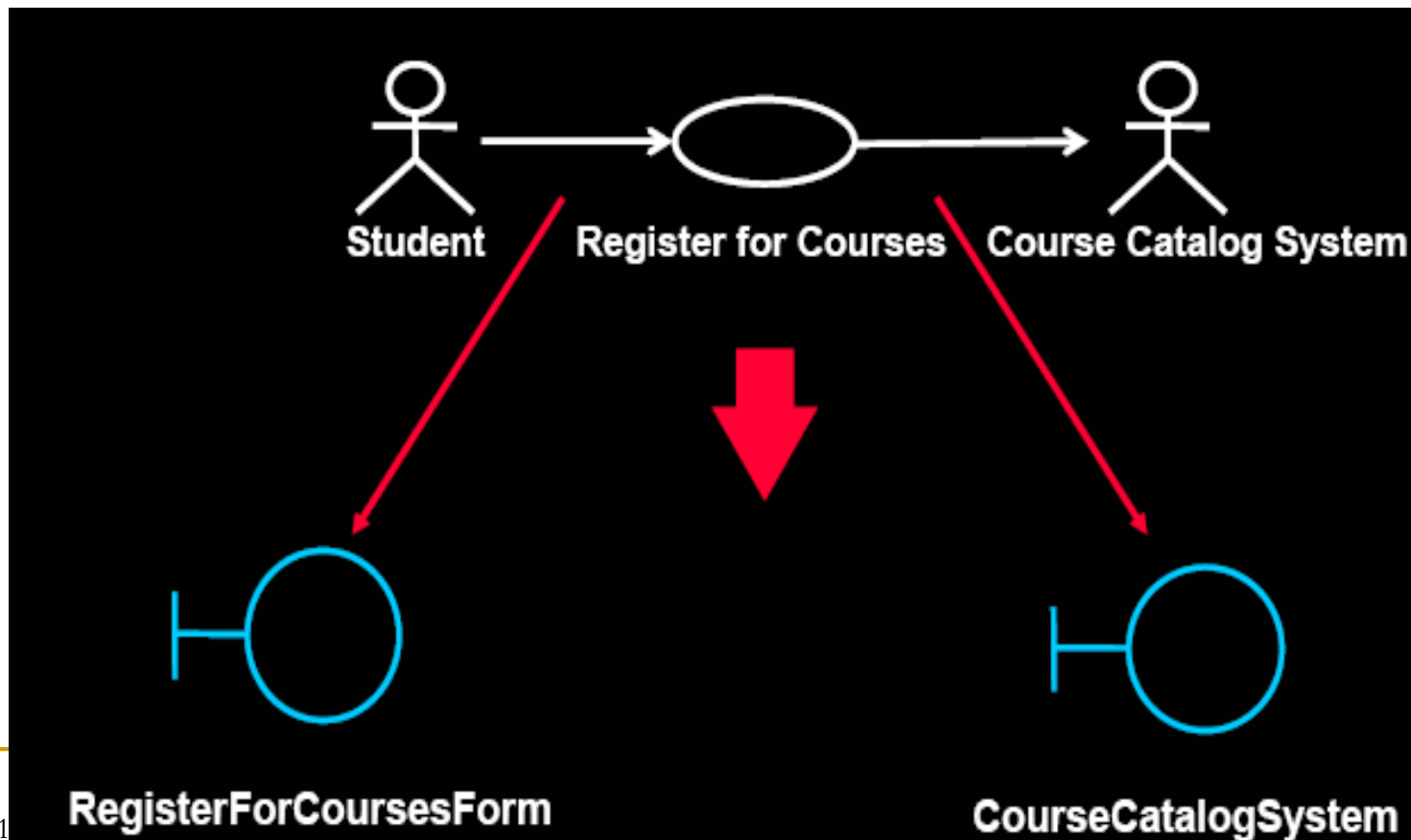 - 如用户界面，http://protocols
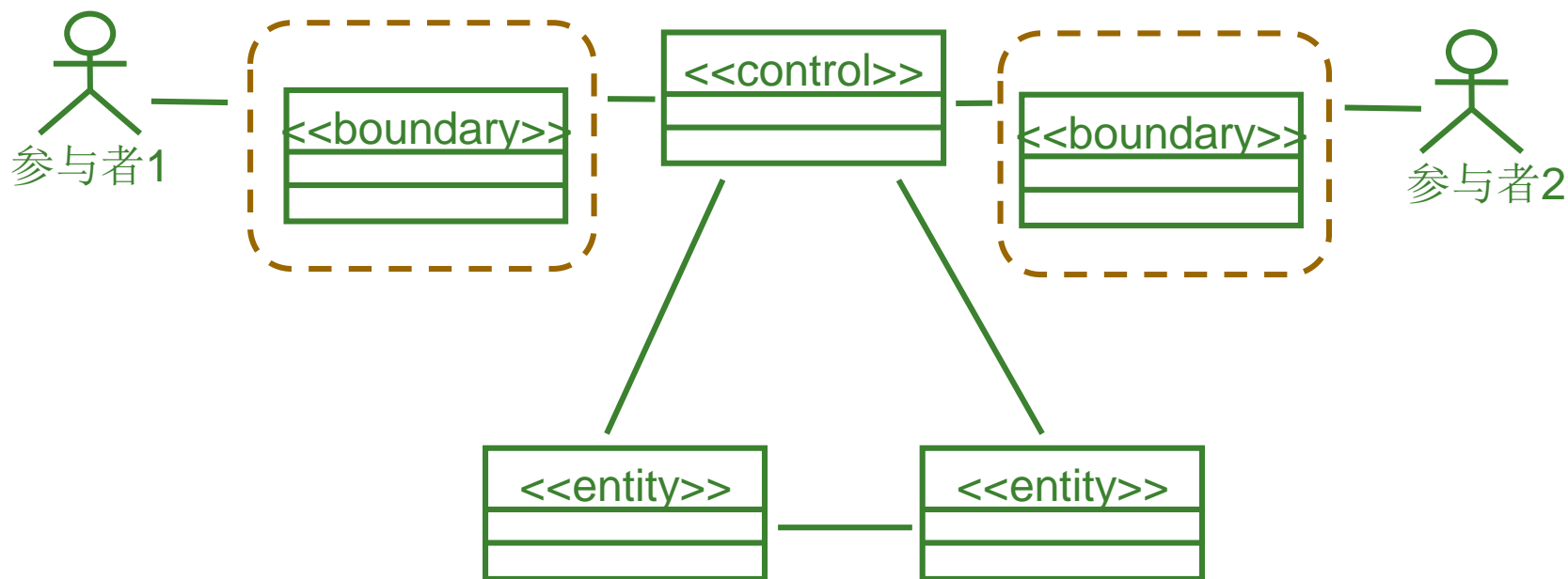
The Role of a Boundary Class

Model interaction between the system and its environment
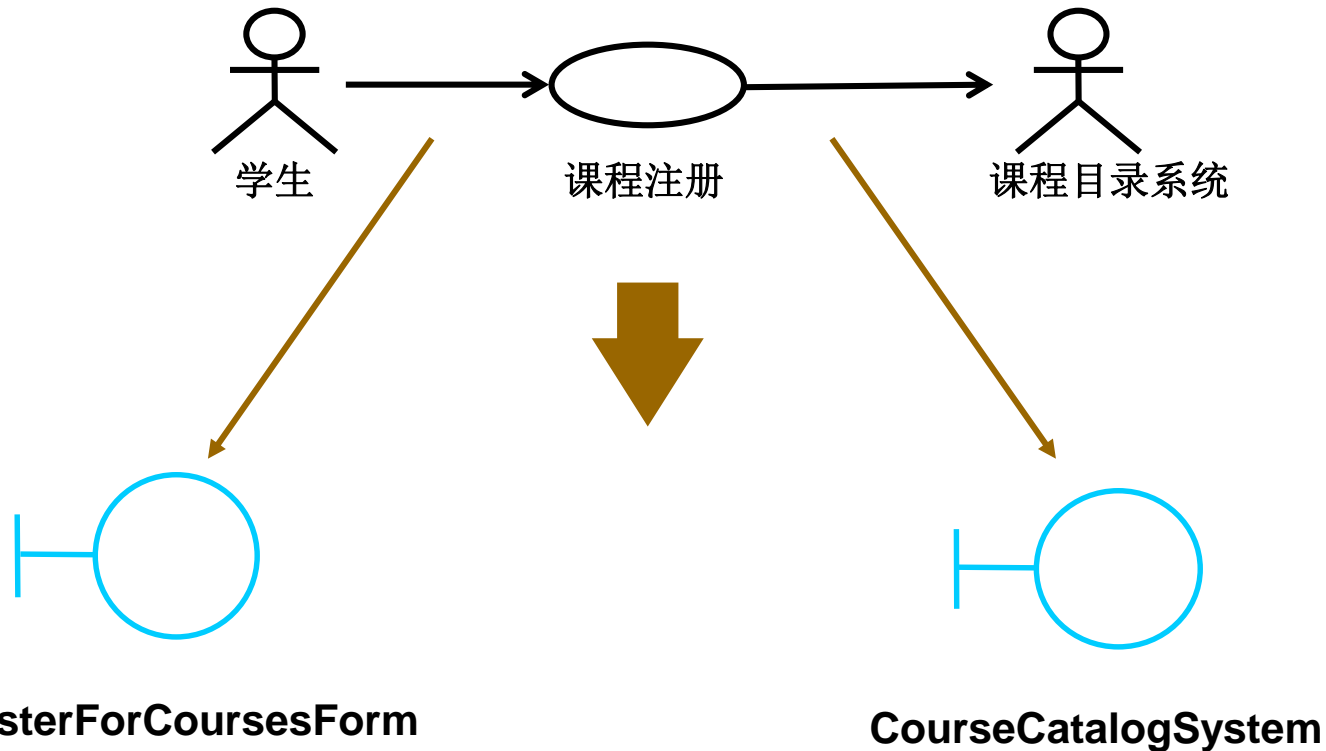
Example: Finding Boundary Classes

- **One boundary class per actor/use case pair**

如何确定边界类

- **为用例中涉及到的每对参与者/用例设计一个边界类来封装面向这个参与者的接口**

学生 　　　　课程注册　　　　课程目录系统

RegisterForCoursesForm　　　　**CourseCatalogSystem**

Guidelines: Boundary Class

- **User Interface Classes**

  - Concentrate on what information is presented to the user

- **Do NOT concentrate on the UI details**

- **System and Device Interface Classes**

  - Concentrate on what protocols must be defined

  - Do NOT concentrate on how the protocols will be implemented

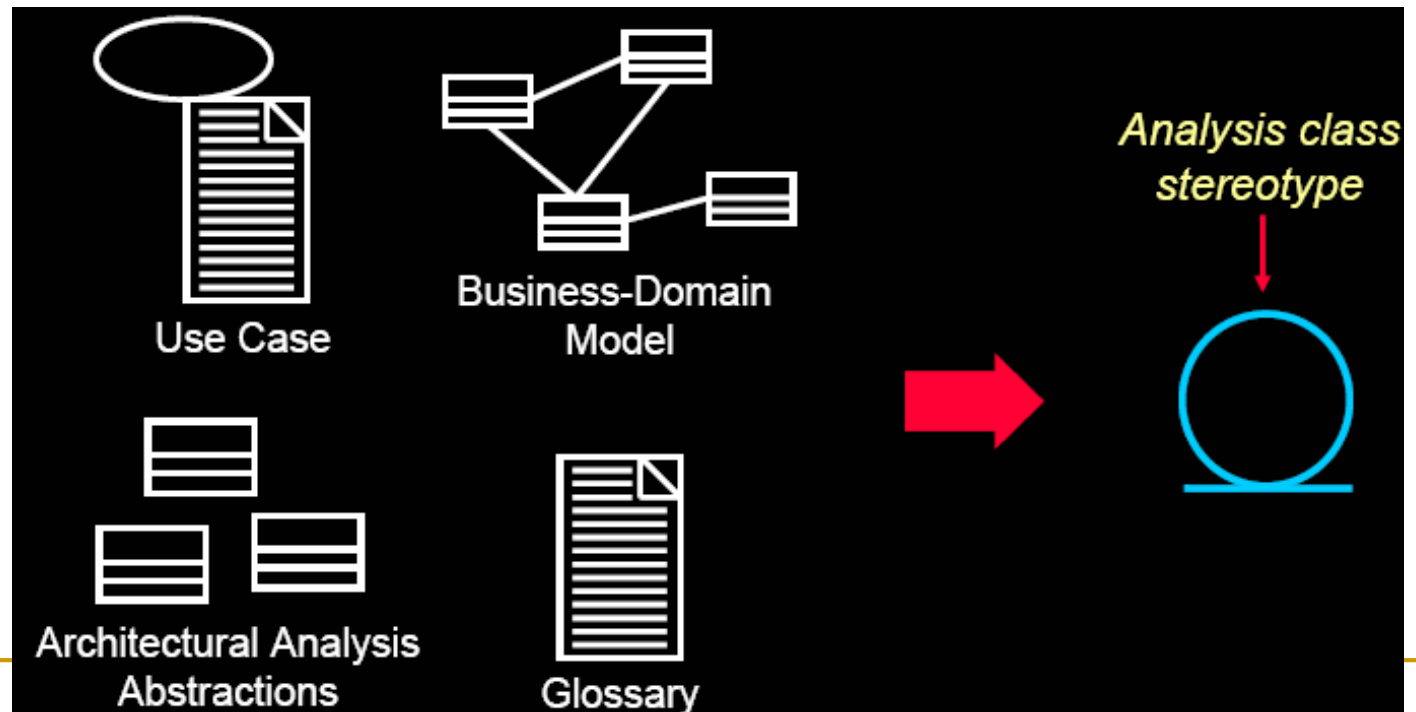■ **边界类的三种类型**

  ❑ **用户接口类**

  ❑ **系统接口类**
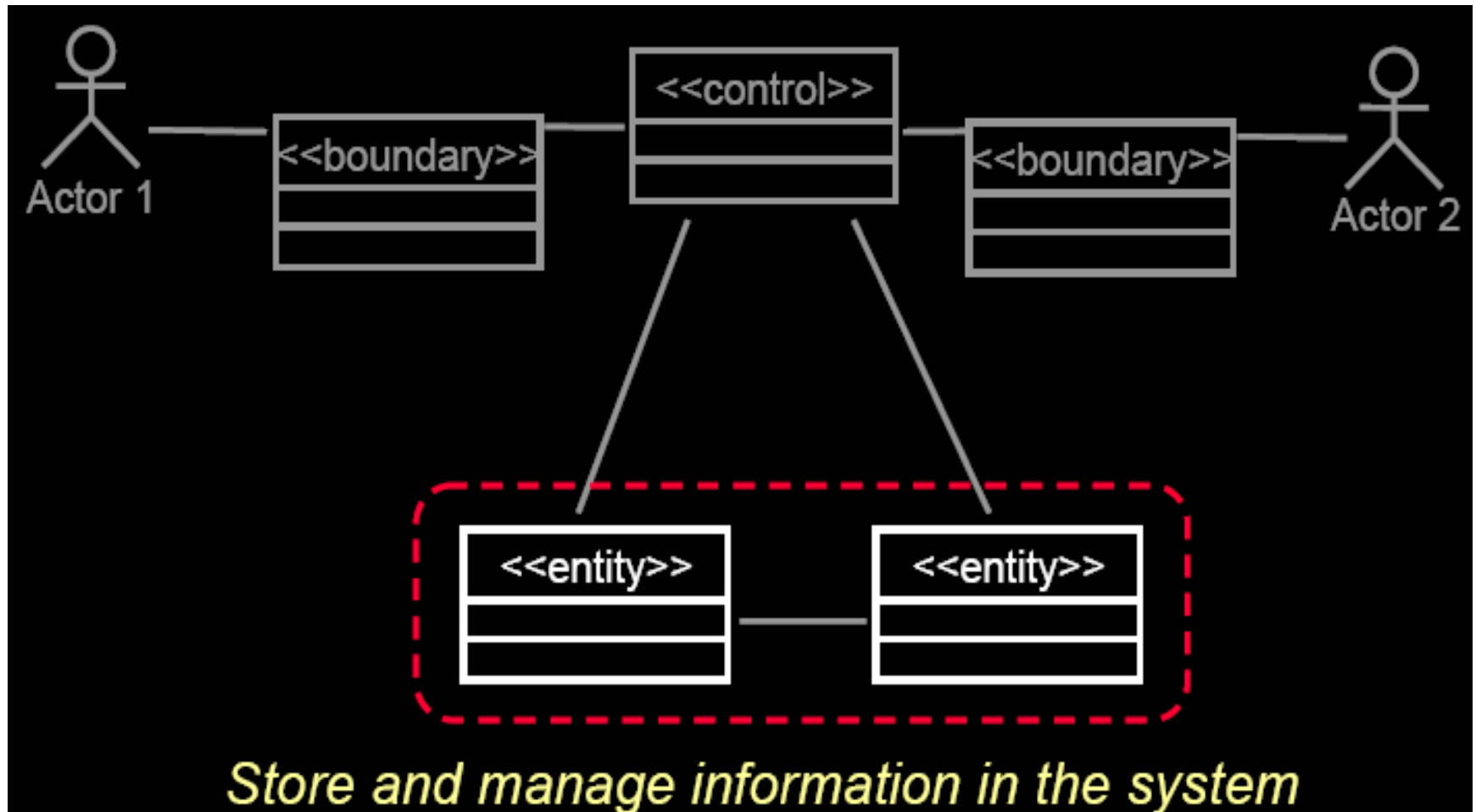
  ❑ **设备接口类**

■ **设计边界类的指导原则**

  ❑ **对于用户接口类，关注于用户界面的交互内容；不是具体窗体构件**

  ❑ **对于系统和设备接口类，关注于定义什么通信协议；不要关注协议的实现细节**

Entity Class

- **Key abstractions of the system**

The Role of an Entity Class

Store and manage information in the system

# Example: Finding Entity Classes

- Use use-case flow of events as input
- Key abstractions of the use case
- Traditional, filtering nouns approach
  - Underline noun clauses in the use-case flow of events
  - Remove redundant candidates
  - Remove vague candidates
  - Remove actors (out of scope)
  - Remove implementation constructs
  - Remove attributes (save for later)
  - Remove operations

