TSINGHUA UNIVERSITY

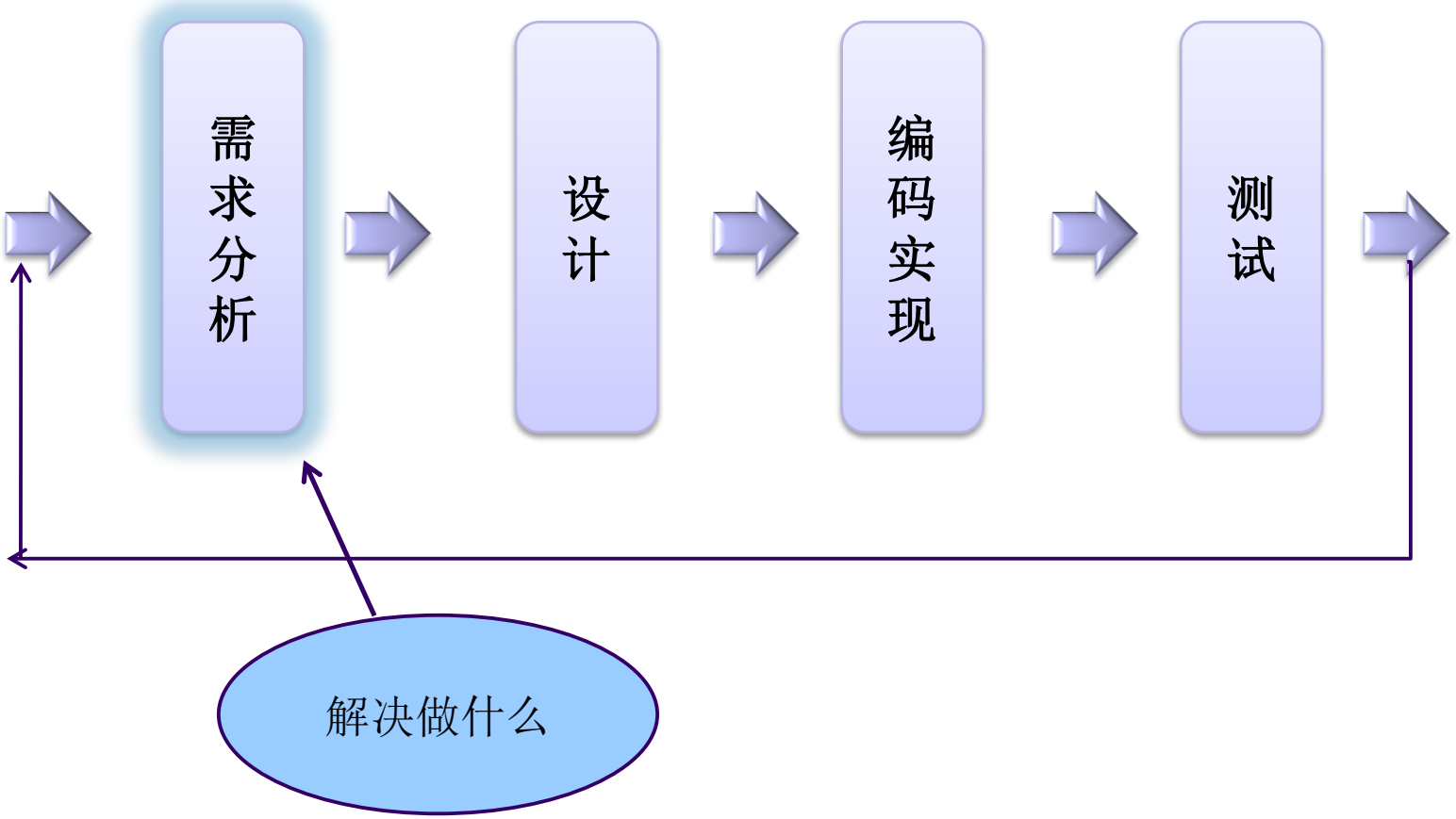
需求分析

董渊，吕勇强

清华大学

清華大學 Tsinghua University

软件工程生命周期回顾



需求预期合理

需求理解偏差

最重要失败因素

项目失败因素列表：	%
1. **Incomplete Requirements**	**13.1%**
2. **Lack of User Involvement**	**12.4%**
3. Lack of Resources	10.6%
4. Unrealistic Expectations	9.9%
5. Lack of Executive Support	9.3%
6. **Changing Requirements & Specifications**	**8.7%**
7. Lack of Planning	8.1%
8. Didn‘t Need It Any Longer	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
11. Others	9.9%

Source: Standish Group，350个公司，8000个项目

内容

- 需求分析
  - 概念
  - 目标
- 需求分析方法
  - 一般流程
    - 需求获取
    - 需求建模
    - 需求验证
    - 需求评审
    - 需求管理
  - 结构化分析方法
  - 非结构化方法
    - **UML**

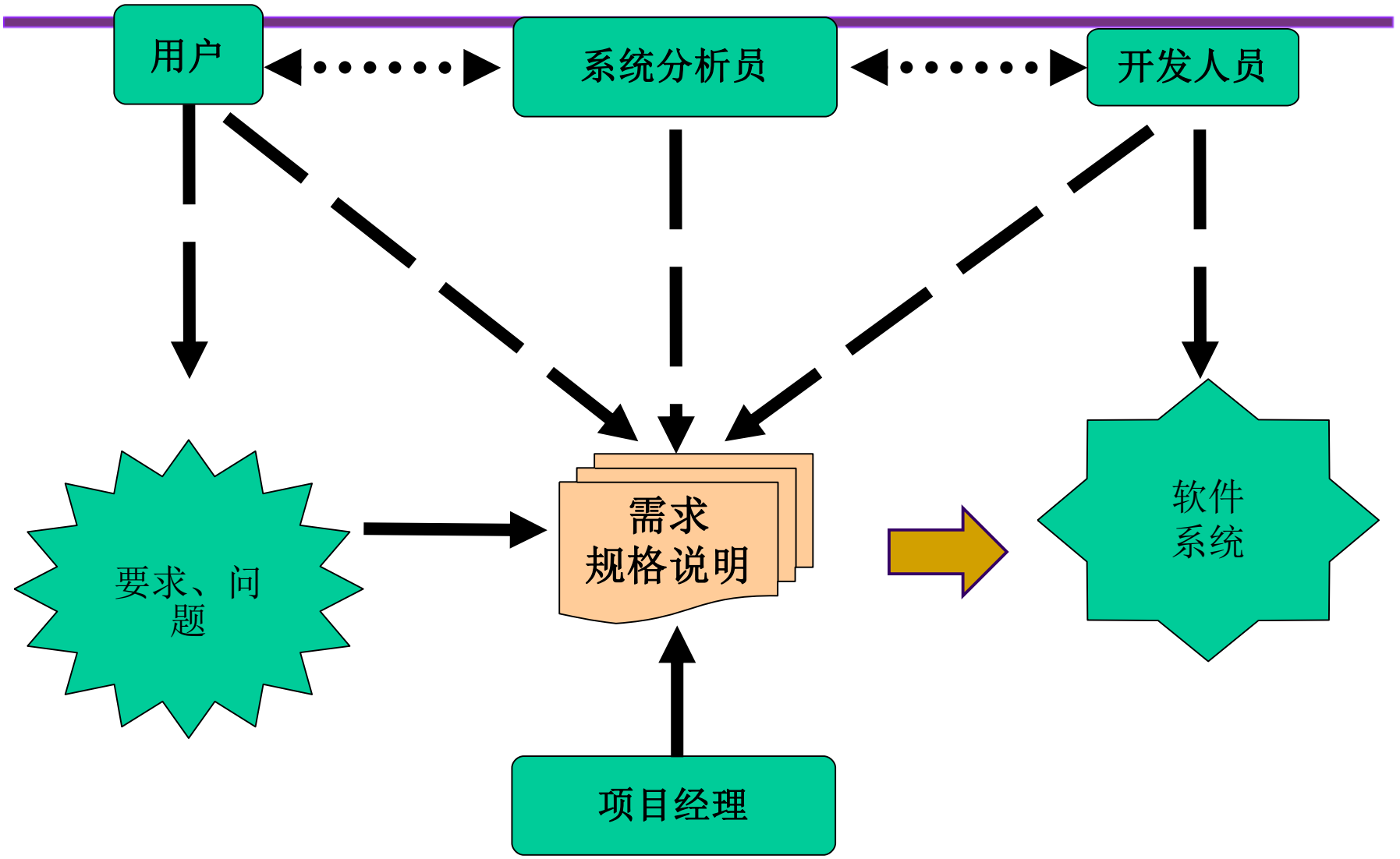
需求与需求分析

- 需求——IEEE
  - 用户解决问题或达到目的所需的条件和能力。
  - 系统或系统部件为满足合同、标准、规范或其他正式规定文档所需具有的条件和能力。
  - 一种反映上述两条所描述的条件或能力的文档说明。

需求	需求分析
是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。	通过对应问题及其环境的理解分析，为问题涉及的信息、功能及系统行为建立模型，将用户需求精确化、完全化，最终形成需求规格说明。
用户角度	系统角度

时空特性、边界特性

总体过程



需求分析目标

- 达成一致
  - **客户和其他涉众**在系统工作内容方面
  - **系统开发人员**清楚地了解系统需求
- 形成规格说明
  - 描述用户对系统的需求
  - 定义系统边界
    - **哪些做、哪些不做**
  - 定义用户界面
  - 估算开发成本
    - **人月**
    - **风险**

需求的三个层次

- 业务需求
  - 对系统、产品高层次的目标要求。
- 用户需求
  - 描述用户使用软件需要完成的任务。
- 功能/非功能需求
  - 定义了开发者必须实现的软件功能、满足的约束和条件。