Example: Candidate Entity Classes

- **Register for Courses (Create Schedule)**
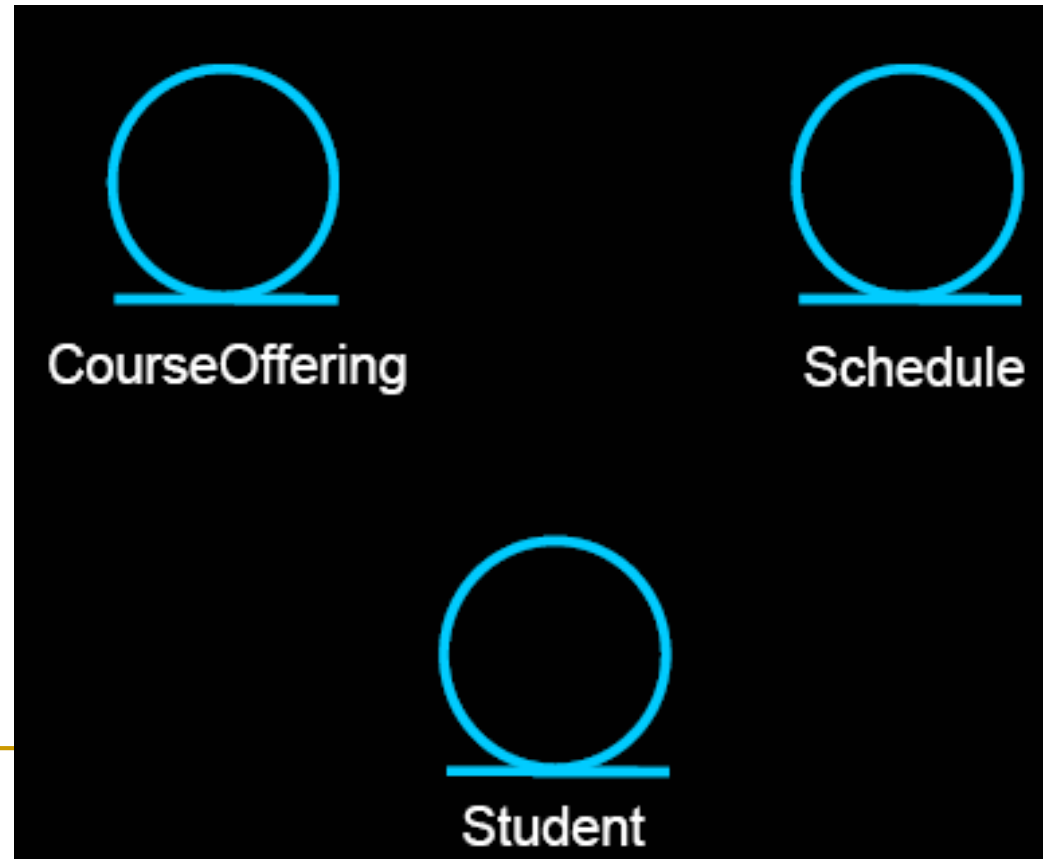
实体类

- **存储（通常具有持久性）一些现象的信息，并包含与这些信息相关的业务规则**
  - 如学生，计划表，课程清单
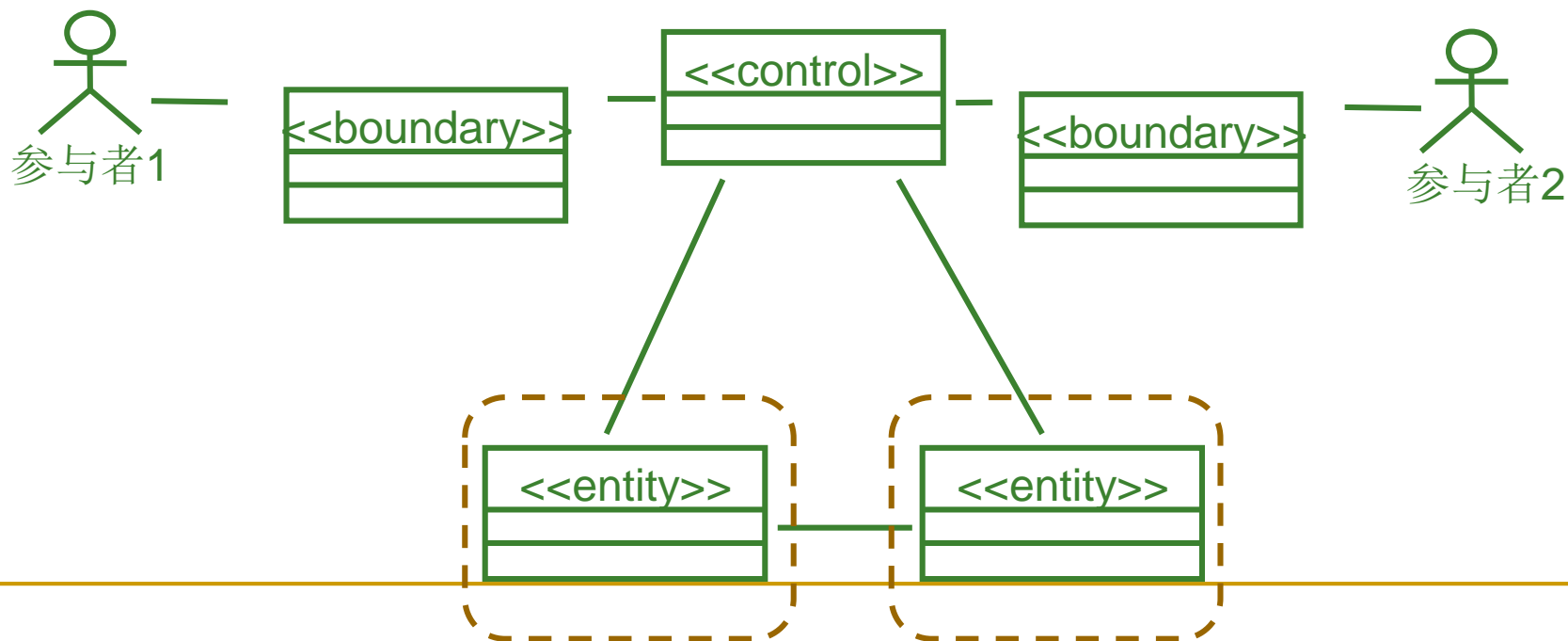
如何确定实体类

- **将用例的事件流作为输入**

- **获取用例的关键抽象**

- **过滤名词的方法：**

  - 对事件流中的名词加下划线

  - 去除冗余的候选名词

  - 去除含义不明确的候选名词

  - 去除参与者

  - 去除实现结构

  - 去除属性

  - 去除操作

■ **课程注册（建立课程表）中的实体类**

课程提供

课程表

学生
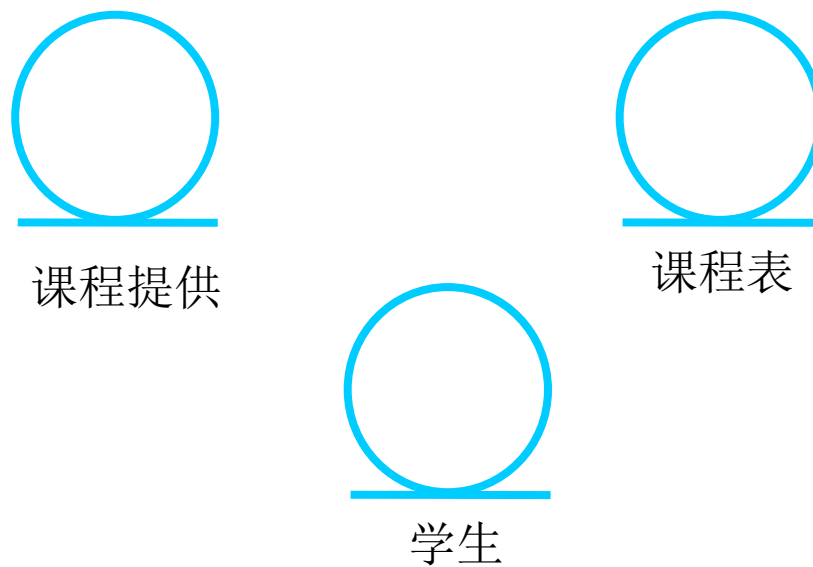
Control Class

- **Use-case behavior coordinator**

- **More complex use cases generally require one or more control cases**
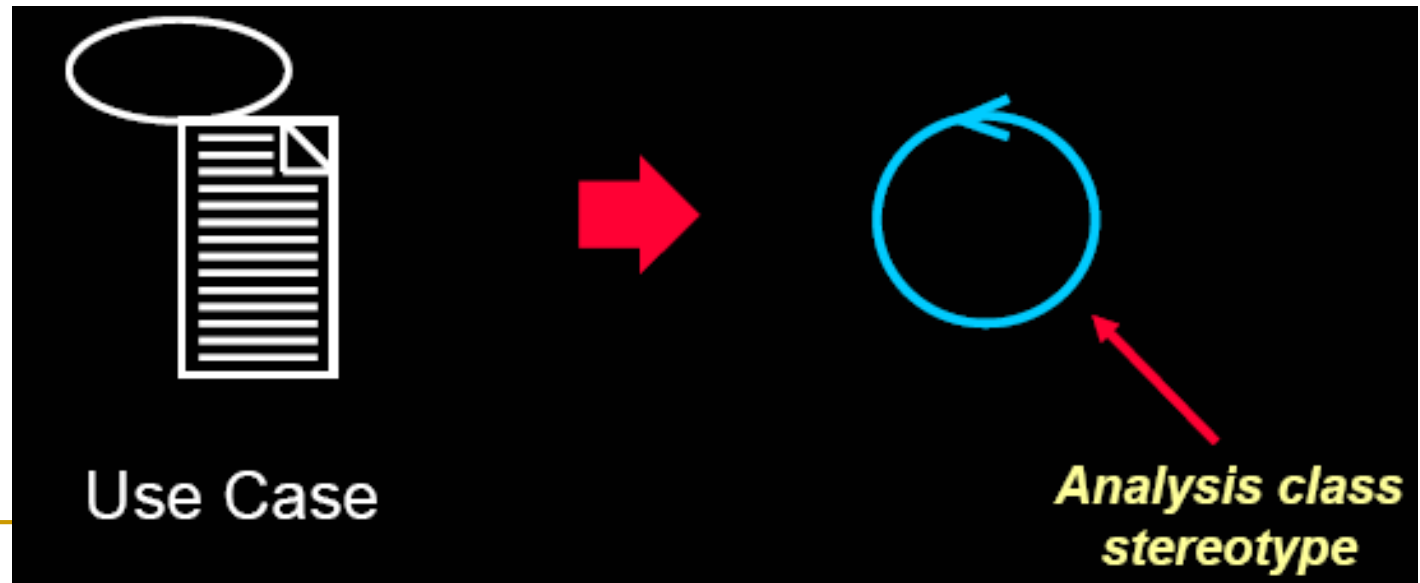
The Role of a Control Class

Coordinate the use-case behavior
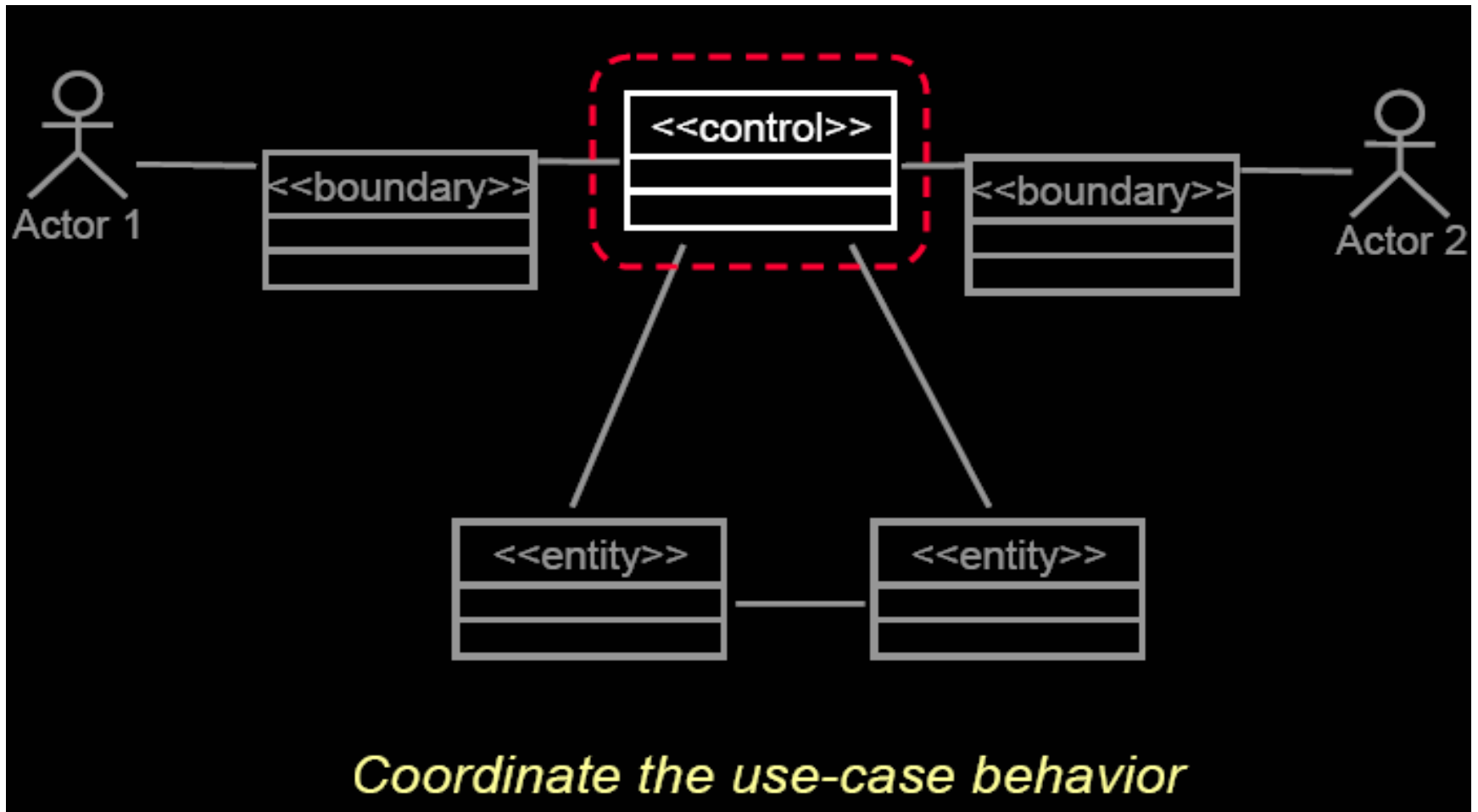
控制类

- **封装一个或多个用例所特有的控制行为**

- **控制类有效地分离了边界对象和实体对象，使系统更能承受系统边界的变更**
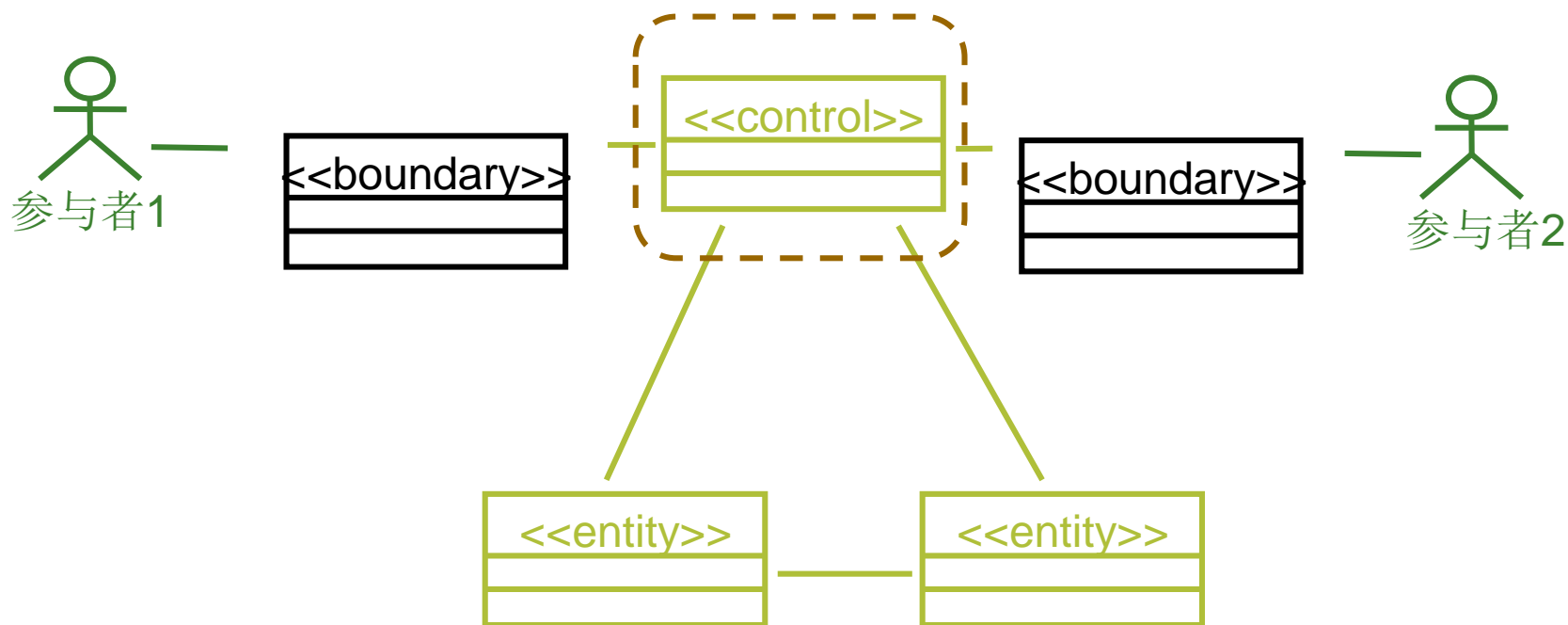
Example: Finding Control Classes

- **In general, identify one control class per use case.**
  - **As analysis continues, a complex use case's control class may evolve into more than one class**
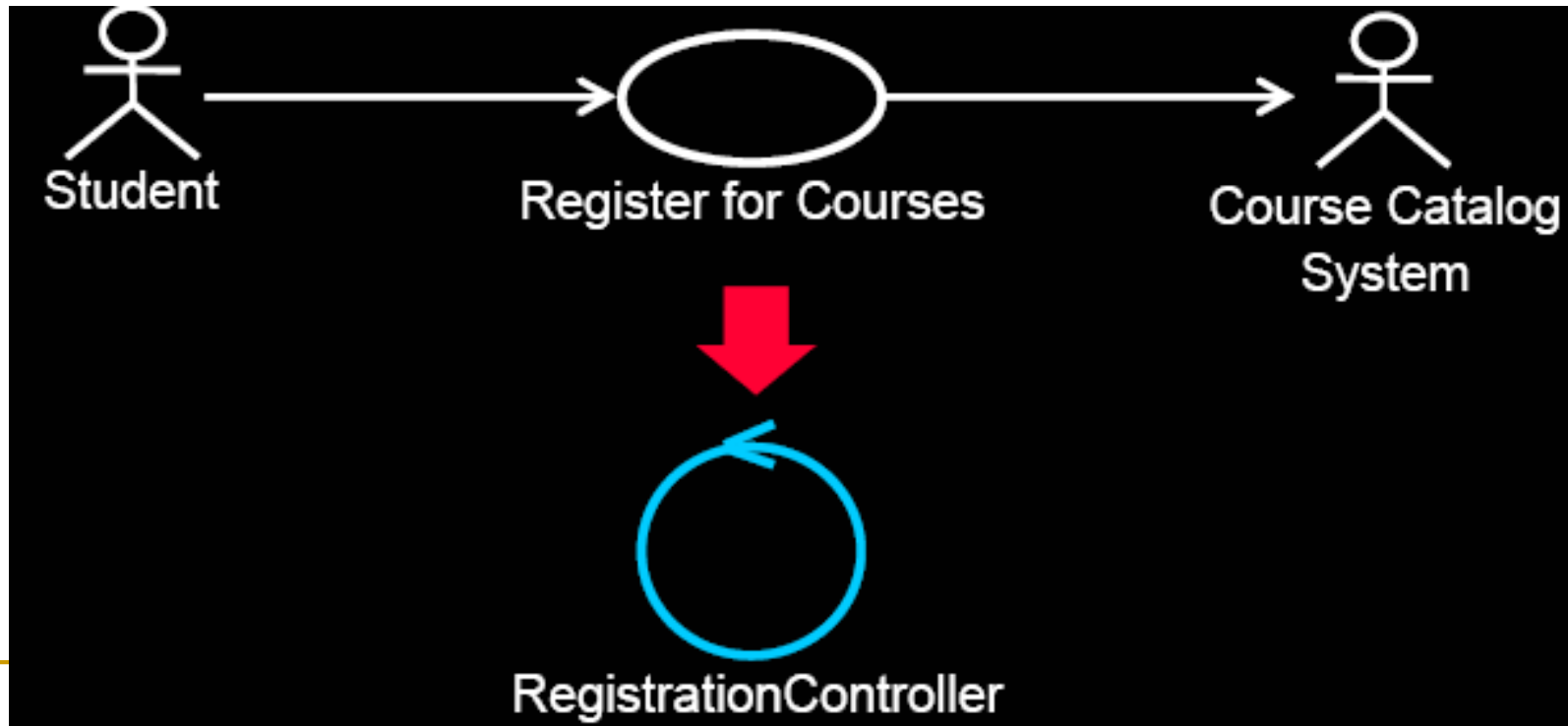
如何确定控制类

- **通常为每个用例设计一个控制类，封装这个用例的顺序**

学生　　　　　　　课程注册　　　　　课程目录系统
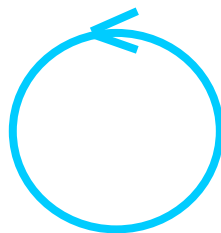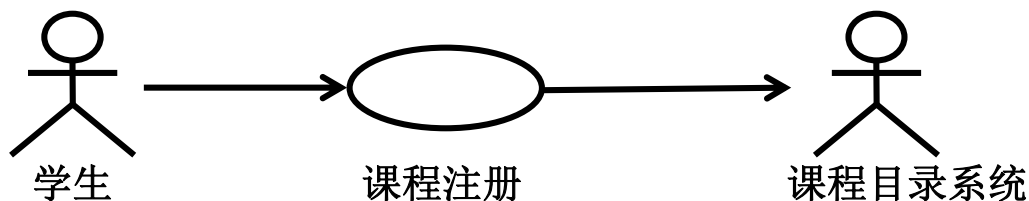
RegistrationController

Example: Summary: classes
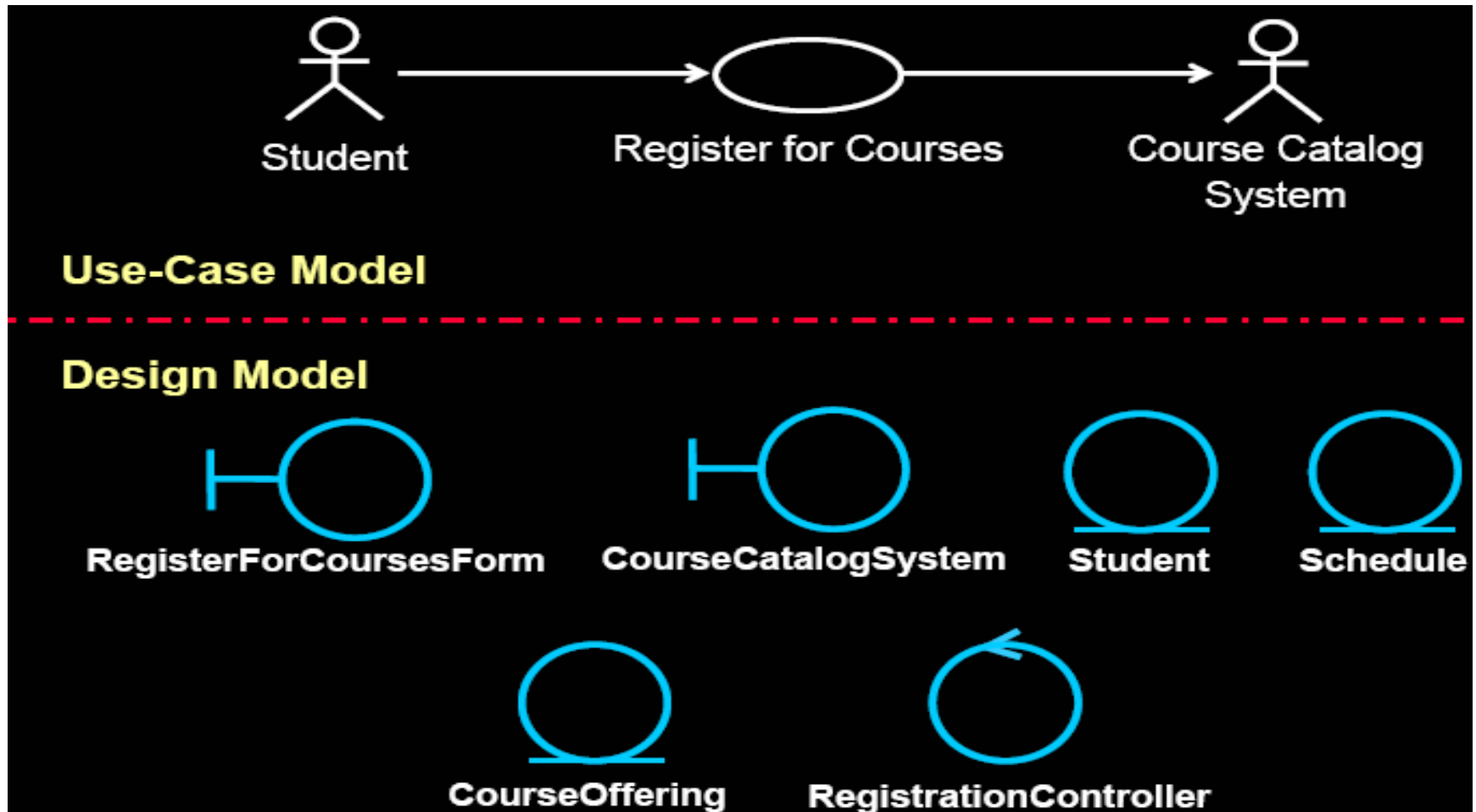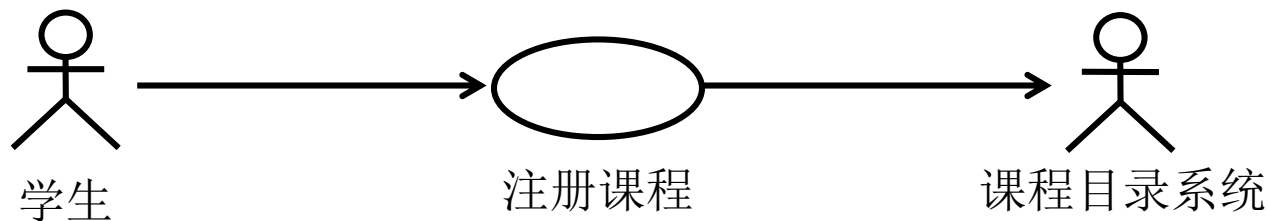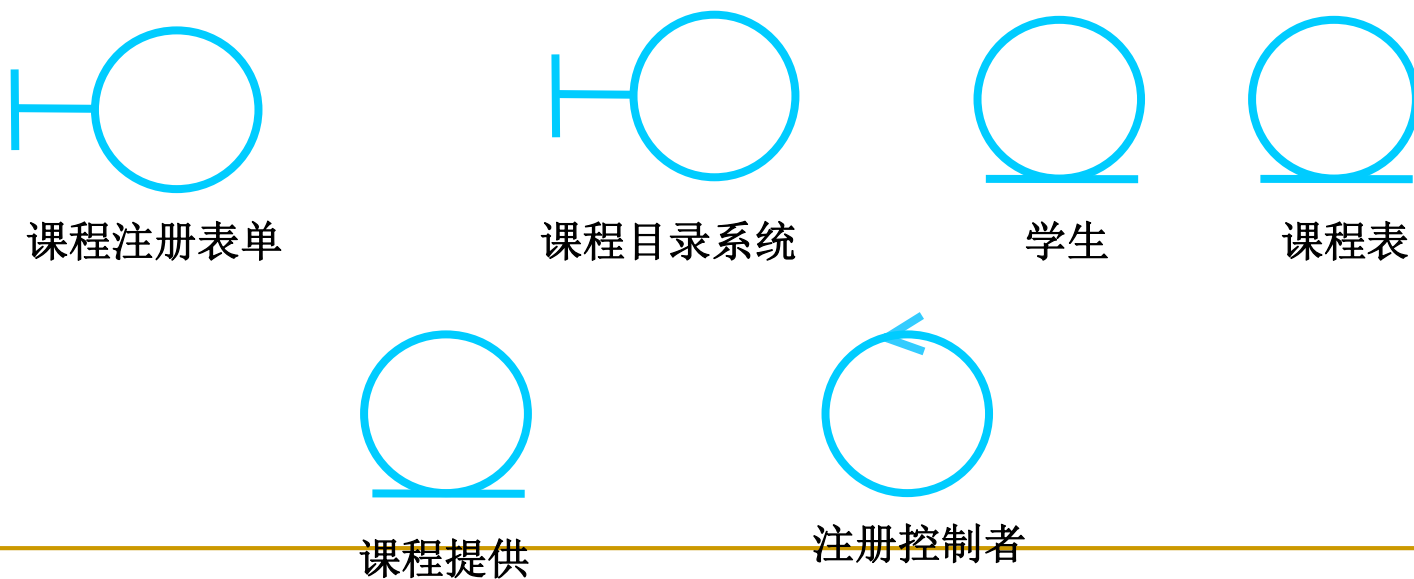
Use-Case Model

Design Model

Student — Register for Courses — Course Catalog System

RegisterForCoursesForm   CourseCatalogSystem   Student   Schedule

CourseOffering   RegistrationController

类总览

学生　　　　　注册课程　　　　课程目录系统

课程注册表单　　　课程目录系统　　　学生　　　课程表

课程提供　　　注册控制者

Use-Case Analysis Steps

- Supplement the Use-Case Description
- For each Use-Case Realization
  - ❑ Find Classes from Use-Case Behavior
  - ❑ Distribute Use-Case Behavior to Classes
- For each resulting class
  - ❑ Describe Responsibilities
  - ❑ Describe Attributes and Associations
  - ❑ Qualify Analysis Mechanisms
- Unify classes
- Checkpoints