业务需求

项目文档

用户需求

用例文档

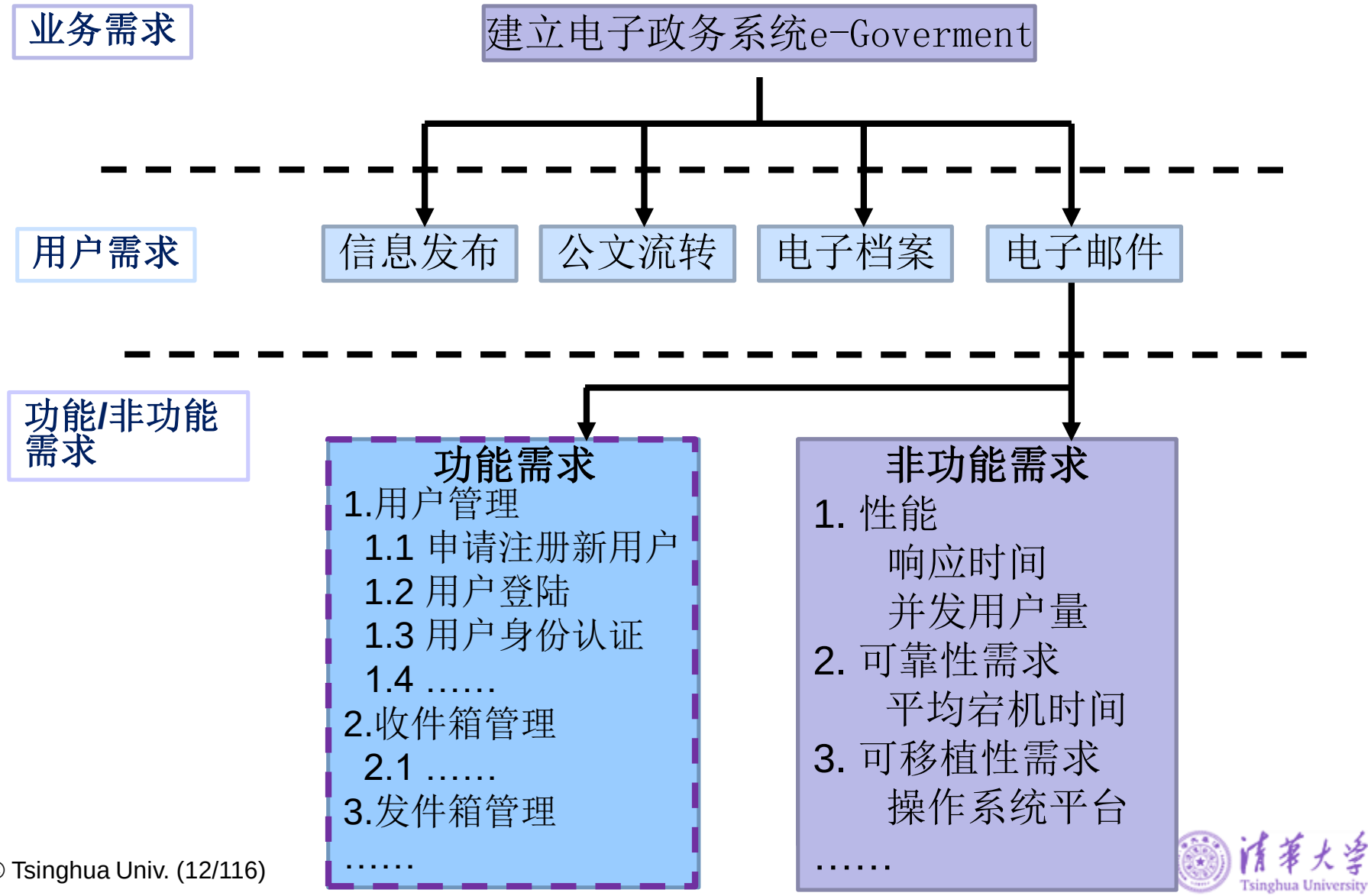
功能需求

非功能

约束条件

需求
规格说明

案例——电子政务


业务需求
建立电子政务系统e-Goverment
用户需求
信息发布
公文流转
电子档案
电子邮件
功能/非功能需求
功能需求
1.用户管理
1.1 申请注册新用户
1.2 用户登陆
1.3 用户身份认证
1.4 ……
2.收件箱管理
2.1 ……
3.发件箱管理
……
非功能需求
1. 性能
响应时间
并发用户量
2. 可靠性需求
平均宕机时间
3. 可移植性需求
操作系统平台
……

非功能性需求度量

属性	度量
速度、性能	每秒事务处理的数目 每个事件的响应时间 屏幕刷新时间
大小	内存占用大小
可用性	培训时间 帮助支持的详尽程度
可靠性	宕机次数、平均故障时间
健壮性	故障修复时间 故障的破坏程度
可移植性	平台相关的功能/程序量 可以适应的不同平台数量

需求规格说明——IEEE SRS

1.引言

- **1.1** 目的
- **1.2** 文档约定
- **1.3** 预期的读者
- **1.4** 产品范围
- **1.5** 参考文献

2.综合描述

- **2.1** 产品的前景
- **2.2** 产品的功能
- **2.3** 用户类和特征
- **2.4** 运行环境
- **2.5** 设计和实现限制
- **2.6** 假设和依赖

3. 外部接口需求

- **3.1** 用户界面
- **3.2** 硬件接口
- **3.3** 软件接口
- **3.4** 通信接口

IEEE SRS (续1)

4. 系统特性 （功能）

4.1 特征1

4.1.1 说明和优先级

4.1.2 激励/响应序列

4.1.3 功能需求

4.2 特征2

4.2.1 说明和优先级

4.2.2 激励/响应序列

4.2.3 功能需求

5. 其他非功能需求

5.1 性能需求

5.2 安全设施需求

5.3 安全性需求

5.4 软件质量属性

5.5 业务规则

5.6 用户文档

6. 其他需求

附录A 词汇表

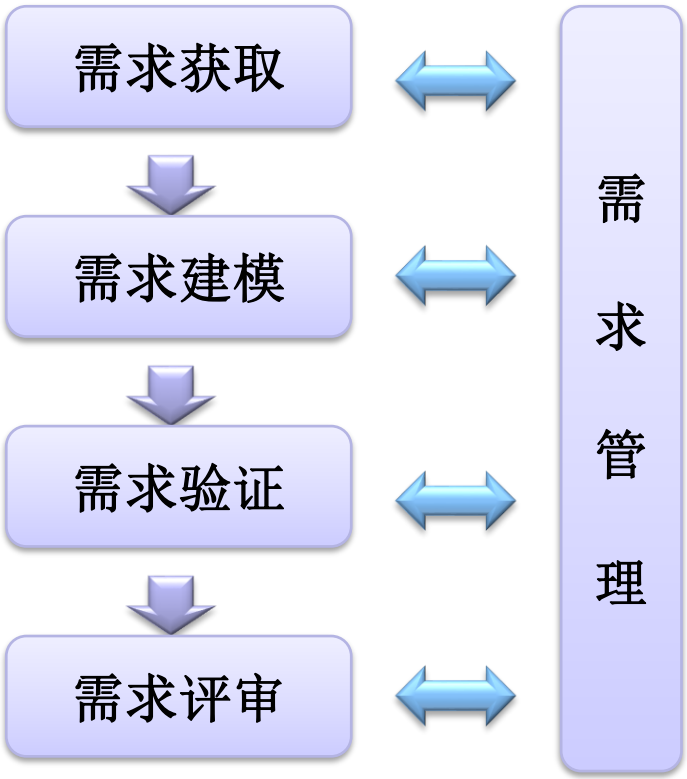
附录B 分析模型

附录C 待确定问题

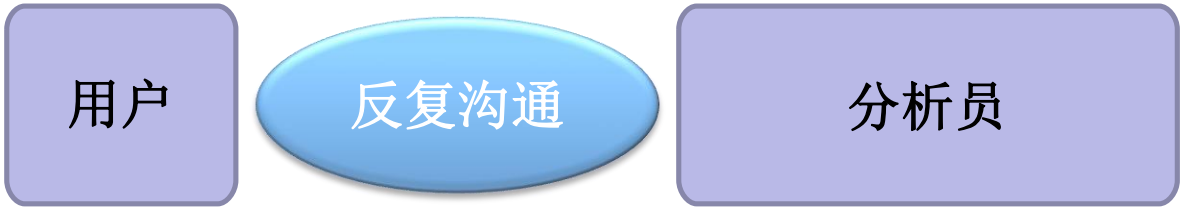
内容

- 需求分析
  - 概念
  - 目标
- 需求分析方法
  - 需求获取
    - 方法
    - 主要风险
  - 需求建模
  - 需求验证
  - 需求评审
  - 需求管理

需求分析方法



需求获取



主要工作：

- 归纳、整理用户提出的各种问题和要求
- 明确用户要求和条件
- 从非形式化的需求陈述中提取用户的实际需求
- 形成软件需求的文档化描述

主要方法

- 调查研究
  - 了解系统需求
  - 市场调查
  - 访问用户及用户领域专家
  - 考察现场、观察用户
  - 调查问卷
  - 召开会议
  - 个别咨询
  - 参考资料
- 培训
  - 需求分析员、开发人员业务培训
  - 客户培训

需求获取的主要风险

- 不全面
- 误解、不充分了解
  - 举例：“透明”
- 交流障碍
- 缺乏共同语言（“常识”问题）
- 需求不明确或不稳定
- 用户意见不一致
- 错误的要求
- 认识混淆
- 组织问题

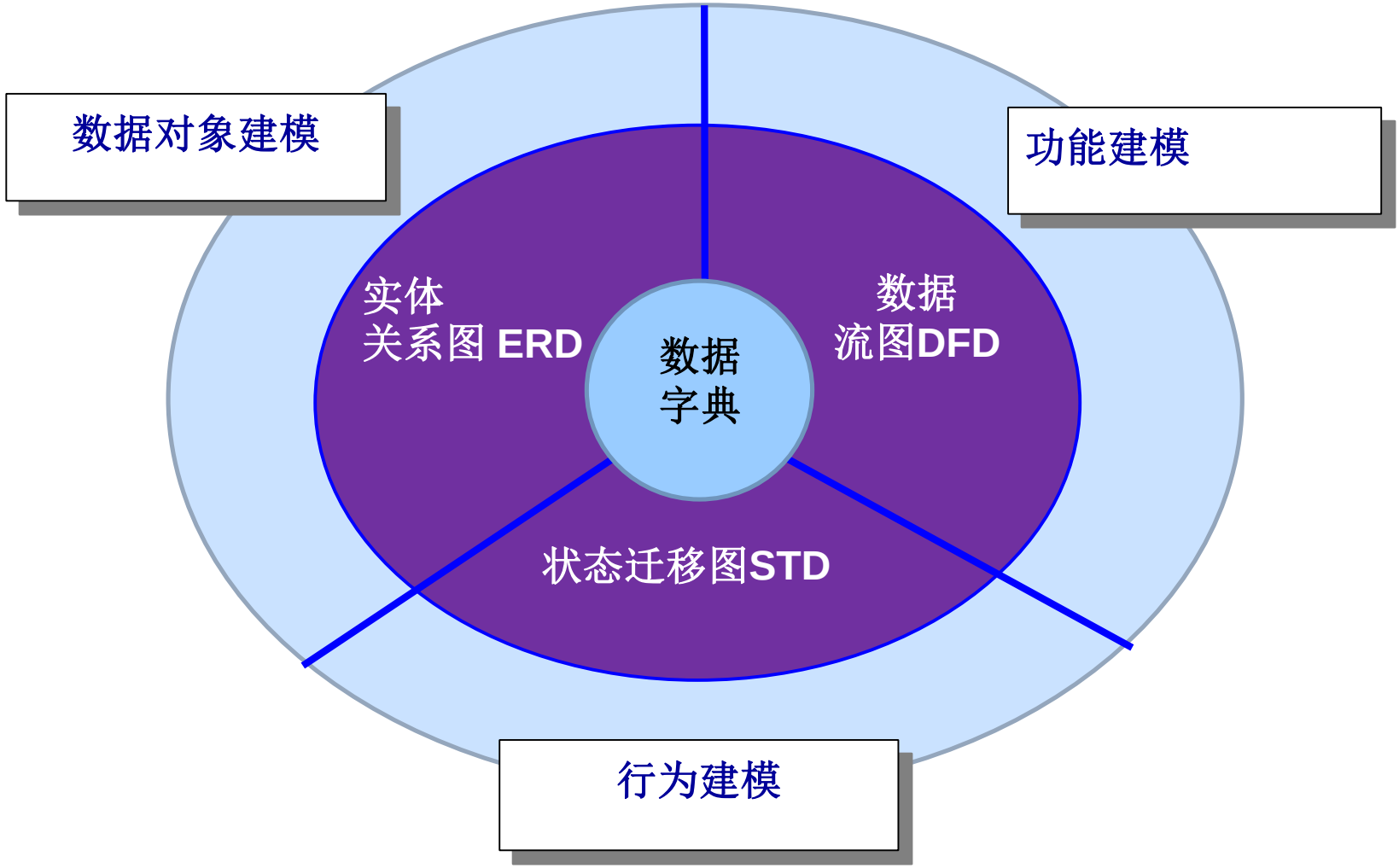
内容

- 需求分析
  - 概念
  - 目标
- 需求分析方法
  - 需求获取
  - 需求建模 （例：结构化分析方法）
    - 数据对象建模
      - » 实体关系图（ERD）
    - 功能建模
      - » IPO和数据流图（DFD）
    - 行为建模
      - » 状态迁移图（STD）
    - 数据字典
  - 需求验证
  - 需求评审
  - 需求管理

需求建模

- 任务
  - 分析和综合，细化软件功能
  - 确定系统的各种限制和约束条件
  - 分析各种功能是否要求合理
  - 根据功能/性能/运行环境等各方面的需求，剔除不合理的部分，综合成目标系统的逻辑模型
- 关键
  - 明确，减少二义性
  - 可验证、可追踪
  - 坚持“需求”而不是“设计”
  - 原型