第八章　用例分析

主要内容

用例分析总述

补充用例规约

查找类

将用例行为分配给类

描述类

描述分析机制

合并类

案例实践

Distribute Use-Case Behavior to Classes

- **For each use-case flow of events:**

  - **Identify classes**

  - **Allocate use-case responsibilities to classes**

  - **Model class interactions in Interaction diagrams**

Sequence Diagrams

Collaboration Diagrams

Use Case

Use-Case Realization
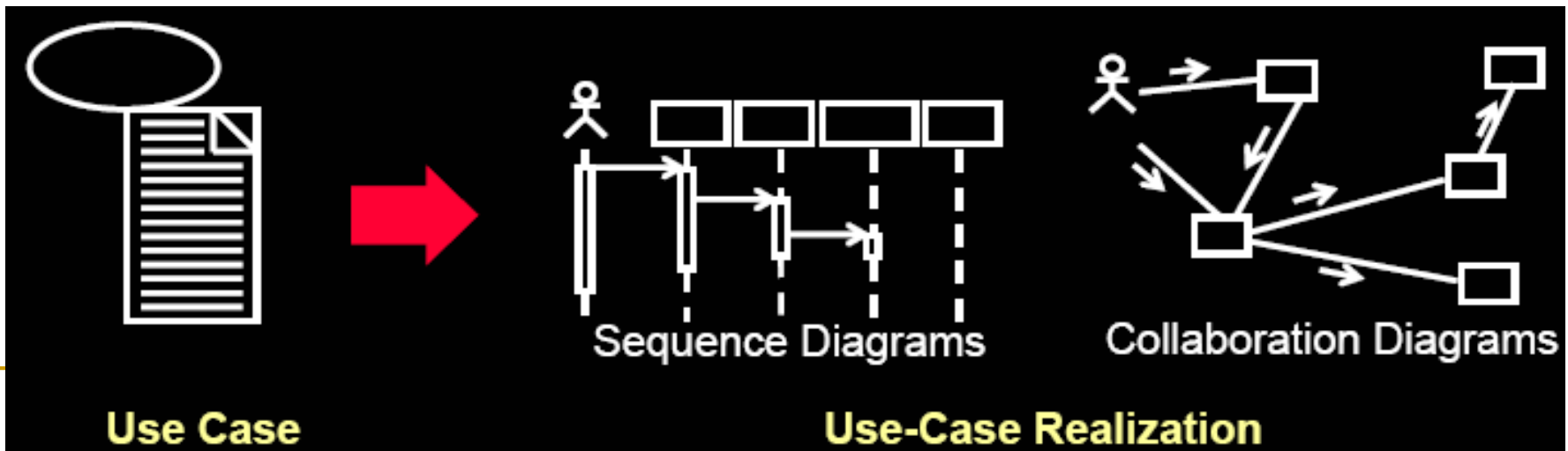
将用例行为分配给类

- **对于每个用例的事件流：**
  - 确定类
  - 将用例的职责分配给类
  - 在交互图中为类建模
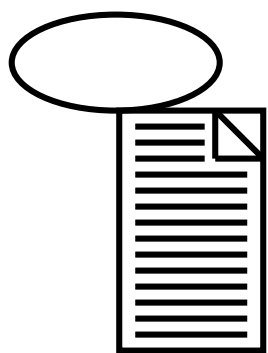
用例图
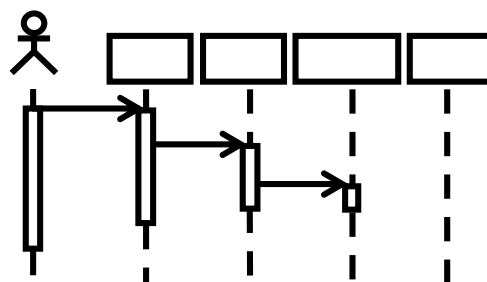
时序图
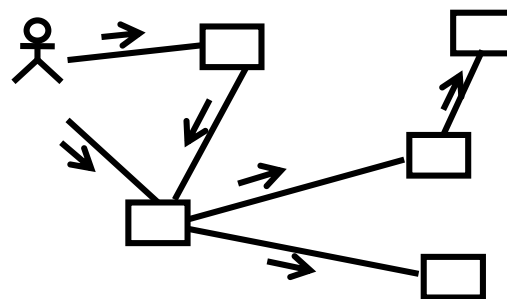
协作图

用例实现

Guidelines: Allocating Responsibilities to Classes

- **Use class stereotypes as a guide**

- **Boundary Classes**

  ❑ Behavior that involves communication with an actor

- **Entity Classes**

  ❑ Behavior that involves the data encapsulated within the abstraction

- **Control Classes**

  ❑ Behavior specific to a use case or part of a very important flow of events

将职责分配给类

- **用类的构造型做指导**

- **边界类**

  - **行为包括与参与者的联系**

- **实体类**

  - **行为包括封装数据**

- **控制类**

  - **用例或事件流特有的行为**

Guidelines: Allocating Responsibilities to Classes (cont.)

- **Who has the data needed to perform the responsibility?**

  - **If one class has the data, put the responsibility with the data**

  - **If multiple classes have the data:**

    - **Put the responsibility with one class and add a relationship to the other**

    - **Create a new class, put the responsibility in the new class, and add relationships to classes needed to perform the responsibility**

    - **Put the responsibility in the control class, and add relationships to classes needed to perform the responsibility**