建模方法——结构化分析

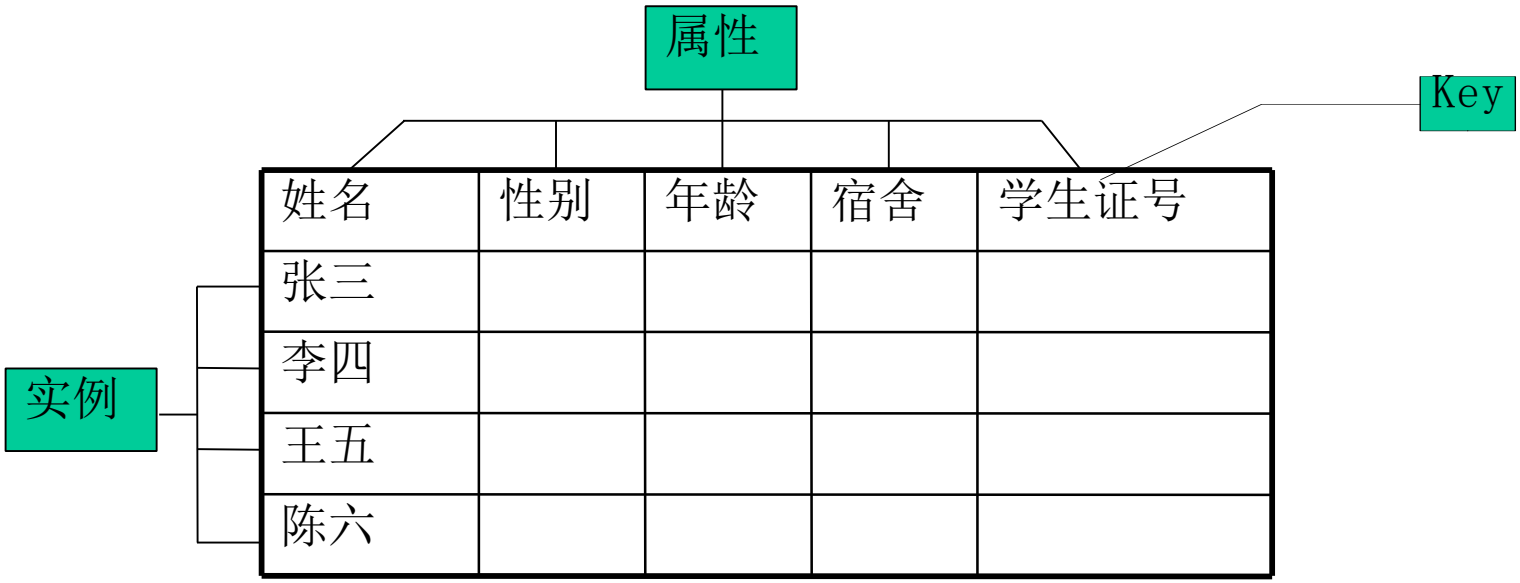


数据对象建模

- 数据模型包括三种互相关联的信息：
  - 数据对象（Data Object）
    - 实体，如事物、角色、行为或事件、组织或机构、地点等
    - 只封装数据，而不包括数据上的操作
  - 对象属性（Attribute）
    - 定义数据对象的特征
    - 关键码（Key）
  - 对象间关系（Relationship）
    - 对象之间的关联
    - 基数（Cardinality），数量对应关系
    - 参与度（Modality）
- ERD图
  - E:数据对象+属性
  - R：对象关系

ERD数据描述举例

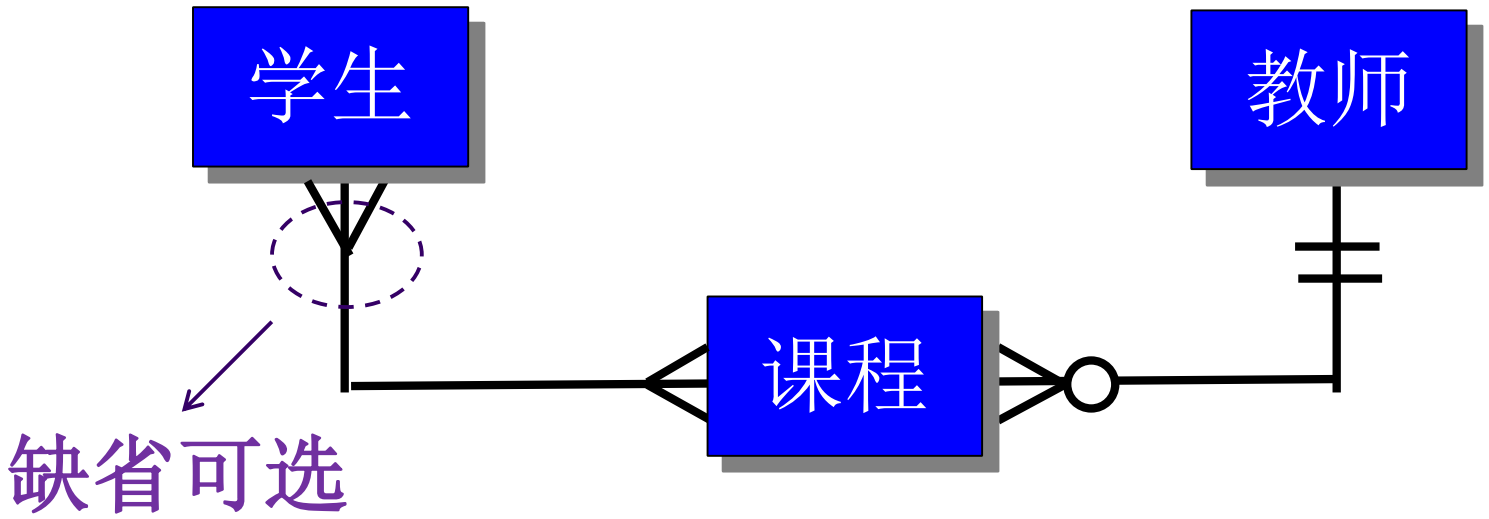
数据对象：学生



ERD对象关系

基数:一位教师　　基数:多位学生

学生选课系统ERD对象关系



学生<->课程： n:n

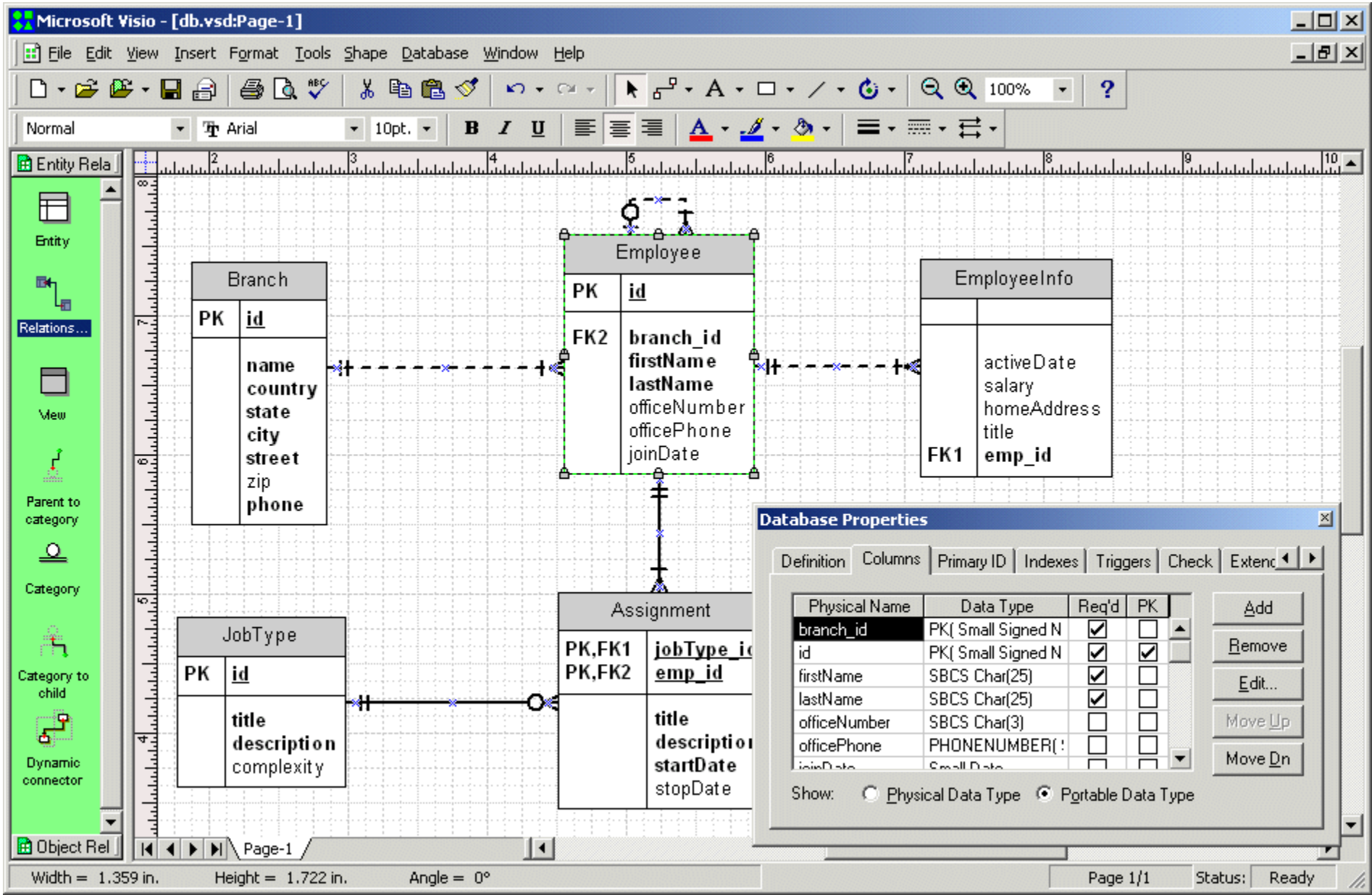
老师<->课程：1：[0, n]

学生<->教师:？

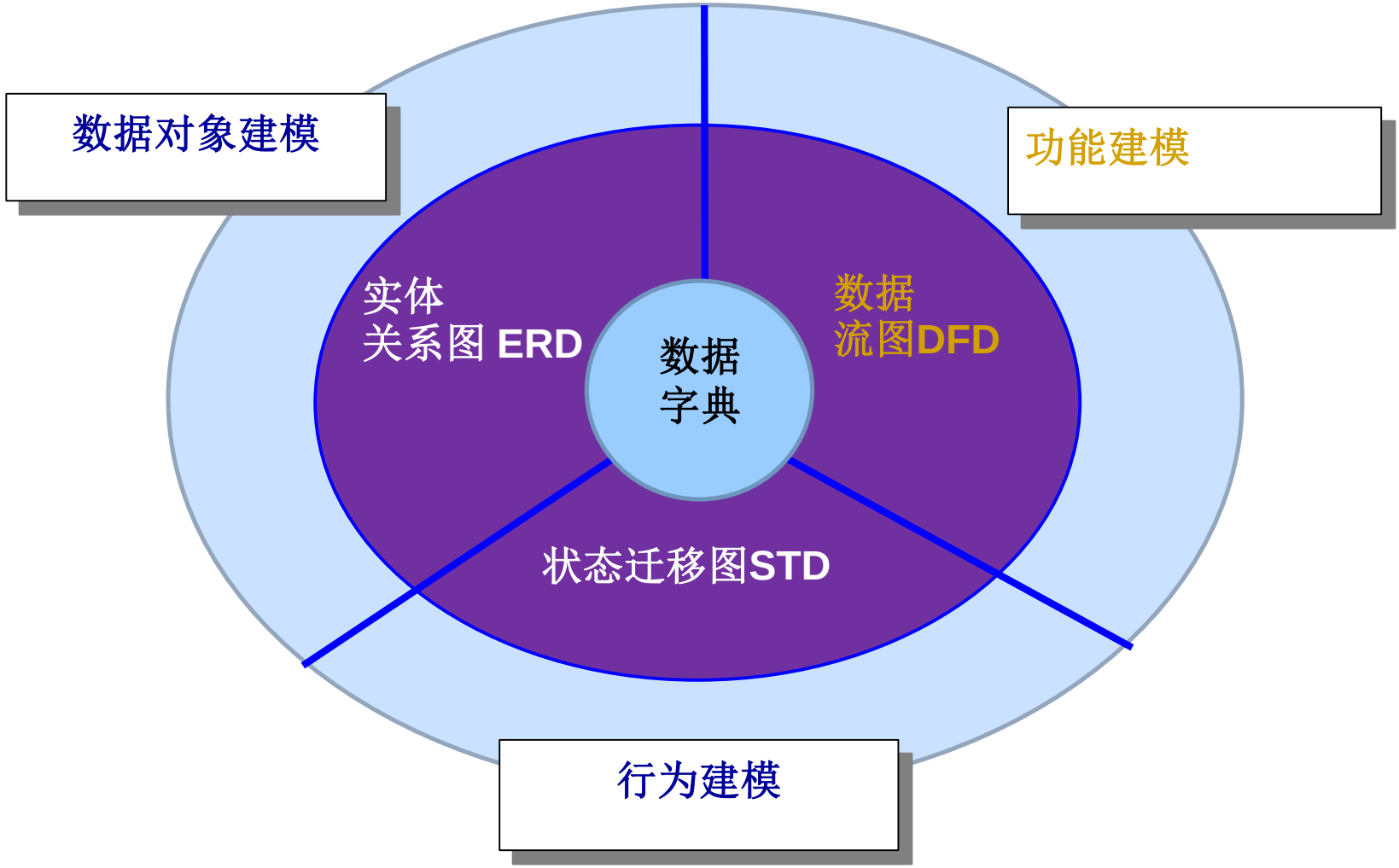
ERD工具

- ASCII SQL TEXT
- DBMS
  - Sybase PowerDesigner
  - Oracle Designer
  - Mysql
- 第三方
  - MS Visio
  - ERWin
  - IBM Rational Rose

ERD 工具


Microsoft Visio - [db.vsd:Page-1]
File Edit View Insert Format Tools Shape Database Window Help
Normal Arial 10pt.
Entity Rela
Entity
Relations...
View
Parent to category
Category
Category to child
Dynamic connector
Object Rel
Branch
PK id
name
country
state
city
street
zip
phone
Employee
PK id
FK2 branch_id
firstName
lastName
officeNumber
officePhone
joinDate
EmployeeInfo
activeDate
salary
homeAddress
title
FK1 emp_id
JobType
PK id
title
description
complexity
Assignment
PK,FK1 jobType_id
PK,FK2 emp_id
title
description
startDate
stopDate
Database Properties
Definition Columns Primary ID Indexes Triggers Check Extenc
Physical Name | Data Type | Req'd | PK
branch_id | PK(Small Signed N
id | PK(Small Signed N
firstName | SBCS Char(25)
lastName | SBCS Char(25)
officeNumber | SBCS Char(3)
officePhone | PHONENUMBER(
Add
Remove
Edit...
Move Up
Move Dn
Show: Physical Data Type Portable Data Type
Page-1
Width = 1.359 in. Height = 1.722 in. Angle = 0°
Page 1/1 Status: Ready

结构化分析

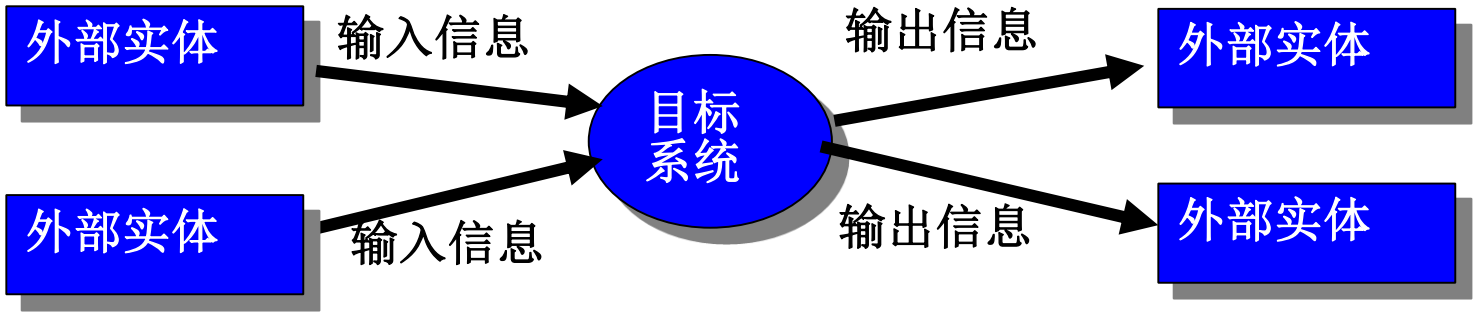


功能建模

- 定义
  - 使用抽象模型
  - 按照软件内部数据传递、变换的关系，自顶向下逐层分解
  - 直到达到满足功能要求的所有可实现的模块
- 方法
  - 数据流图（数据传递）
    - IPO图
    - 数据流图（DFD）
  - 数据处理加工描述（数据变换）
    - 结构化英语
    - 判定表
    - 判定树

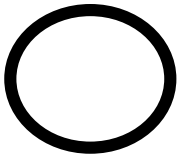
IPO图

- IPO图
  - 描述输入-处理-输出的关系
  - 描述业务的良方



DFD图

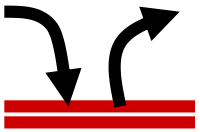
- 从数据传递和加工的角度，以图形的方式刻画数据从输入到输出的移动变换过程。



数据加工 (数据变换)

数据输入的源点或数据输出的汇点

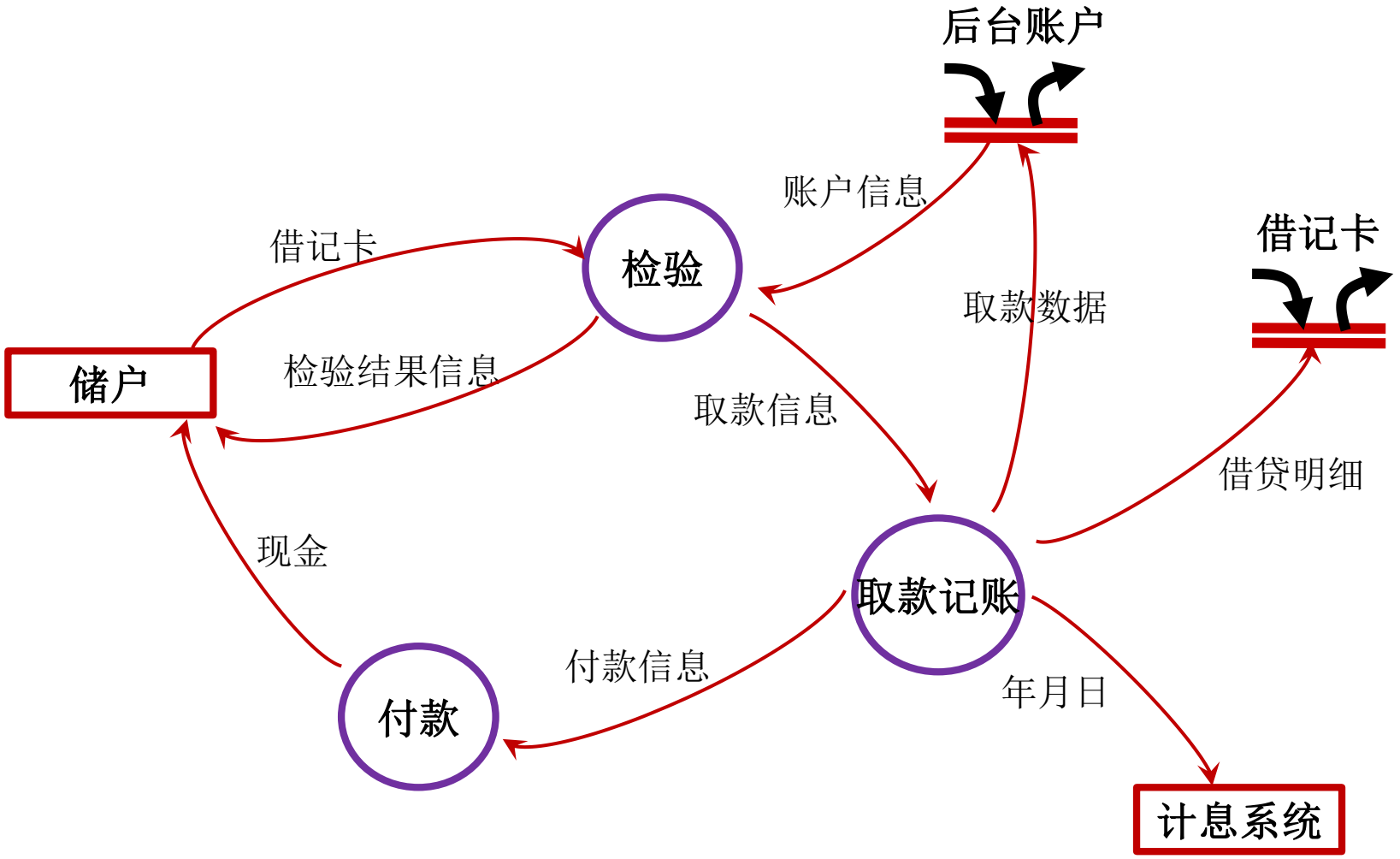
数据流



数据存储文件

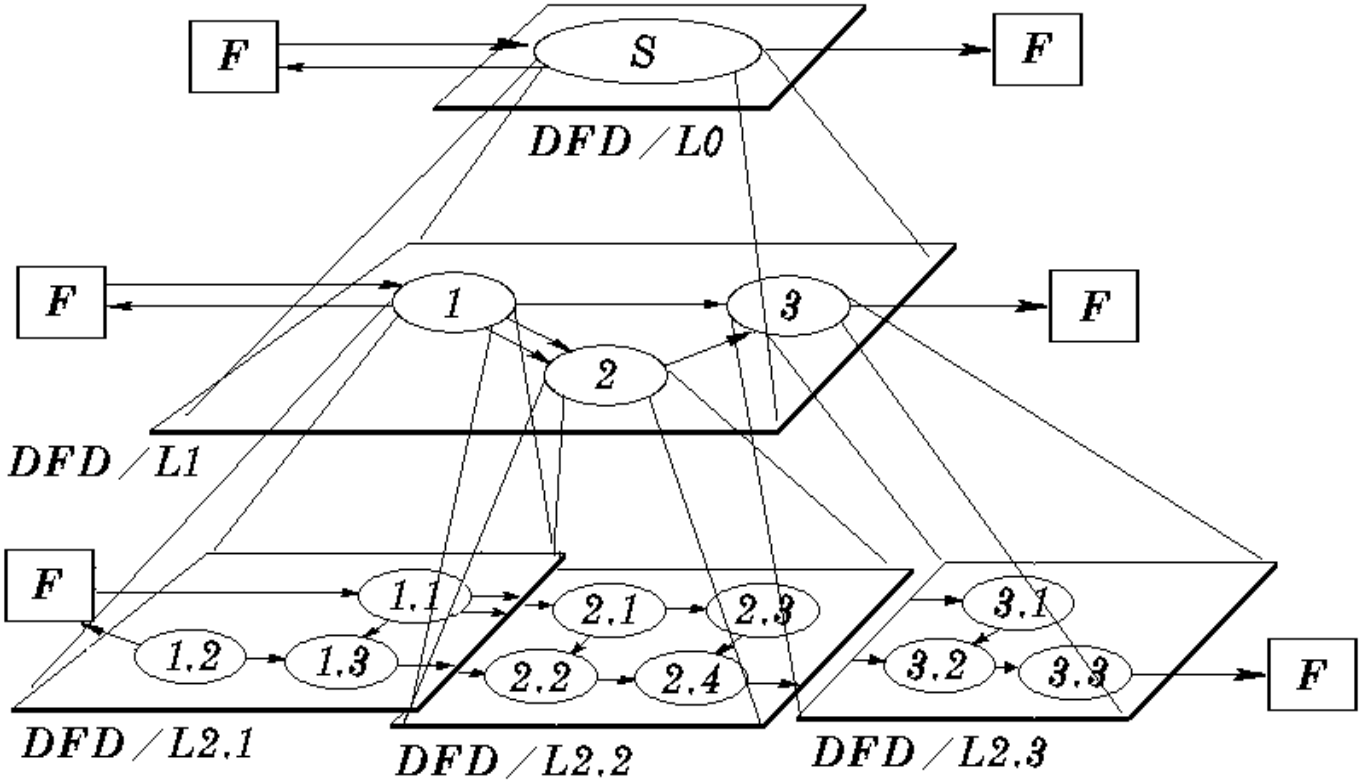
数据流图示例——银行ATM取款

数据流图示例——银行ATM取款



分层数据流图

- 分层DFD
  - 自顶向下逐步分解



DFD步骤

- 画出顶层DFD
  - IPO图
- 顶层DFD
  - 从输入端开始，根据业务工作流程，画出数据流流经的各加工框，逐步画到输出端，得到第一层数据流图
- 底层DFD
  - 逐步细化每一个加工，逐层分解，直至底层DFD