The Anatomy of Sequence Diagrams
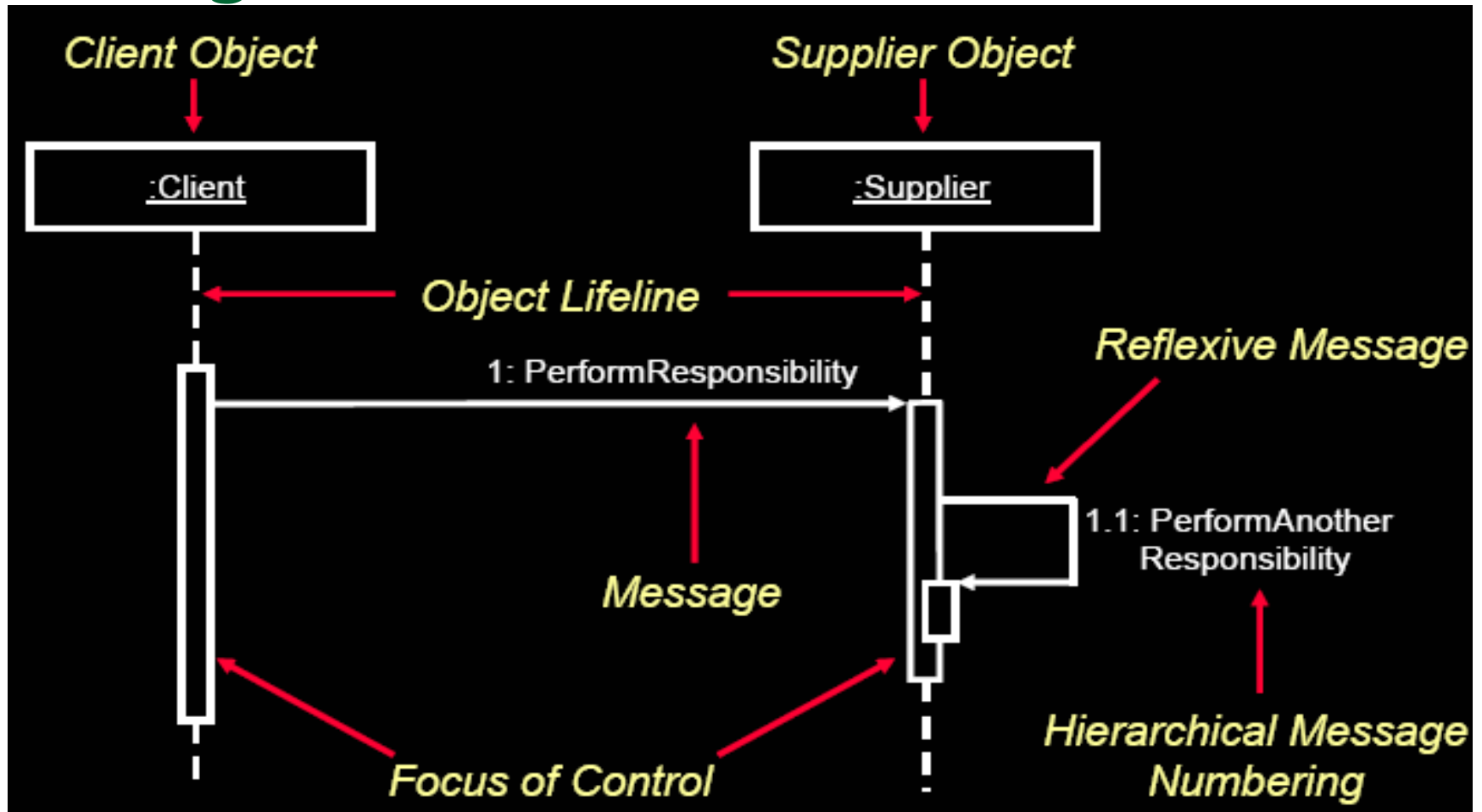
时序图示例
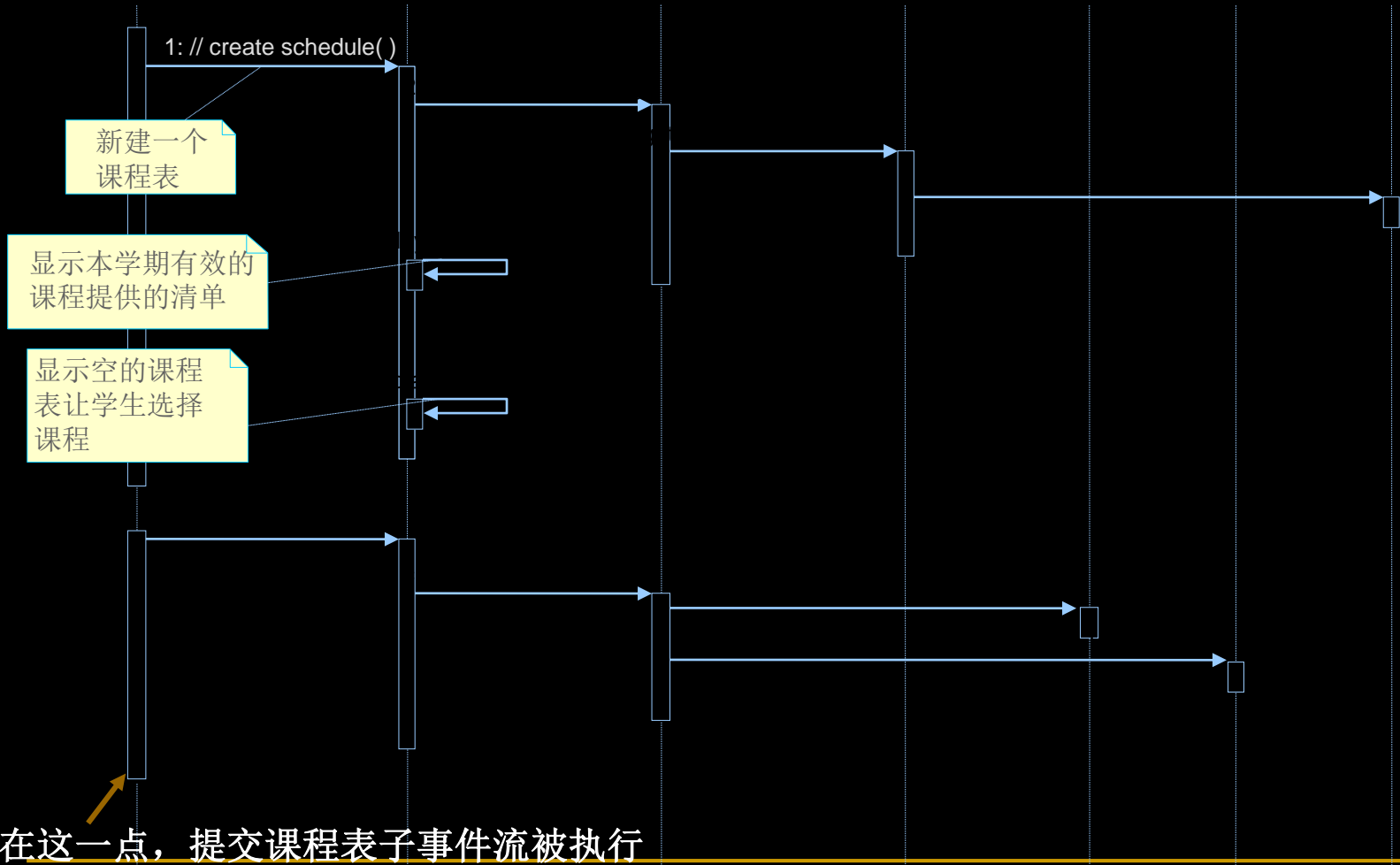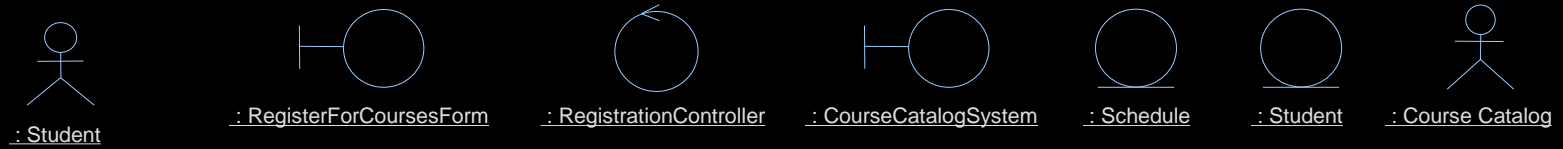
Example: Sequence Diagram
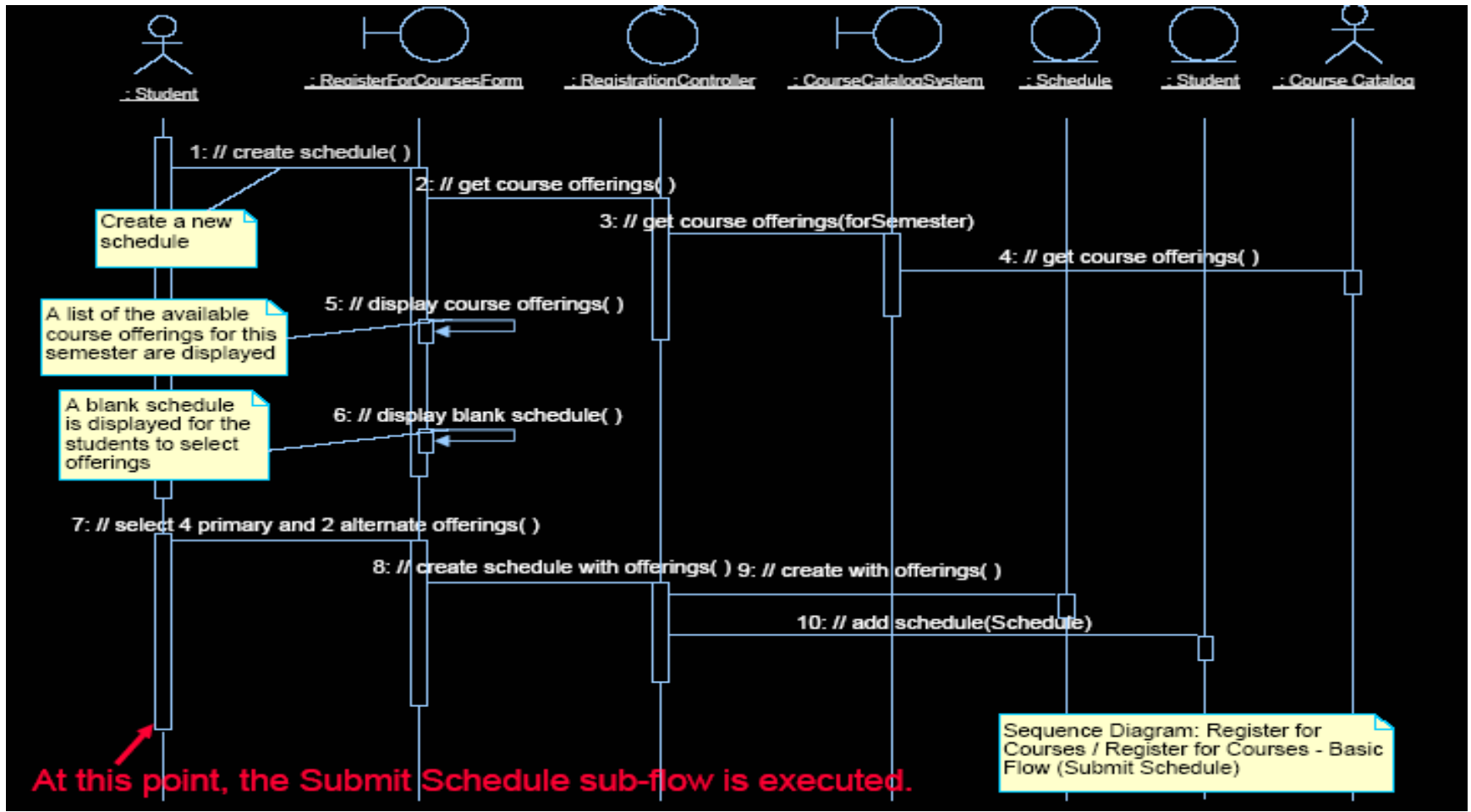
The Anatomy of Collaboration Diagrams
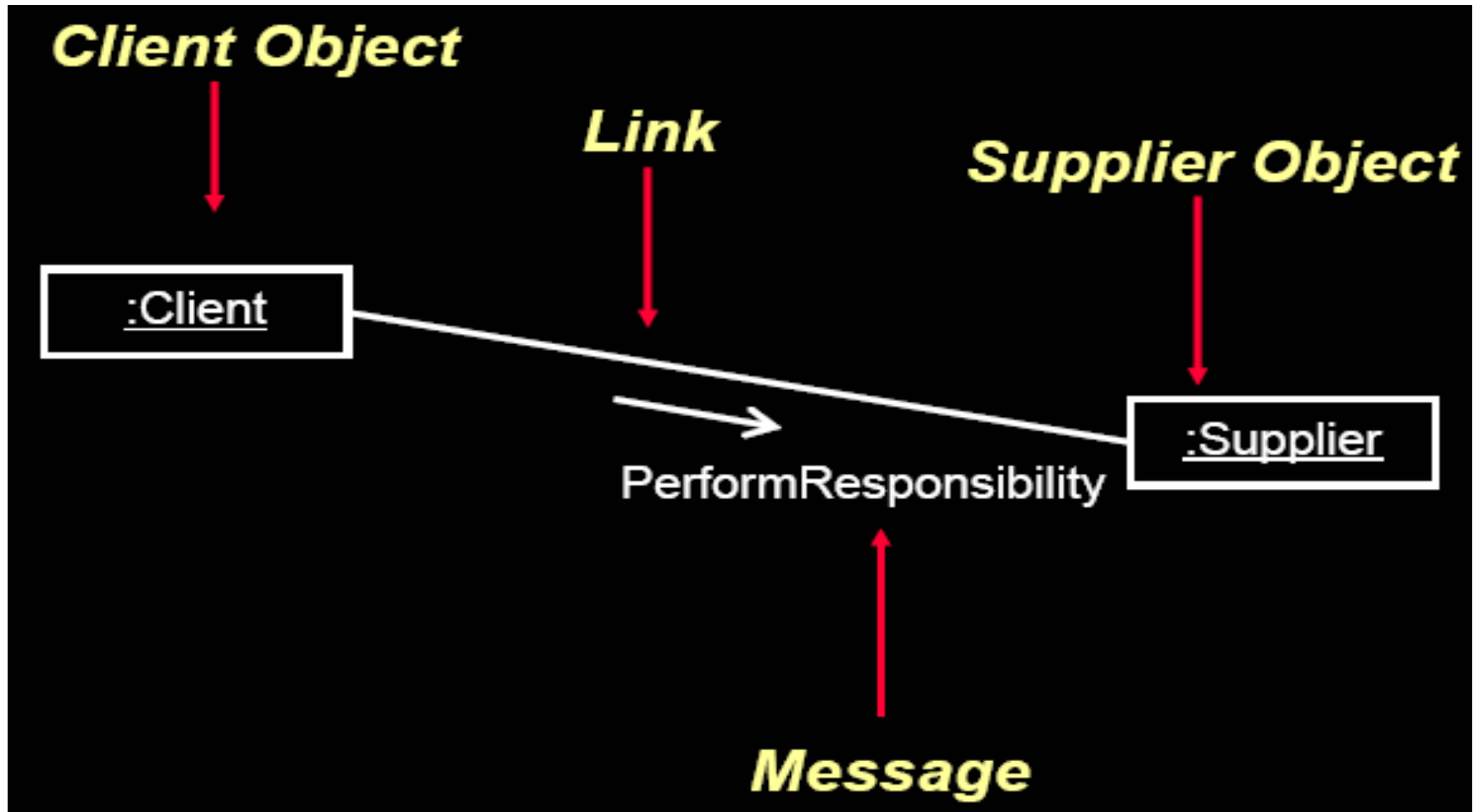
Example: Collaboration Diagram
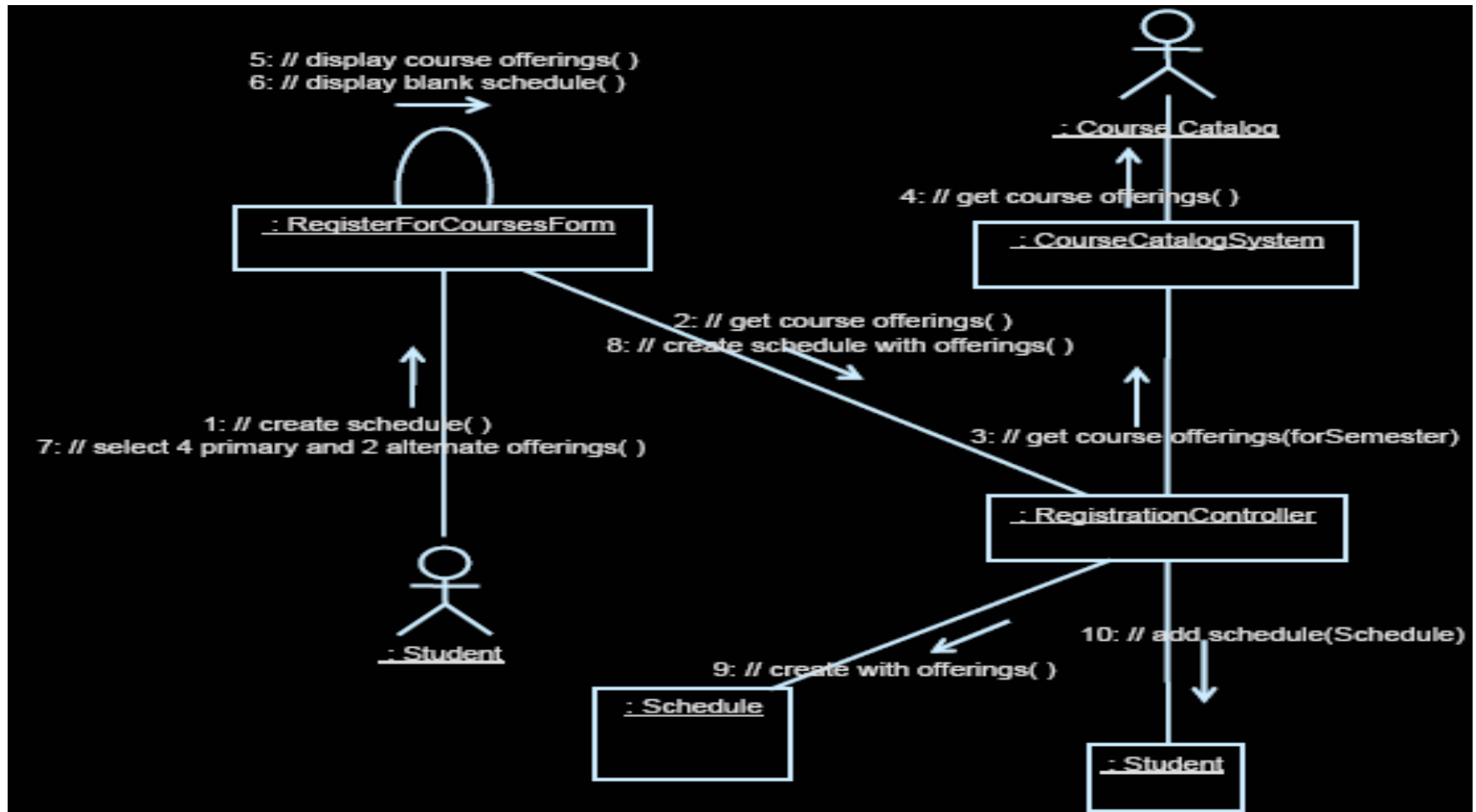
协作图示例
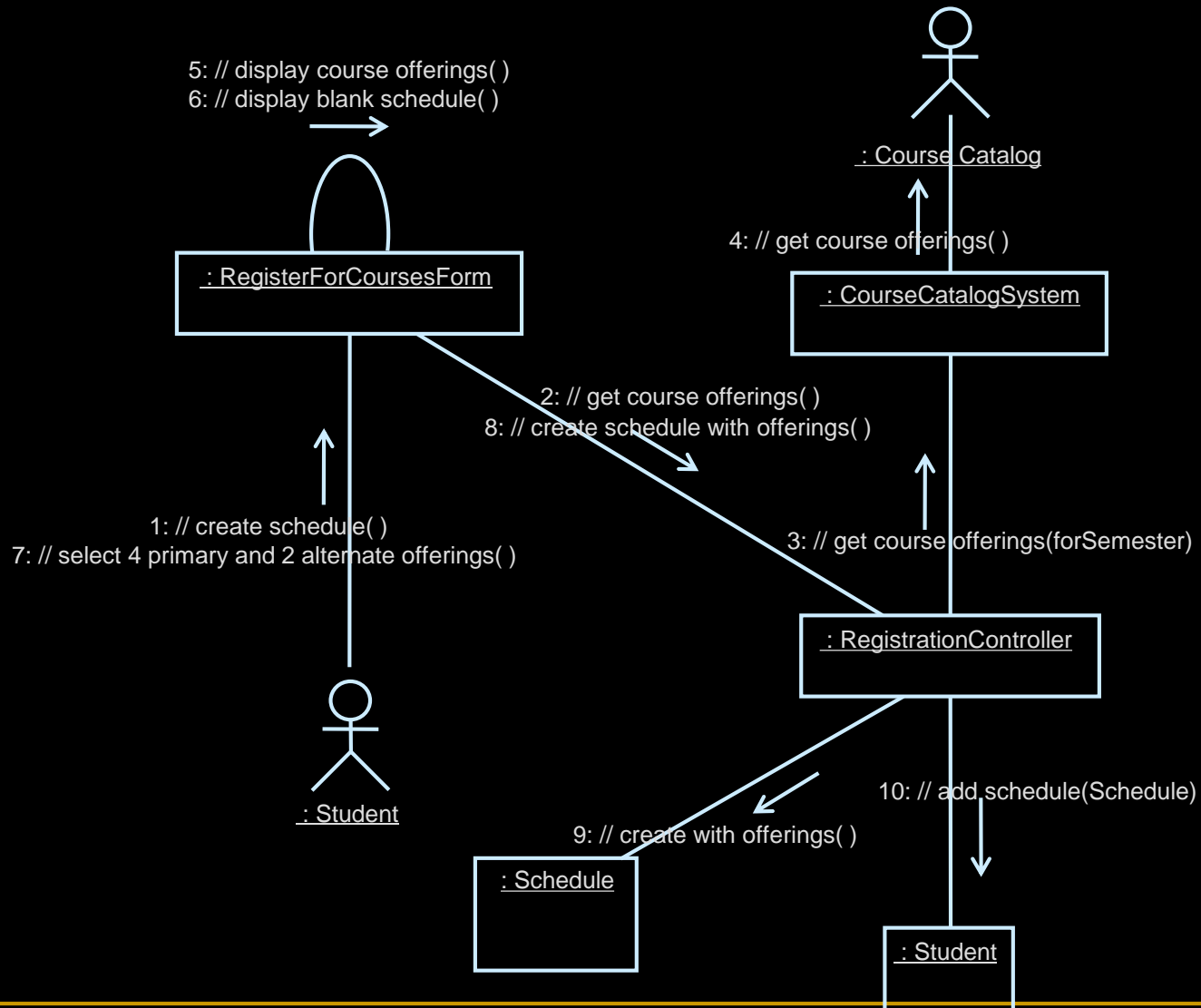
Use-Case Analysis Steps

- **Supplement the Use-Case Description**

- **For each Use-Case Realization**

    - ❑ Find Classes from Use-Case Behavior

    - ❑ Distribute Use-Case Behavior to Classes

- **For each resulting class**

    - ❑ Describe Responsibilities

    - ❑ Describe Attributes and Associations

    - ❑ Qualify Analysis Mechanisms

- **Unify classes**

- **Checkpoints**

第八章 用例分析

主要内容

用例分析总述

补充用例规约

查找类

将用例行为分配给类

描述类

描述分析机制

合并类

案例实践

One Interaction Diagram Is Not Good Enough
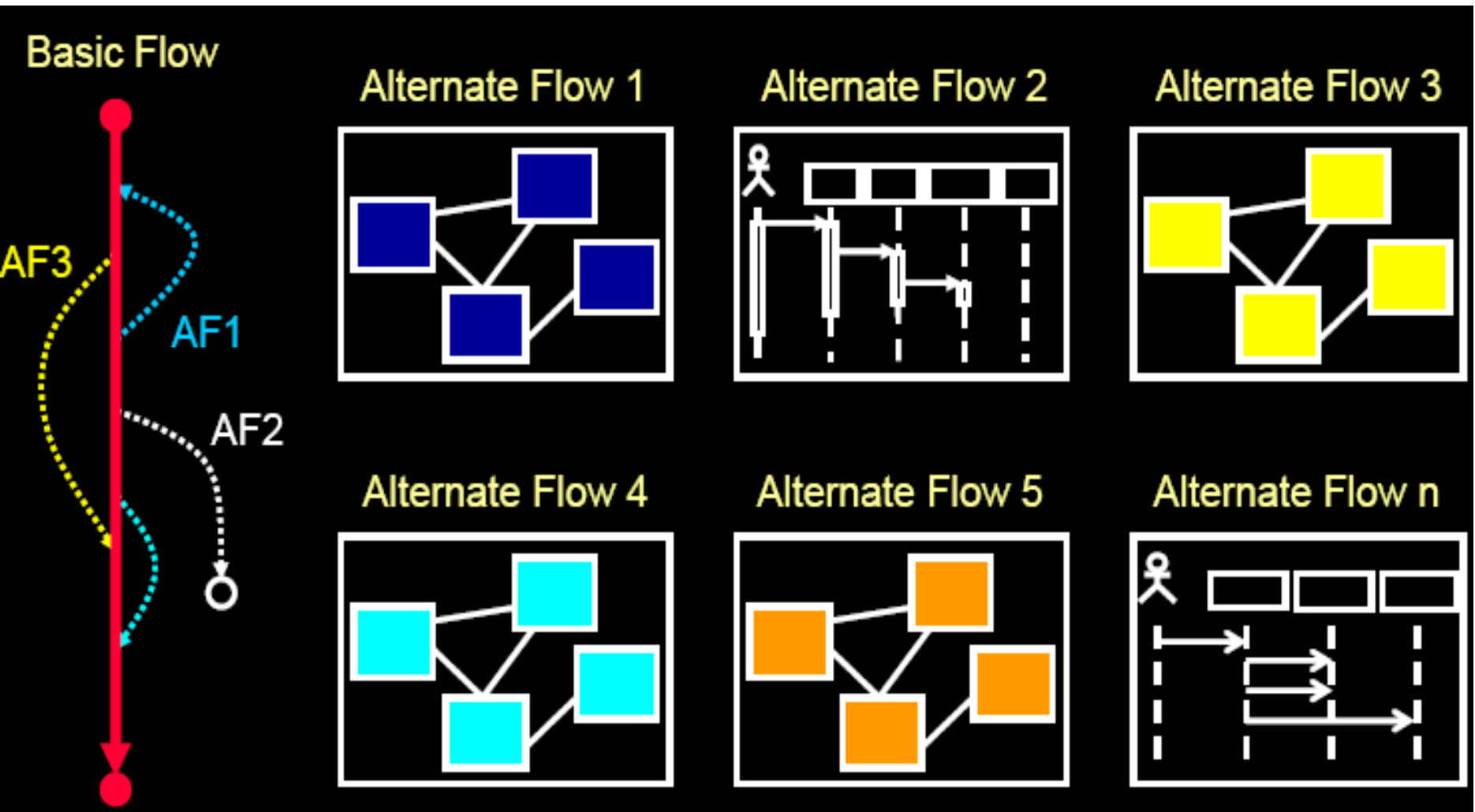
Describe Responsibilities