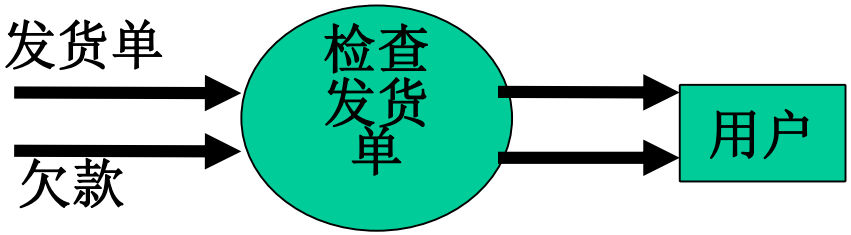
基本加工逻辑说明

- 基本加工的逻辑说明
  - 说明把输入数据流变换为输出数据流的加工规则
  - 描述实现加工的策略而不是实现加工的细节
  - 包含的信息应是充足的，完备的，有用的，无冗余的
- 主要方法
  - 结构化英语
  - 判定表 (Decision Table)
  - 判定树 (Decision Tree)

结构化英语

- 介于自然语言与形式化语言之间
- 正文用基本控制结构进行分割
  - 简单陈述句结构
  - 重复结构：*while_do*；*repeat_until*
  - 判定结构：*if_then_else*；*case_of*
- 加工的操作用自然语言短语表示
  - 精确，避免二义性

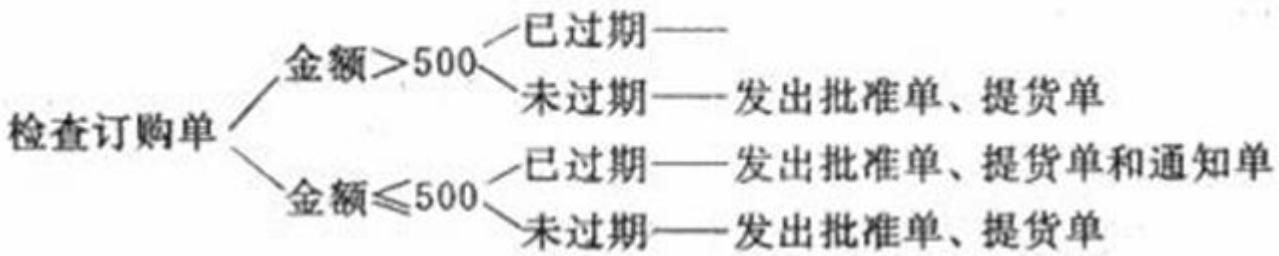
加工逻辑示例



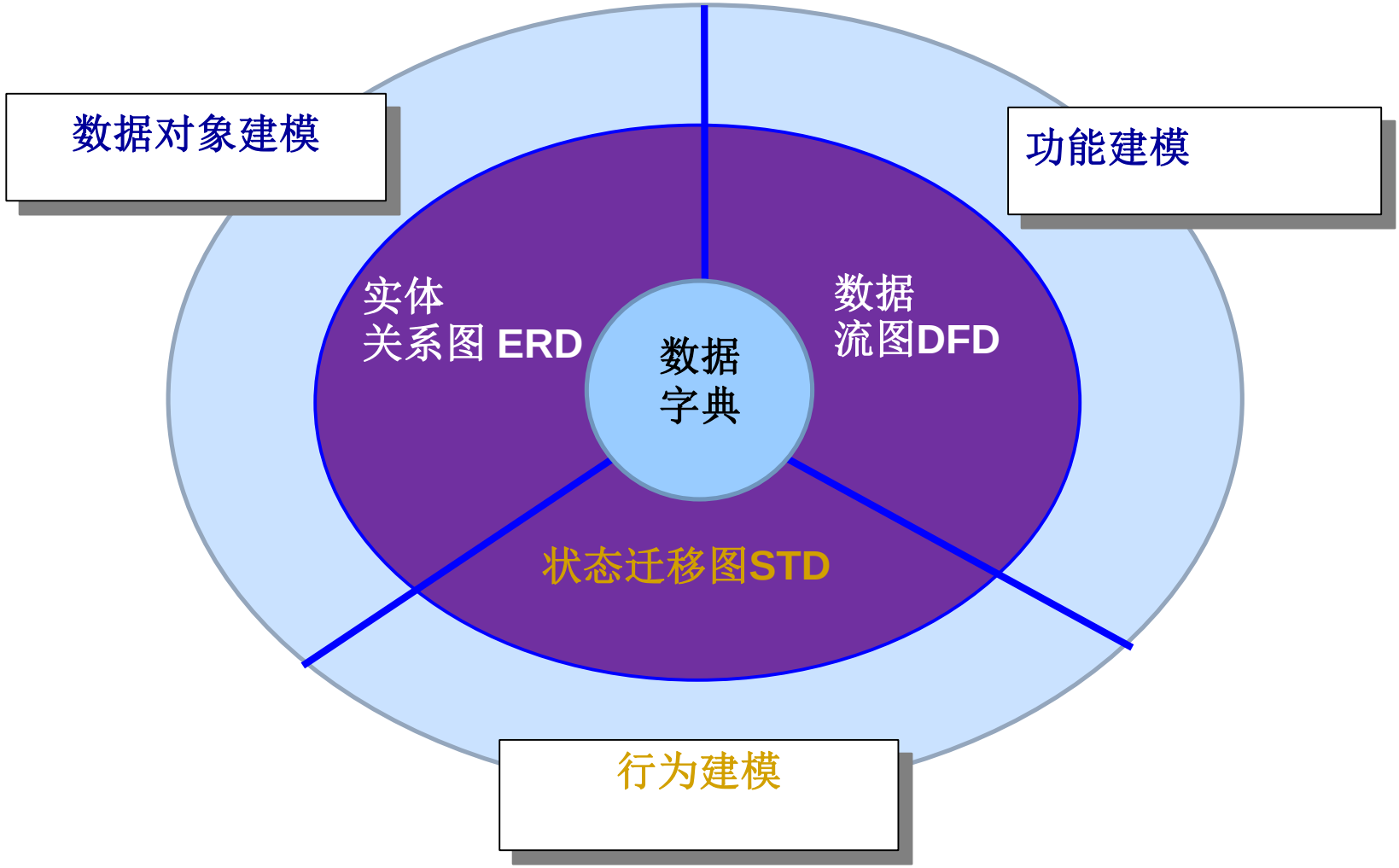

```
if 发货单金额超过$500 then
    if 欠款超过了60天 then
        在偿还欠款前不予批准
    else （欠款未超期）
        发批准书，发货单
else （发货单金额未超过$500）
    if 欠款超过60天 then
        发批准书，发货单及赊欠报告
    else （欠款未超期）
        发批准书，发货单
```

判定表/树

		1	2	3
条件	赊欠情况	>60天	>60天	≤60天
	发货单金额	>$500	≤$500	——
操作	不发出批准书	√		
	发出批准书 发出发货单		√	√
	发出赊欠报告		√	



结构化分析



行为建模

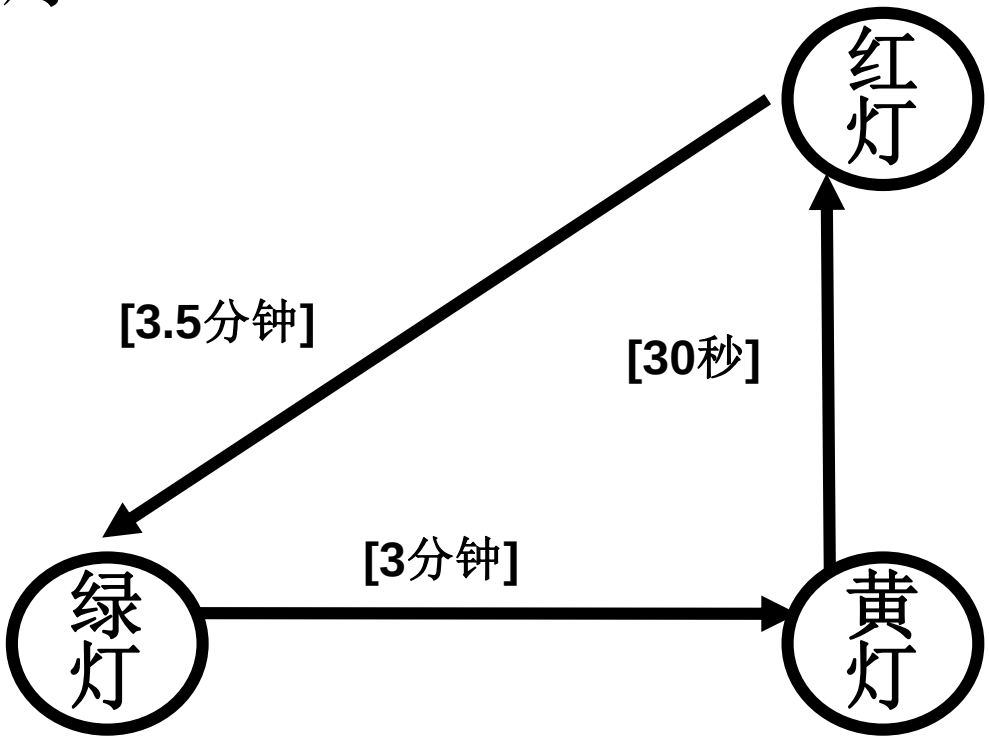
- 描述系统的动态特性
  - 状态迁移图，STD（State Transition Diagram）
    - 描述系统如何响应外部的信号，进行状态迁移

系统状态

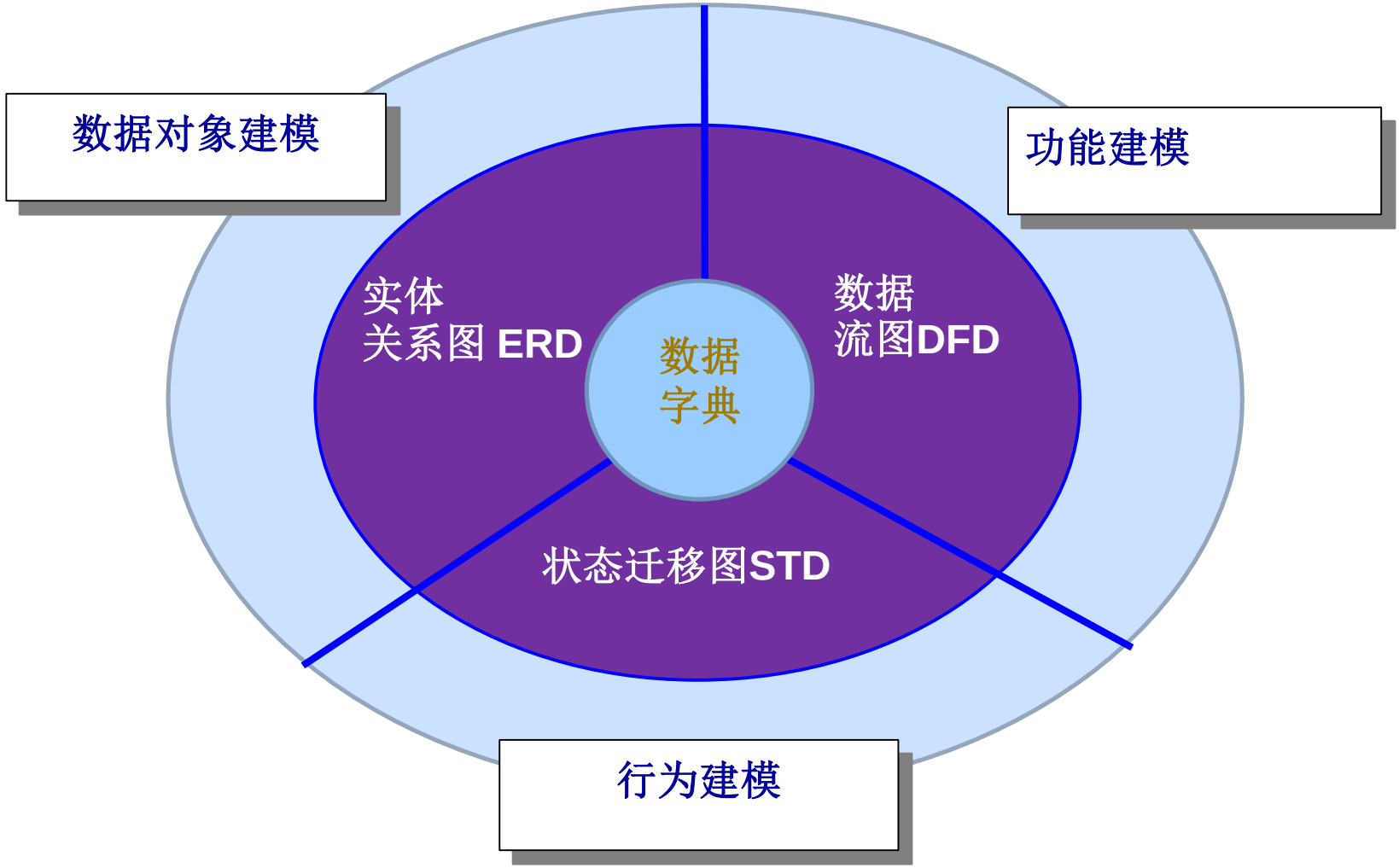
从一种状态到另一种状态的转移

状态迁移图示例

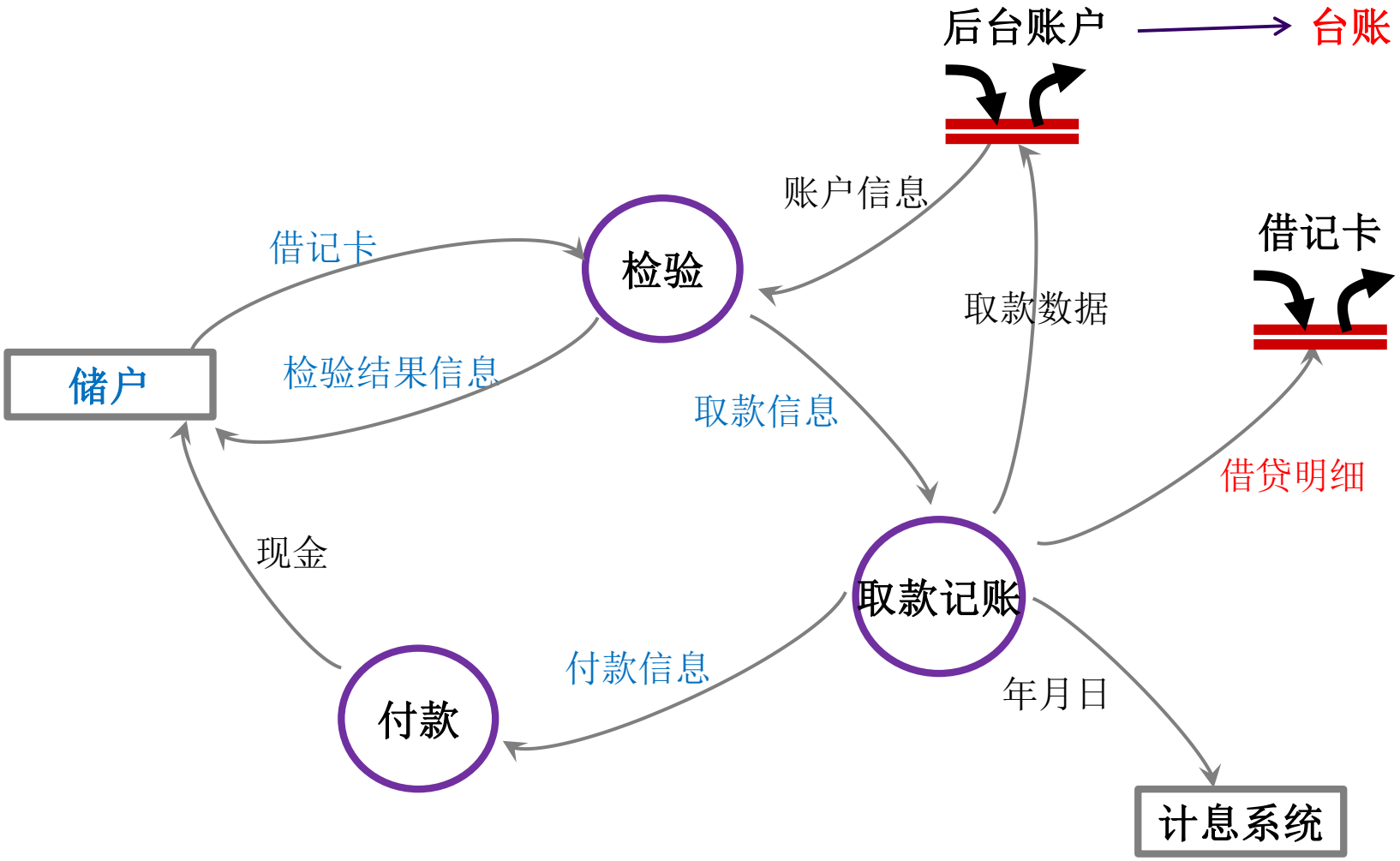
- 红绿灯



结构化分析



数据字典


后台账户
台账
账户信息
借记卡
检验
借记卡
储户
检验结果信息
取款数据
取款信息
借贷明细
现金
取款记账
付款信息
年月日
付款
计息系统

数据字典

- 目的
  - 对数据流图中出现的所有被命名的图形元素在数据字典中作为一个词条加以定义，使得每一个图形元素的名字都有一个确切的解释。
- 作用
  - 查询、检索、统计分析各种数据
  - 检查需求定义与软件设计实现之间的完整性和一致性
  - 软件整个生命周期活动的基本依据
- 内容
  - 图形元素如数据存储、数据加工、数据流
  - 图形元素的名字、别名、编号、分类、描述、定义、位置等
  - 数据流词条描述
  - 数据元素词条描述
  - 数据文件词条描述
  - 加工逻辑词条描述
  - 源点及汇点词条描述

符号定义示例

符号	定义	举例
=	被定义为	*x* = *a*/2
+	与	*x* = *a*+*b*
[...,...] 或 [...|...]	或	或
"..."	基本数据元素	*x* = "*a*"
{ ... }或 m{...}n	重复	*x* = {*a*}，*x* =3{*a*}8
..	连结符	*x* = 1..9

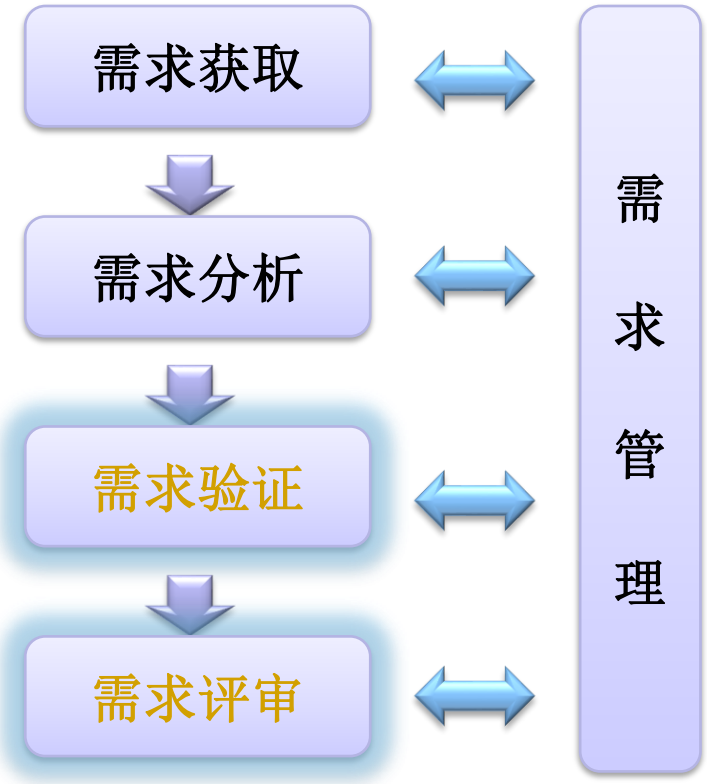
其它条目定义示例

- 数据流条目：
  - 报名单 = 姓名+单位名称+年龄+性别+课程名称
  - 课程 = 课程名称+教师+教材+课程表
  - 借记卡 = {账号+密码 }
- 文件条目：
  - 定期帐户 = { 帐号+户名+地址+款额+存期 }
  - 组织方式 = 按帐号递增次序排列
- 数据项条目：
  - 帐号 = 00000－99999； 存期 = [1 | 3 | 6 | 12]

内容

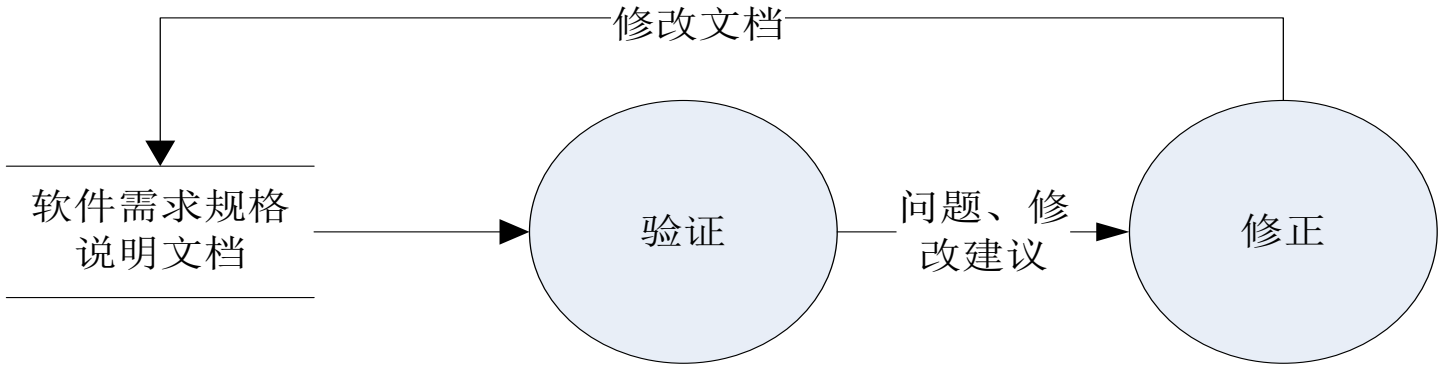
- 需求分析
  - 概念
  - 目标
- 需求分析方法
  - 需求获取
  - 需求建模
  - 需求验证
  - 需求评审
  - 需求管理