- **What are responsibilities?**

- **How do I find them?**

Example: View of Participating Classes (VOPC) Class Diagram

<<entity>>
Student

// get tuition()
// add schedule()
// get schedule()
// delete schedule()
// has pre-requisites()

<<control>>
RegistrationController

// get course offerings()
// get current schedule()
// delete current schedule()
// submit schedule()
// is registration open?()
// save schedule()
// create schedule with offerings()
// update schedule with new selections()

<<entity>>
Schedule

// commit()
// select alternate()
// remove offering()
// level()
// cancel()
// get cost()
// delete()
// submit()
// save()
// any conflicts?()
// create with offerings()
// update with new selections()

<<boundary>>
CourseCatalogSystem

// get course offerings()

<<boundary>>
RegisterForCoursesForm

// display course offerings()
// display blank schedule()
// update offering selections()

参与类图示例

```
┌─────────────────────────────┐
│       <<entity>>            │
│        Student              │
├─────────────────────────────┤
│                             │
├─────────────────────────────┤
│ // get tuition()            │
│ // add schedule()           │
│ // get schedule()           │
│ // delete schedule()        │
│ // has pre-requisites()     │
└─────────────────────────────┘
```

```
┌─────────────────────────────────────────┐
│              <<control>>                 │
│          RegistrationController          │
├─────────────────────────────────────────┤
│                                          │
├─────────────────────────────────────────┤
│ // get course offerings()                │
│ // get current schedule()                │
│ // delete current schedule()             │
│ // submit schedule()                     │
│ // is registration open?()               │
│ // save schedule()                       │
│ // create schedule with offerings()      │
│ // update schedule with new selections() │
└─────────────────────────────────────────┘
```

```
┌─────────────────────────────┐
│       <<entity>>            │
│        Schedule             │
├─────────────────────────────┤
│                             │
├─────────────────────────────┤
│ // commit()                 │
│ // select alternate()       │
│ // remove offering()        │
│ // level()                  │
│ // cancel()                 │
│ // get cost()               │
│ // delete()                 │
│ // submit()                 │
│ // save()                   │
│ // any conflicts()          │
│ // create with offerings()  │
│ // update with new selections() │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│       <<boundary>>          │
│     CourseCatalogSystem     │
├─────────────────────────────┤
│                             │
├─────────────────────────────┤
│ // get course offerings()   │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│       <<boundary>>          │
│    RegisterForCoursesForm   │
├─────────────────────────────┤
│                             │
├─────────────────────────────┤
│ // display course offerings() │
│ // display blank schedule() │
│ // update offering selections() │
└─────────────────────────────┘
```

说明职责

- **什么是职责**

- **怎样找到职责**

  交互图

:Client		:Supplier

  // PerformResponsibility

  类图

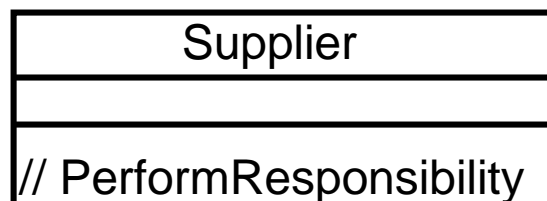 | Supplier |
  |----------|
  | |
  | // PerformResponsibility |

Maintaining Consistency: What to Look For

- **In order of criticality**

  - ❑ Redundant responsibilities across classes

  - ❑ Disjoint responsibilities within classes

  - ❑ Class with one responsibility

  - ❑ Class with no responsibilities

  - ❑ Better distribution of behavior

  - ❑ Class that interacts with many other classes

维持一致性

- **类中多余的职责**

- **类中分离的职责**

- **只有一个职责的类**

- **没有职责的类**

- **更好的行为分配方式**

- **与许多其他类有交互作用的类**

Use-Case Analysis Steps

- Supplement the Use-Case Description

- For each Use-Case Realization

  - Find Classes from Use-Case Behavior

  - Distribute Use-Case Behavior to Classes

- For each resulting class

  - Describe Responsibilities

  - Describe Attributes and Associations

  - Qualify Analysis Mechanisms

- Unify classes

- Checkpoints

Finding Attributes

- **Properties/characteristics of identified classes**

- **Information retained by identified classes**

- **"Nouns" that did not become classes**

  - Information whose value is the important thing

  - Information that is uniquely "owned" by an object
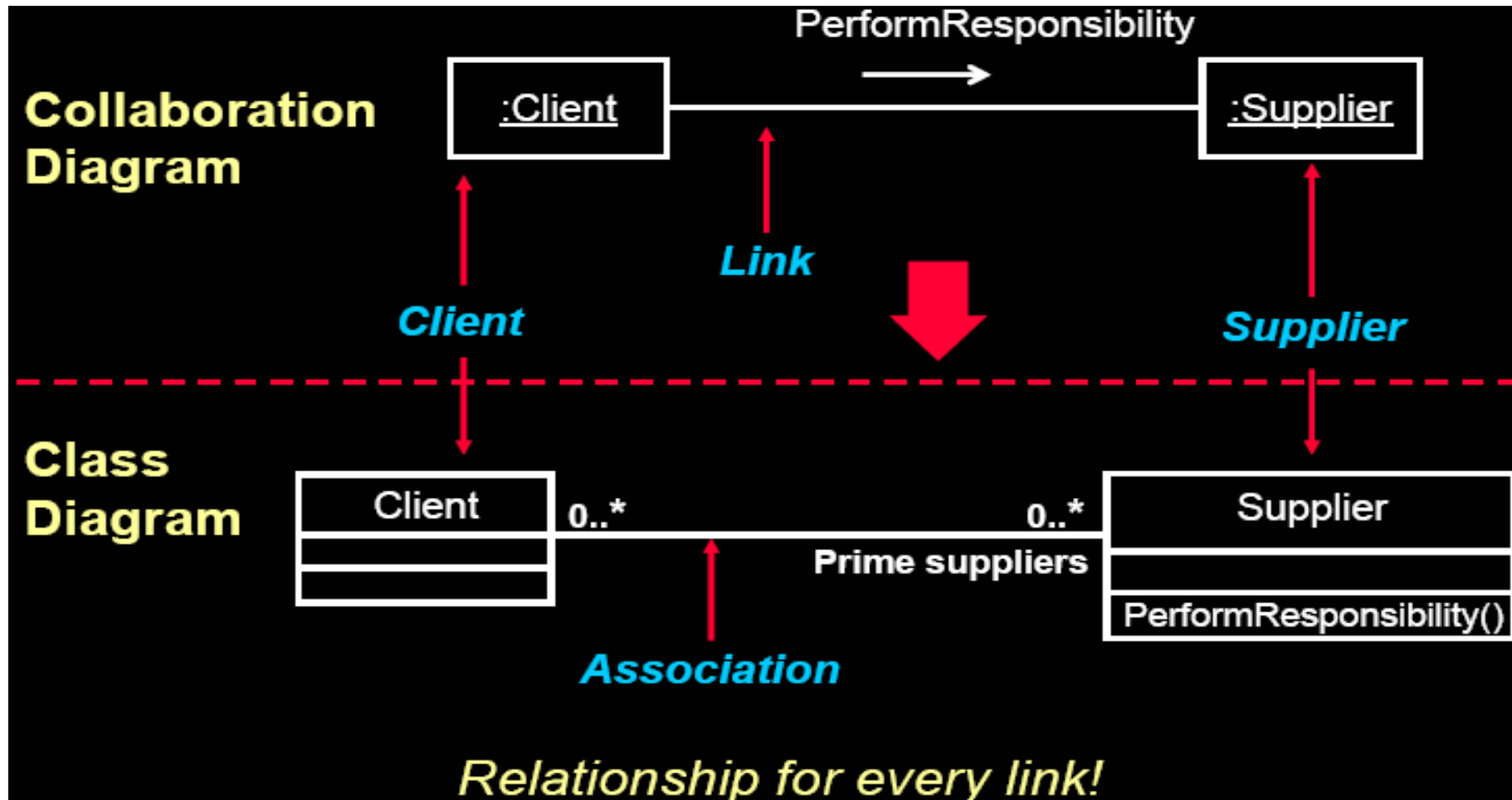
  - Information that has no behavior

确定属性

- **类的特征**

- **类要保留的信息**

- **不能成为类的名词**
  - 值很重要的信息
  - 某个对象独有的信息
  - 没有行为的信息

Review: What Is an Association?

- **The semantic relationship between two or more classifiers that specifies connections among their instances**

    - **A structural relationship, specifying that objects of one thing are connected to objects of another**
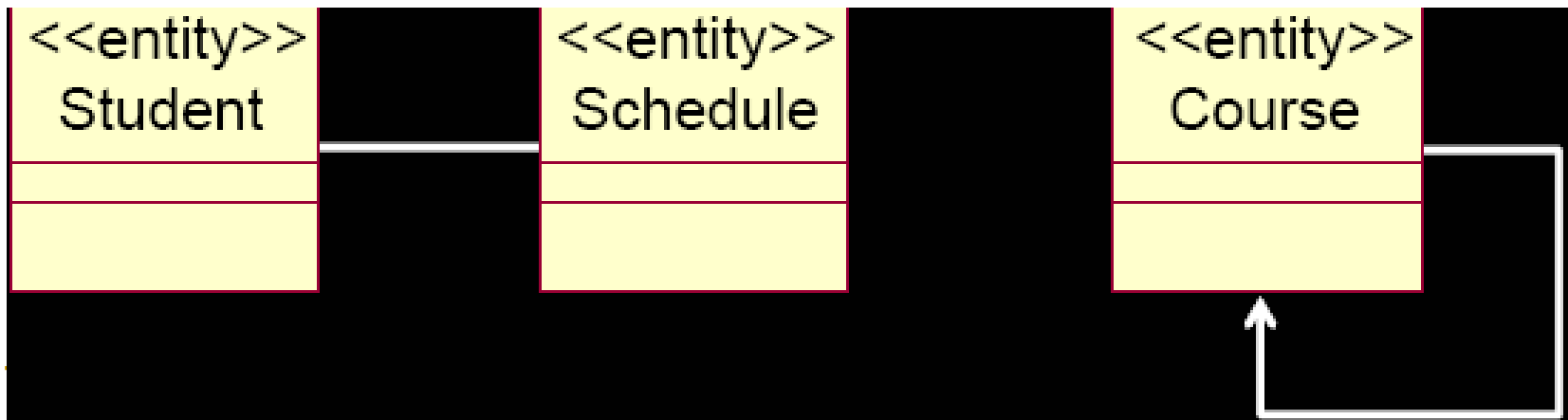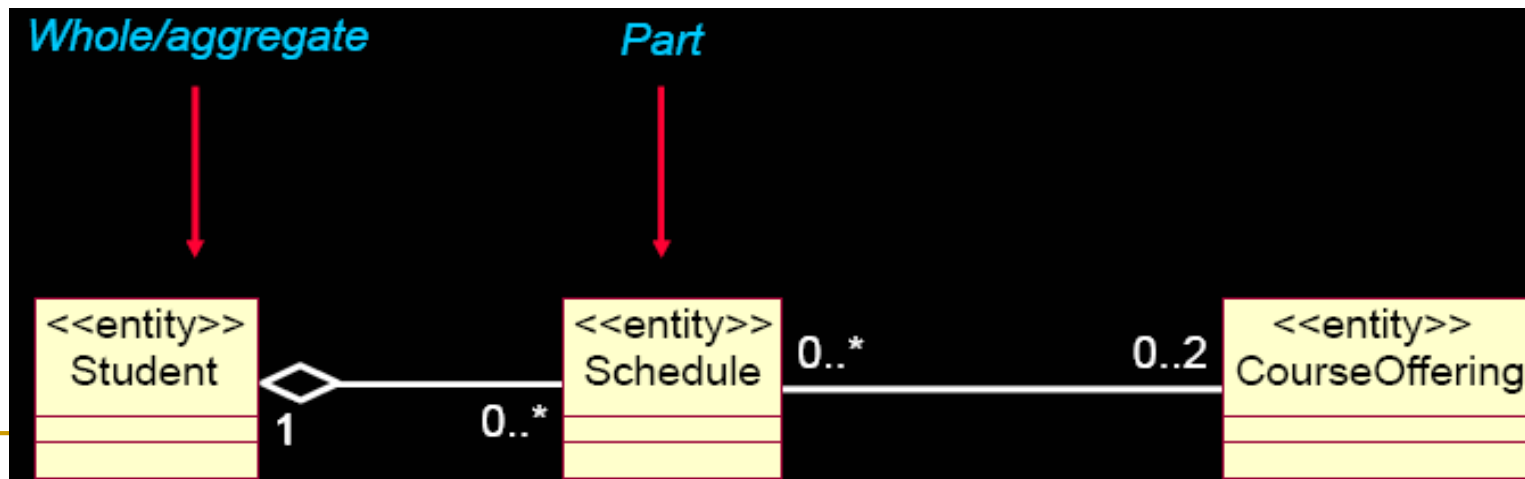
Finding Relationships

Review: What Is Aggregation?

- **A special form of association that models whole-part relationship between an aggregate (the whole) and its parts**

Association or Aggregation?

- **If two objects are tightly bound by a whole-part relationship**
  - The relationship is an aggregation.

Car		Door
1	0..2,4	

- **If two objects are usually considered as independent, although they are often linked**
  - The relationship is an association.

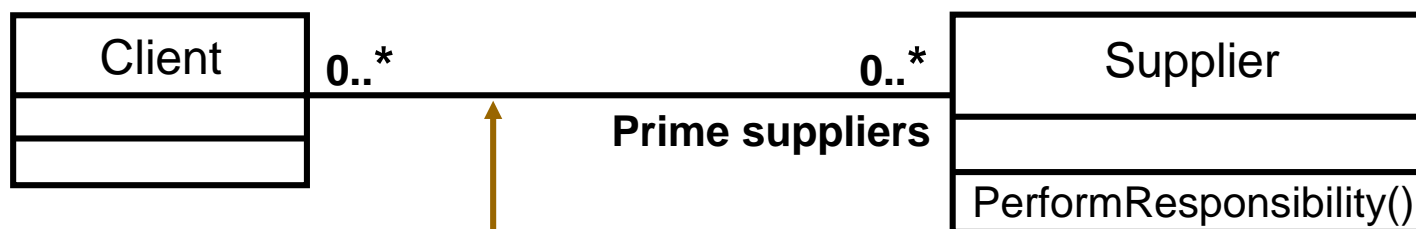| Car		Door
1	0..2,4	

When in doubt use association

确定关联

PerformResponsibility

协作图

:Client ——————————→ :Supplier

Client

链

Supplier

类图

Client	

0..* ————————————————— 0..*
Prime suppliers

Supplier
PerformResponsibility()