需求验证、评审



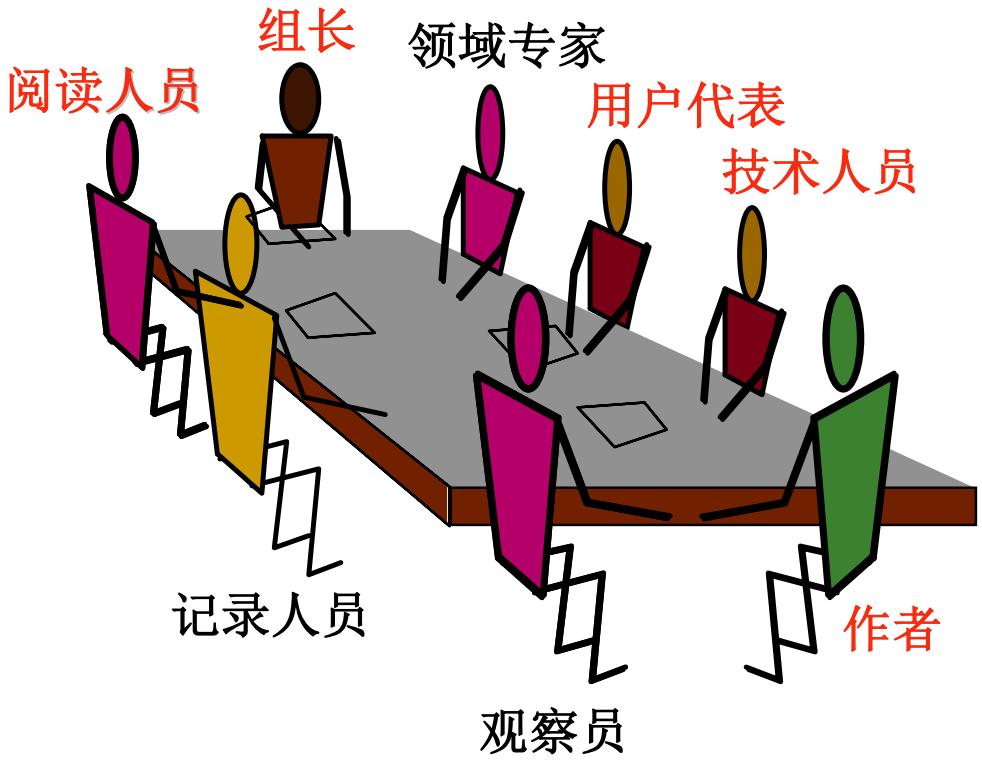
- 需求验证
  - 专指在需求规格说明完成之后，对需求规格说明文档进行的验证活动
  - 获得的用户需求是否正确和充分的支持业务需求？
  - 建立的分析模型是否正确的反映了问题域特性和需求？细化的系统需求是否充分和正确的支持用户需求？
  - 需求规格说明文档是否组织良好、书写正确？需求规格说明文档内的需求是否充分和正确的反映了涉众的意图？需求规格说明文档是否可以作为后续开发工作（设计、实现、测试等等）的基础？

需求验证

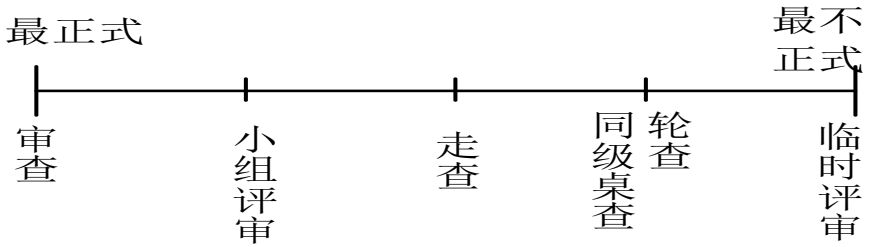


1. 评审
   - 内部评审
2. 原型与模拟
3. 开发测试用例
4. 用户手册编制
5. 利用跟踪关系
   - 业务需求→←用户需求→←系统需求
6. 自动化分析

需求评审



检查方法
自由方法（Ad-hoc）
检查清单（Checklist-Based）
缺陷（Defect-Based）
功能点（Function Point-Based）
视角（Perspective-Based）
场景（Scenario-Based）
逐步抽象（Stepwise Abstraction）



内容总结

- 需求分析
  - 概念
    - 定义
    - 重要性
  - 目标
    - 三个层次
    - 规格说明文档（**IEEE SRS**）
- 需求分析方法（结构化）
  - 需求获取
    - 方法
    - 风险
  - 需求分析
    - 数据对象建模
      - 实体关系图（**ERD**）
    - 功能建模
      - **IPO**和数据流图（**DFD**）
    - 行为建模
      - 状态迁移图（**STD**）
    - 数据字典
- 需求验证、评审、管理

需求分析方法

- 结构化分析（Structured Analysis）
- 面向对象分析（非结构化）（Object Oriented Analysis）

需求分析方法之面向对象

- 面向对象分析（Object Oriented Analysis）
  - 1950s，LISP，动态绑定(Dynamic Binding)及交互式开发环境
  - 1960s，SIMULA 67，类(Class)与继承机制(Inheritance)
  - 1970s，MODULA-2、Ada，抽象数据类型(Abstract Data Type)
  - 1971，Smaltalk-72，Xerox Inc. Palo Alto 研究中心（PARC）
    1981，Smaltalk-80
  - 1980s，Eiffel、C++
  - 1990s，Java
  - 1980s，面向对象方法学
- 面向对象 = 对象（Object）
  + 分类（Classification）
  + 继承（Inheritance）
  + 基于消息的通信（Comm. with messages）

优点

- 结构化程序设计方法的问题
  - 功能与数据相分离
  - 函数必须知道数据结构
  - 系统围绕一定的行为来进行
  - 难以维护
  - 可复用性低
- 面向对象方法的优点
  - 符合人们对客观世界的认识规律
  - 结构易于理解、扩充和修改，系统易于维护
  - 易于复用

概念

- Object

Something perceptible by one or more of the senses, especially by vision or touch; a material thing.

物体，用一种或多种感官，尤指用视觉或触觉可以感觉到的东西，实物

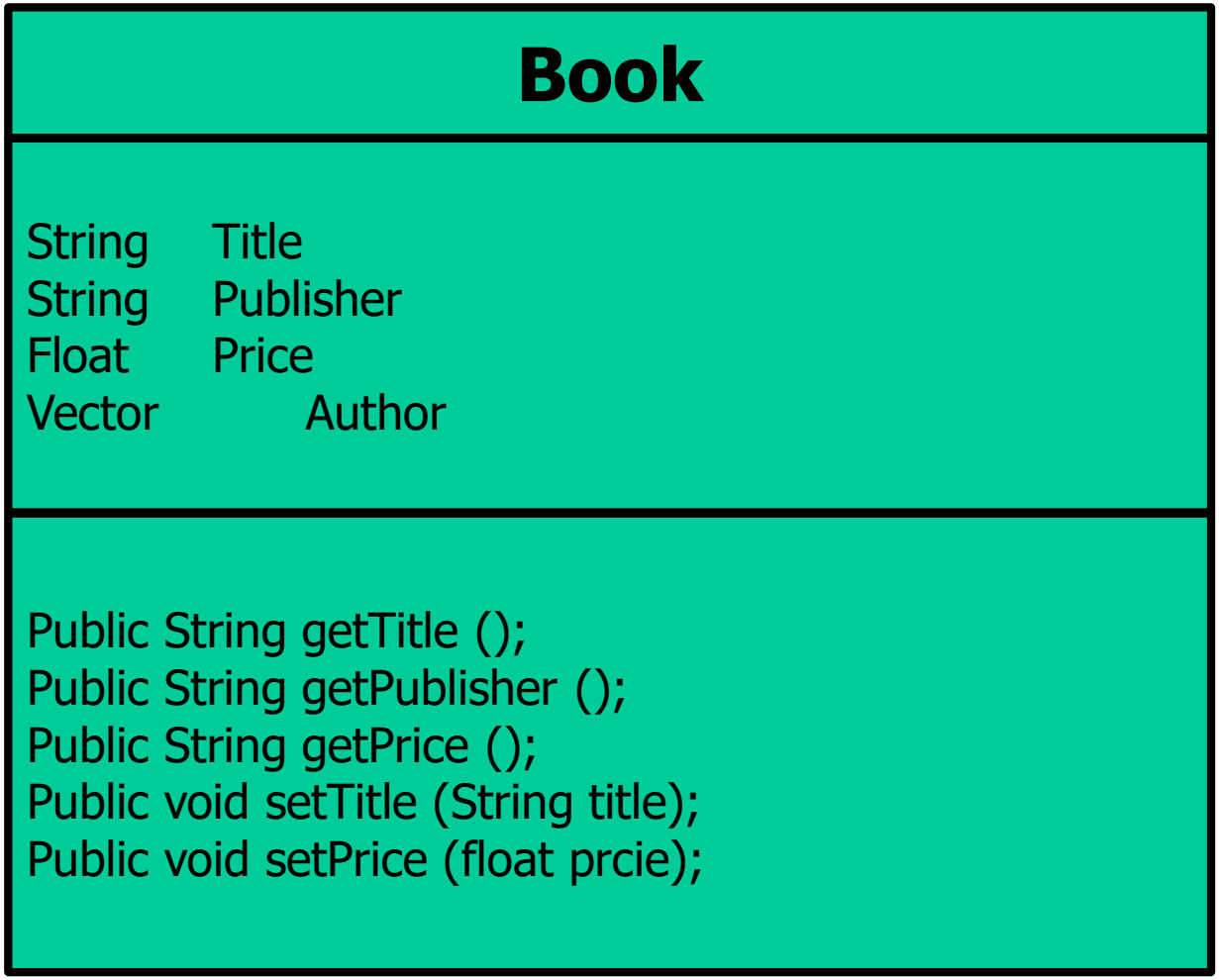
定义

- 类（Class）
  - 一组属性以及这组属性上的专有操作的封装体
  - 具有相同数据结构和相同操作的对象的集合
- 对象（Object）
  - 类的实例（Instance）
  - 属性值：唯一识别该对象
- 继承（Inheritance）
  - 类间基于层次关系共享数据和操作的一种机制
    - 子类继承其父类（或祖先类）的属性和操作
    - 子类可以对父类（或祖先类）的操作重新定义其实现方法
    - 子类可定义自己特有的属性和操作
  - 单一继承：一个子类有唯一一个父类
  - 多重继承：一个子类有多个父类