关联

每条链都转换为一个关联

关联还是聚集

- **两个对象有整体–部分的关系**

  - 是聚集关系

Car		Door
1	0..2,4	
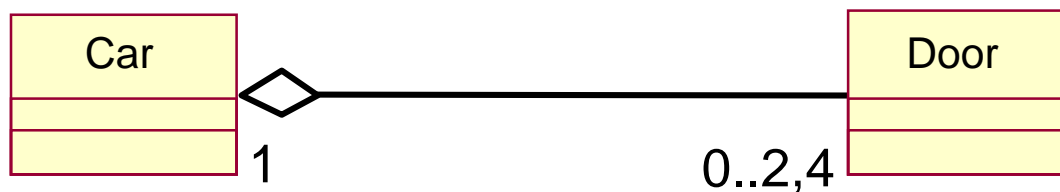
- **两个对象被认为是独立的**

  - 是关联关系

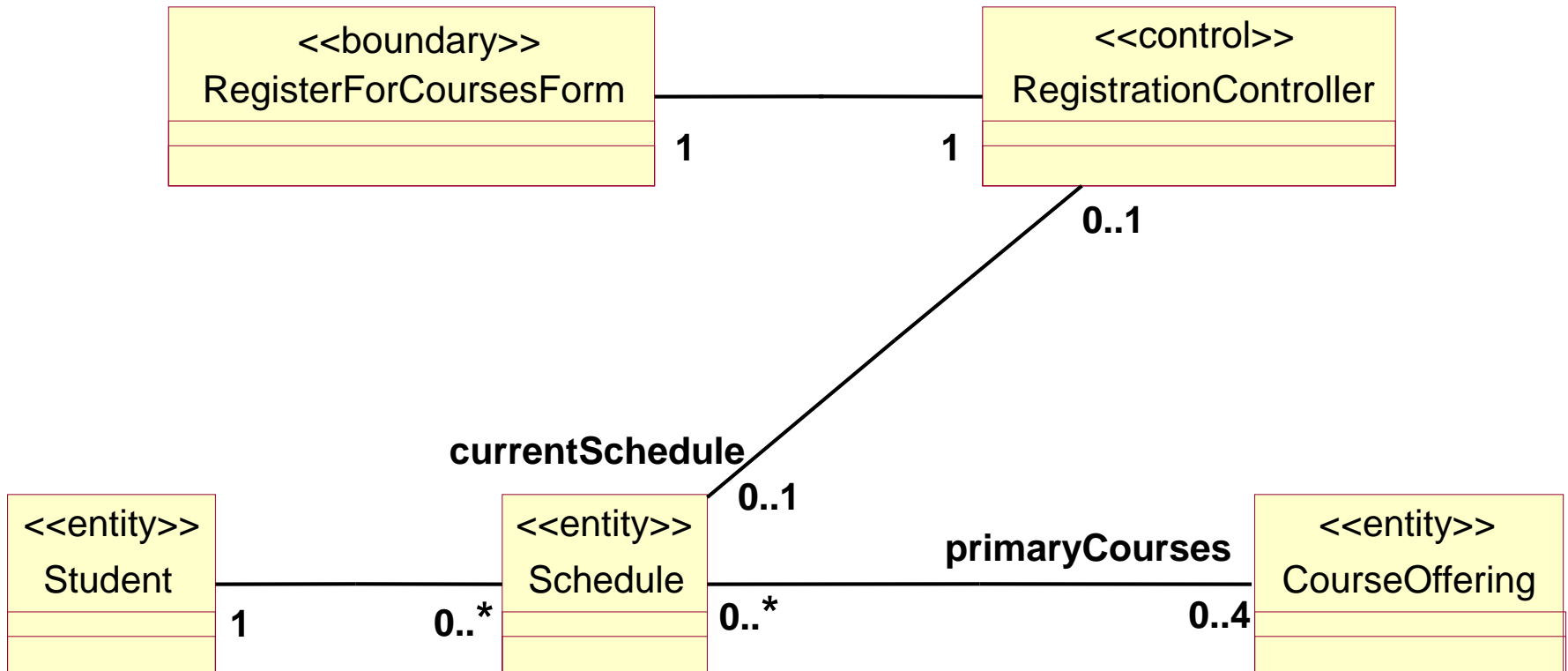 | Car | | Door |
    |---|---|---|
    | | | |
    | 1 | 0..2,4 | |
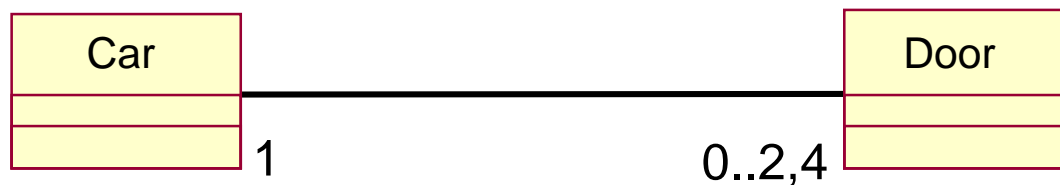
不确定时使用关联关系

参与类图：关联关系示例

Use-Case Analysis Steps

- Supplement the Use-Case Description
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify classes
- Checkpoints

第八章 用例分析
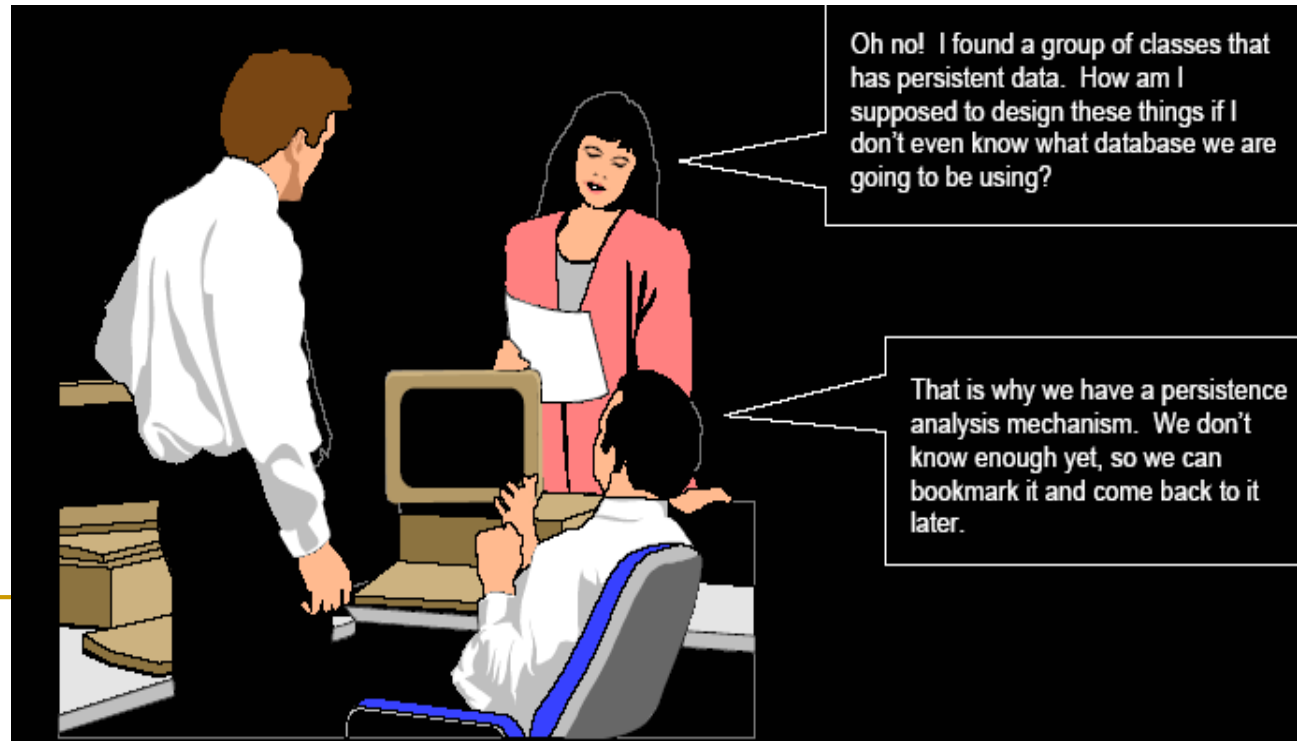
主要内容

用例分析总述

补充用例规约

查找类

将用例行为分配给类

描述类

描述分析机制

合并类

案例实践

Review: Why Use Analysis Mechanisms?

■ *Analysis mechanisms are used during analysis to reduce the complexity of analysis, and to improve its consistency by providing designers with a shorthand representation for complex behavior.*

Describing Analysis Mechanisms

- Collect all analysis mechanisms in

  a list

- Draw a map of the client classes

  to the analysis mechanisms

- Identify characteristics of the

  analysis mechanisms

描述分析机制

- **收集所有分析机制形成列表**

- **绘制客户类到分析机制的映射图**

- **确定分析机制的特征**

Example: Describing Analysis Mechanisms

- class to analysis mechanism map

Analysis Class	Analysis Mechanism(s)
Student	Persistency, Security
Schedule	Persistency, Security
CourseOffering	Persistency, Legacy Interface
Course	Persistency, Legacy Interface
RegistrationController	Distribution

描述分析机制示例

类到分析机制的映射图

类	分析机制
学生	持久性, 安全性
课程表	持久性, 安全性
课程提供	持久性, 遗留界面
课程	持久性, 遗留界面
注册控制器	分布性

Example: Describing Analysis Mechanisms (cont.)

- **Analysis mechanism characteristics**

- **Persistency for Schedule class:**

  - Granularity: 1 to 10 Kbytes per product

  - Volume: up to 2,000 schedules

  - Access frequency

    - • Create: 500 per day

    - • Read: 2,000 access per hour

    - • Update: 1,000 per day

    - Delete: 50 per day

  - Other characteristics

描述分析机制示例

课程表类使用的持久性：

- ❑ 粒度：每张课程表占用1到10千字节
- ❑ 容量：上限为2,000张课程表
- ❑ 访问频率
  - ■ 创建：每天500次
  - ■ 读取：每小时2000次
  - ■ 更新：每天1000次
  - ■ 删除：每天50次
- ❑ 其它特征

Use-Case Analysis Steps

- Supplement the Use-Case Description
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify classes
- Checkpoints

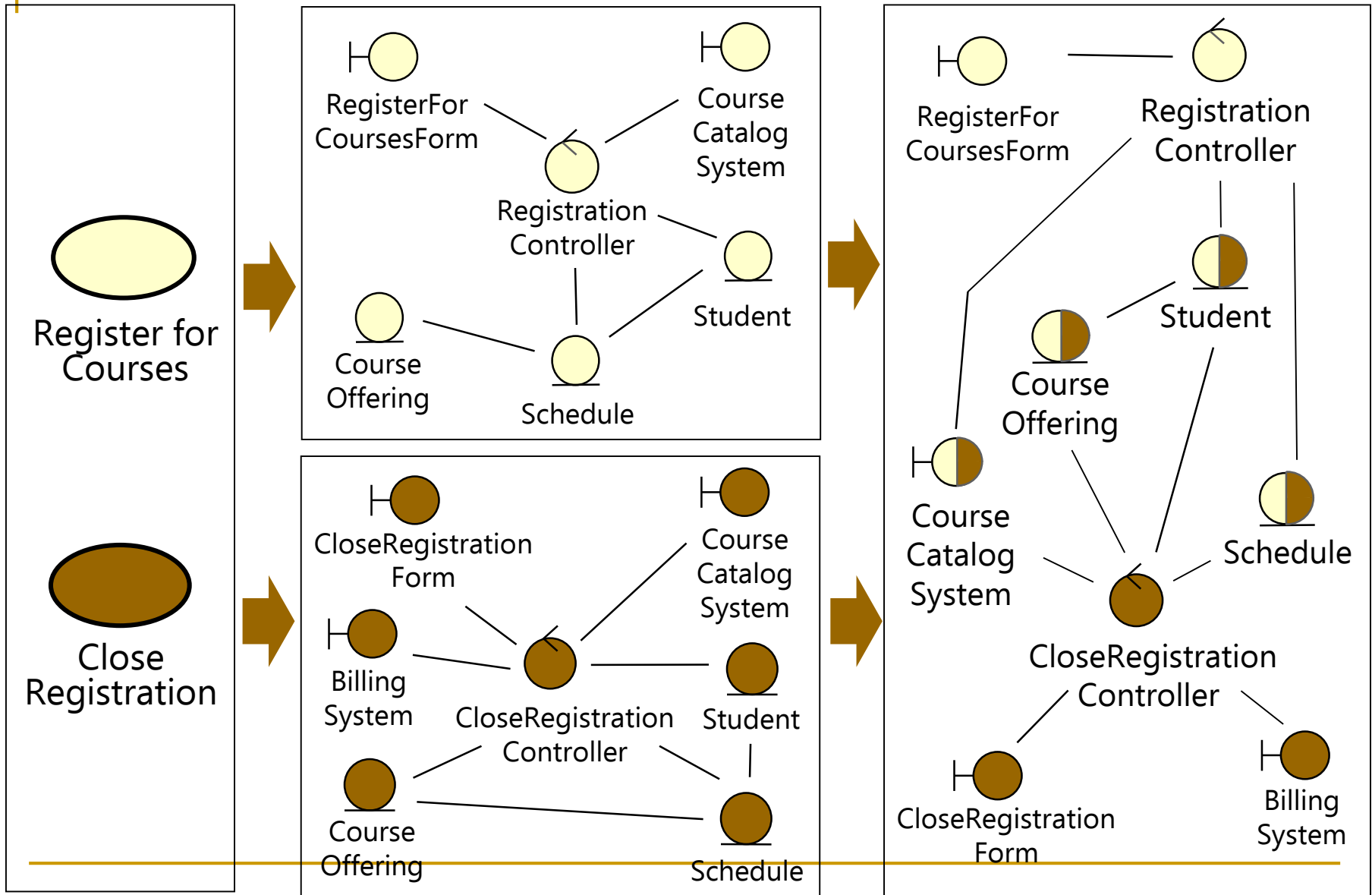# 第八章 用例分析

主要内容

用例分析总述

补充用例规约

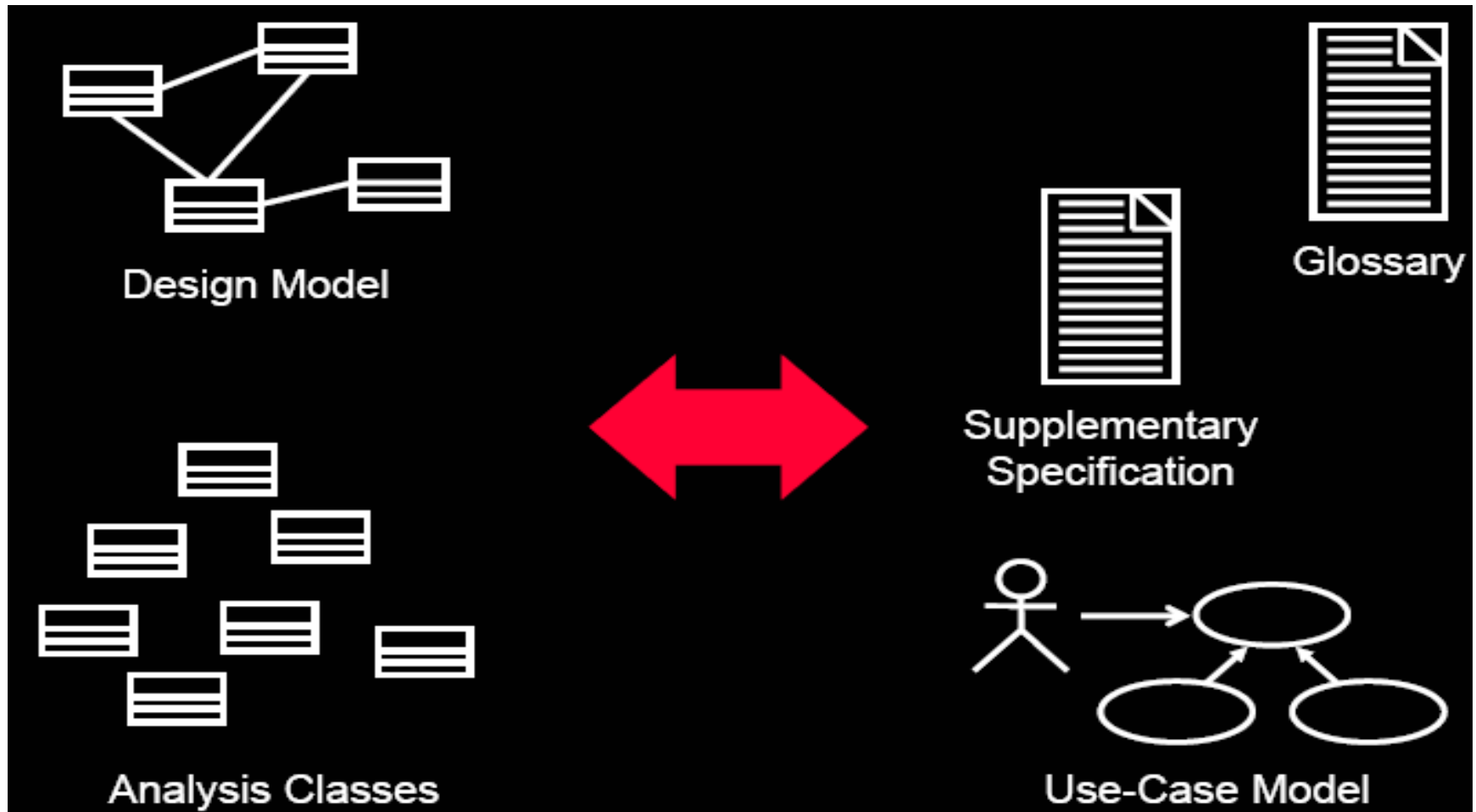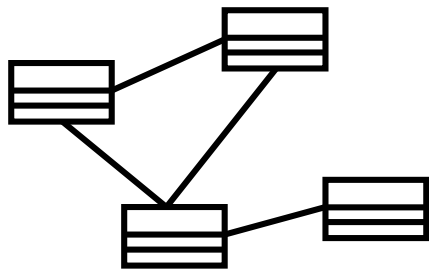查找类

将用例行为分配给类

描述类

描述分析机制

合并类

案例实践

Unify classes
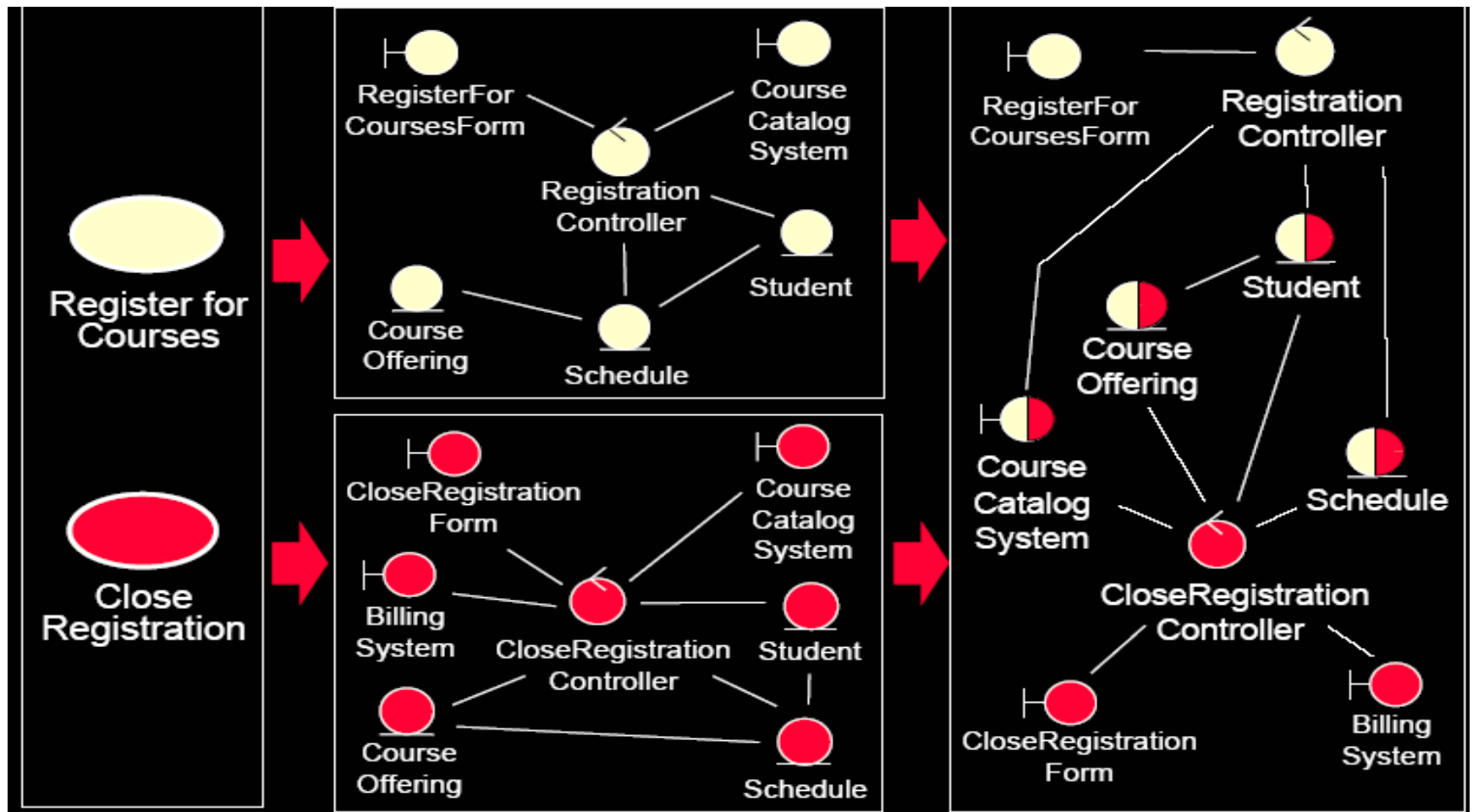
合并类

Evaluate Your Results

Design Model

Analysis Classes

Supplementary Specification
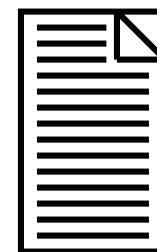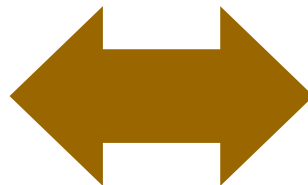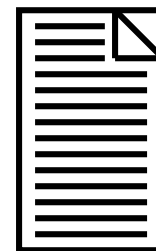
Glossary

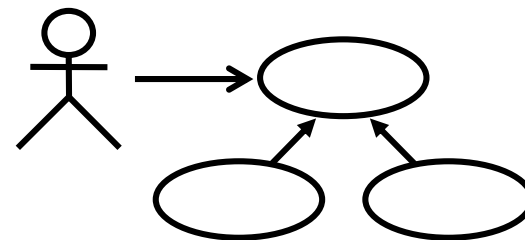Use-Case Model

评估用例分析结果

设计模型

术语表

补充规约

类
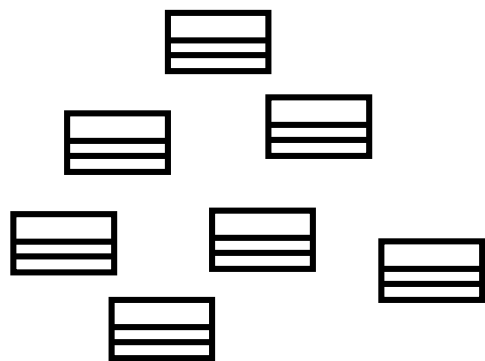
用例模型

Use-Case Analysis Steps

- Supplement the Use-Case Description
- For each Use-Case Realization
    - Find Classes from Use-Case Behavior
    - Distribute Use-Case Behavior to Classes
- For each resulting class
    - Describe Responsibilities
    - Describe Attributes and Associations
    - Qualify Analysis Mechanisms
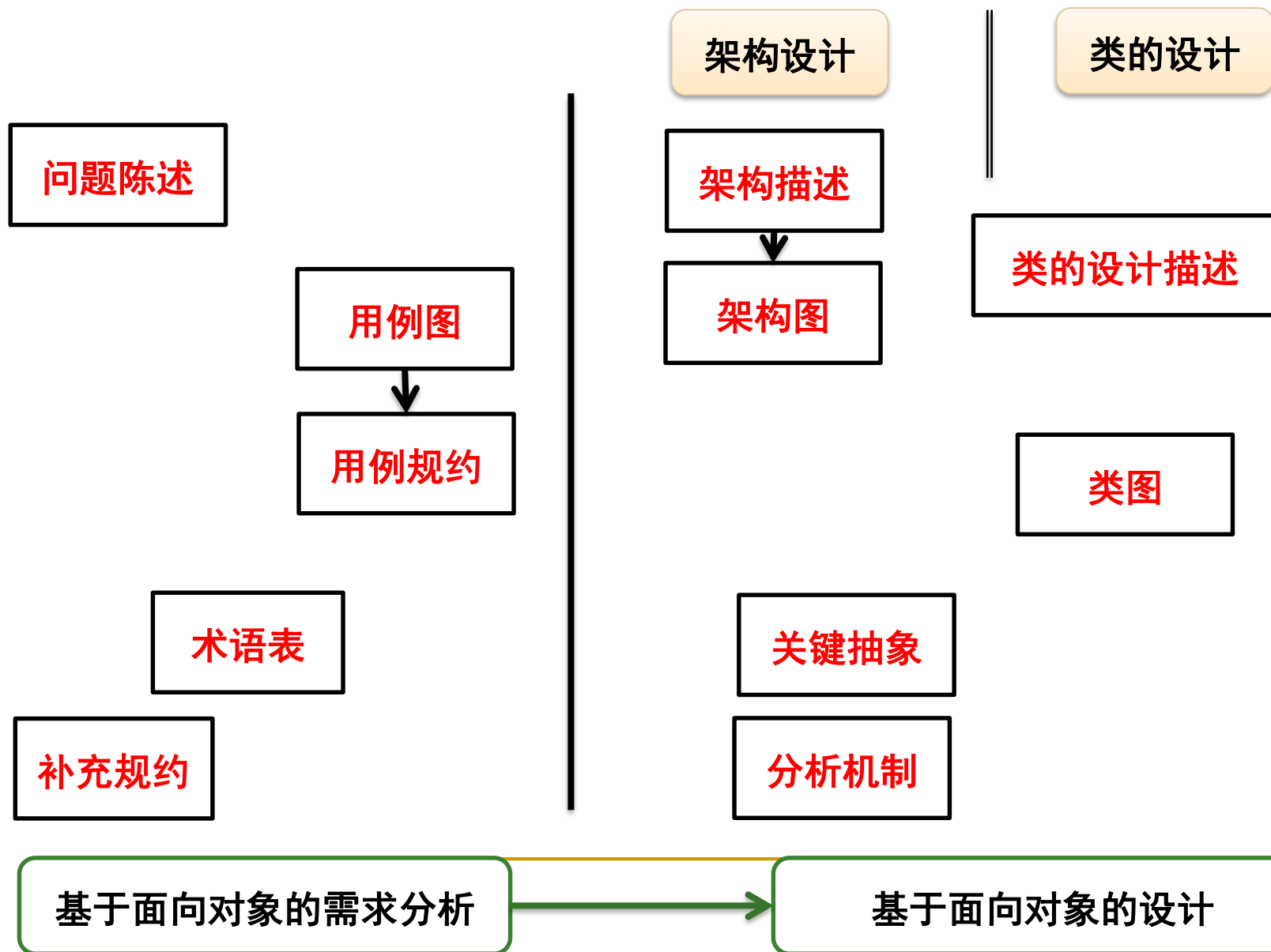- Unify classes
- Checkpoints

Checkpoints: classes

- **Are the classes reasonable?**
- **Does the name of each class clearly reflect the role it plays?**
- **Does the class represent a single well-defined abstraction?**
- **Are all attributes and responsibilities functionally coupled?**
- **Does the class offer the required behavior?**
- **Are all specific requirements on the class addressed?**

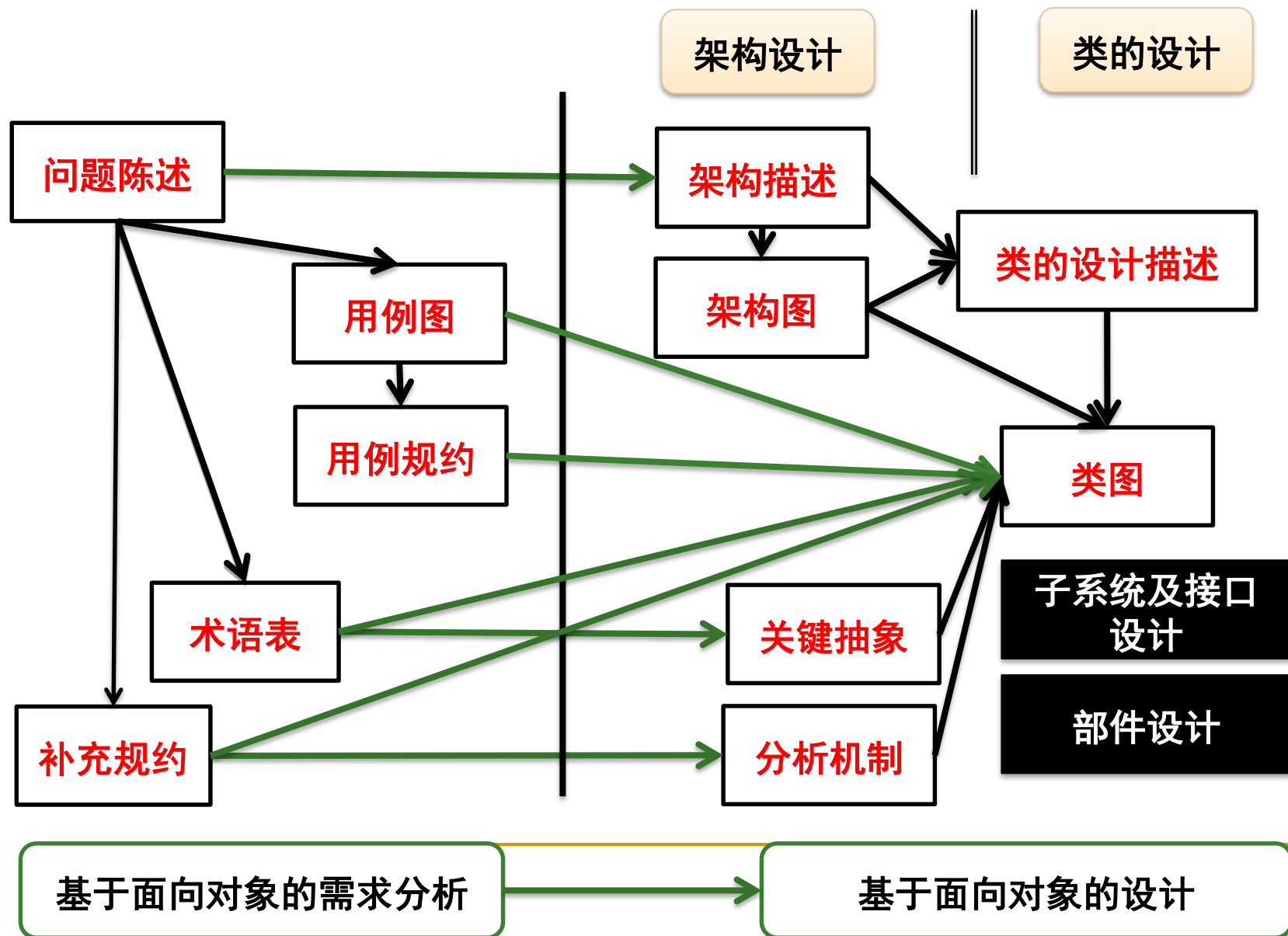# Checkpoints: Use-Case 功能的实现

- **Have all the main and/or sub-flows been handled, including exceptional cases?**

- **Have all the required objects been found?**

- **Has all behavior been unambiguously distributed to the participating objects?**

- **Has behavior been distributed to the right objects?**

- **Where there are several Interaction diagrams, are their relationships clear and consistent?**

基于面向对象的分析.vs.设计

架构设计

类的设计

问题陈述

架构描述

↓

架构图

类的设计描述

用例图

↓

用例规约

类图

术语表

关键抽象

补充规约

分析机制

基于面向对象的需求分析 → 基于面向对象的设计

基于面向对象的分析.vs.设计

架构设计

类的设计

问题陈述 → 架构描述

架构描述 → 架构图

用例图

用例规约

架构图 → 类的设计描述

类的设计描述 → 类图

术语表

补充规约

关键抽象

分析机制

类图

子系统及接口设计

部件设计

基于面向对象的需求分析 → 基于面向对象的设计

Exercise: Use-Case Analysis (cont.)

- **Given the following:**

  - Use-Case Model, especially the use-case flows of events

  - Key abstractions/classes

  - The Supplementary Specification

  - The possible analysis mechanisms

实验: Use-Case Analysis

- **Identify the following for a particular use case:**
    - The classes, along with their:
        - • Brief descriptions
        - • Stereotypes
        - • Responsibilities
    - The collaborations needed to implement the use case
    - class attributes and relationships
    - class analysis mechanisms

实验: Use-Case Analysis (cont.)

- **Produce the following for a particular use case:**

  - VOPC class diagram, containing the classes, their stereotypes, responsibilities, attributes, and relationships

  - class to analysis mechanism map

实验：Use-Case Analysis（cont.）

- **提交的制品**
  - **补充用例规约**
  - **用例分析**
    - **针对用例的类的析取**
    - **3-5个用例**
    - **每个用例的类图须有类的名称、属性、操作**
  - **分析机制**
  - **系统总的类图（包括类及其关系）**

理论作业 8th

1. 用例分析（类的析取）的目标是什么？

2. 用例分析的步骤和制品是什么？

3. 分析机制存在的意义是什么？

4. 类的操作是怎样确定的？

5. 类的属性是怎样确定的？

6. 类的命名有哪些需要注意的？