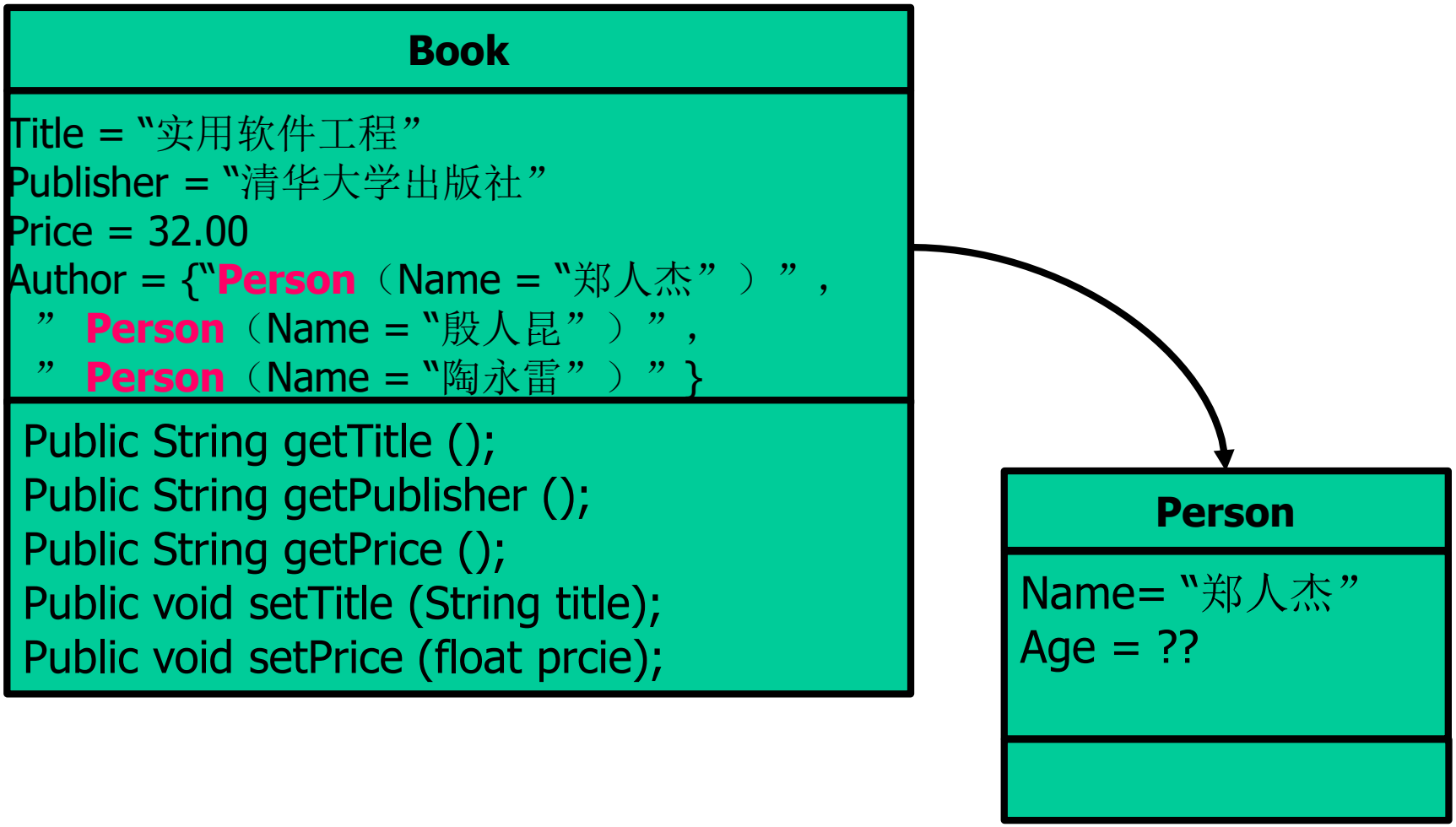
定义（续）

- 消息
  - 对象间的通信手段，通过发送消息请求服务
- 多态性（Polymorphism）
  - 同一操作作用在不同的对象上，可以有不同的解释
- 动态绑定 （Dynamic Binding）
  - 在运行过程中，将请求的操作与具体的实现连接

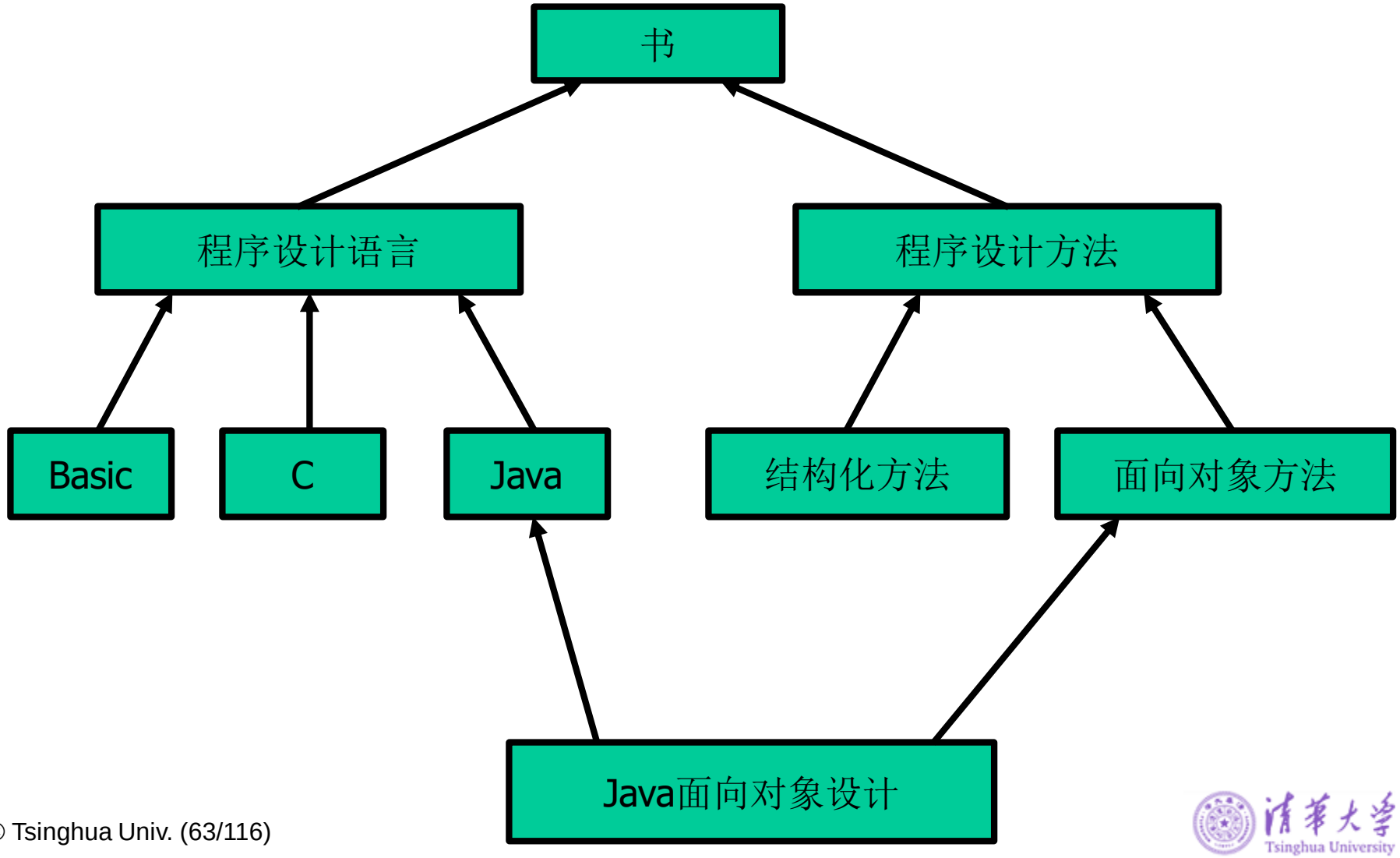
类



实例


Book
Title = "实用软件工程"
Publisher = "清华大学出版社"
Price = 32.00
Author = {"Person（Name = "郑人杰"）"，
" Person（Name = "殷人昆"）"，
" Person（Name = "陶永雷"）"}
Public String getTitle ();
Public String getPublisher ();
Public String getPrice ();
Public void setTitle (String title);
Public void setPrice (float prcie);
Person
Name= "郑人杰"
Age = ??

继承



OO 之 UML

- Unified Modeling Language： A graphical language for
  - Specifying
  - Visualizing
  - Constructing
  - Documenting

  The artifacts of software systems

- 三友（Three Amigos）
  - Booch (Grady Booch) &
  - OMT （Janes Rumbaugh) &
  - OOSE (Ivar Jacobson)

发展历程

current UML 2.4.1 @2011 by OMG

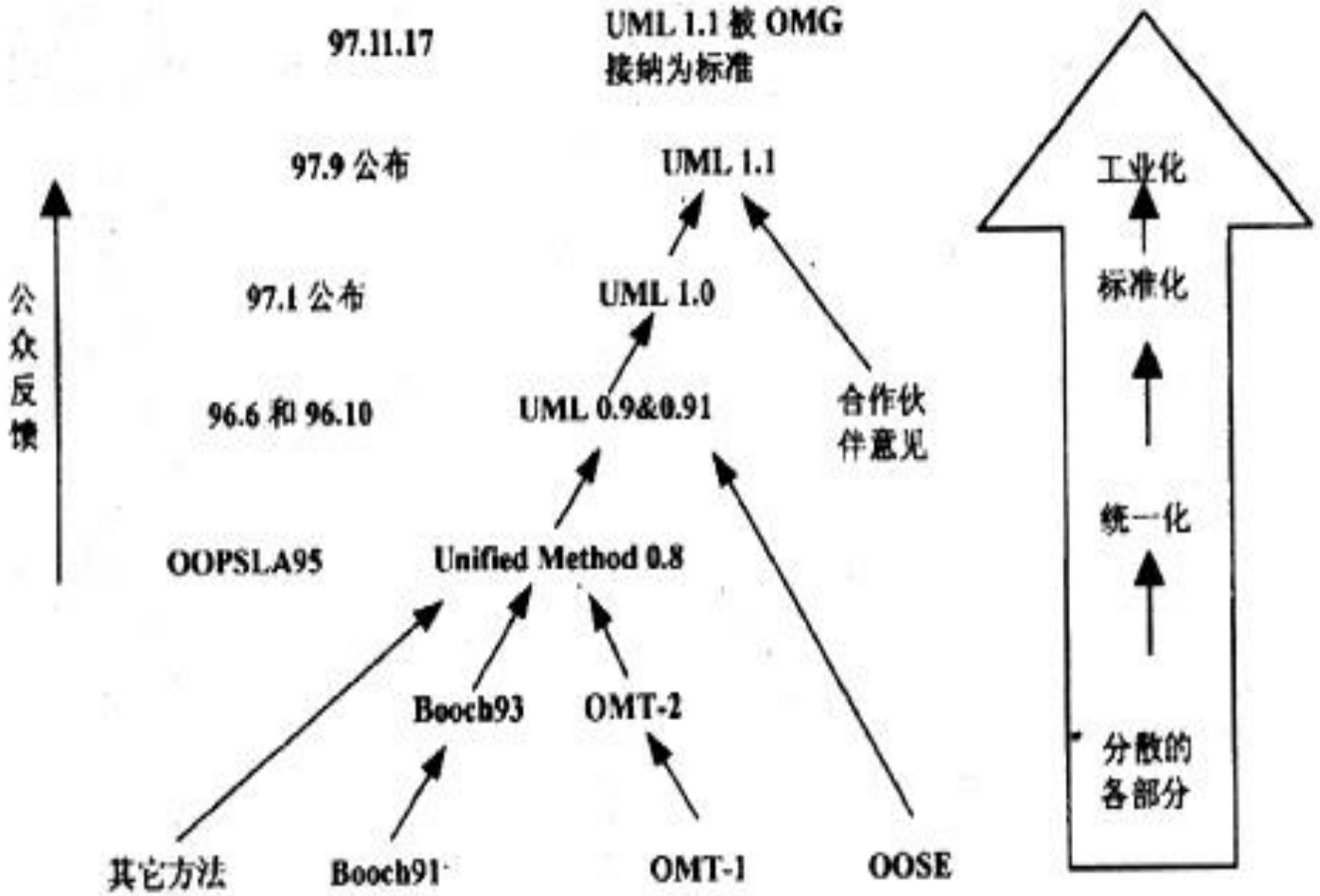
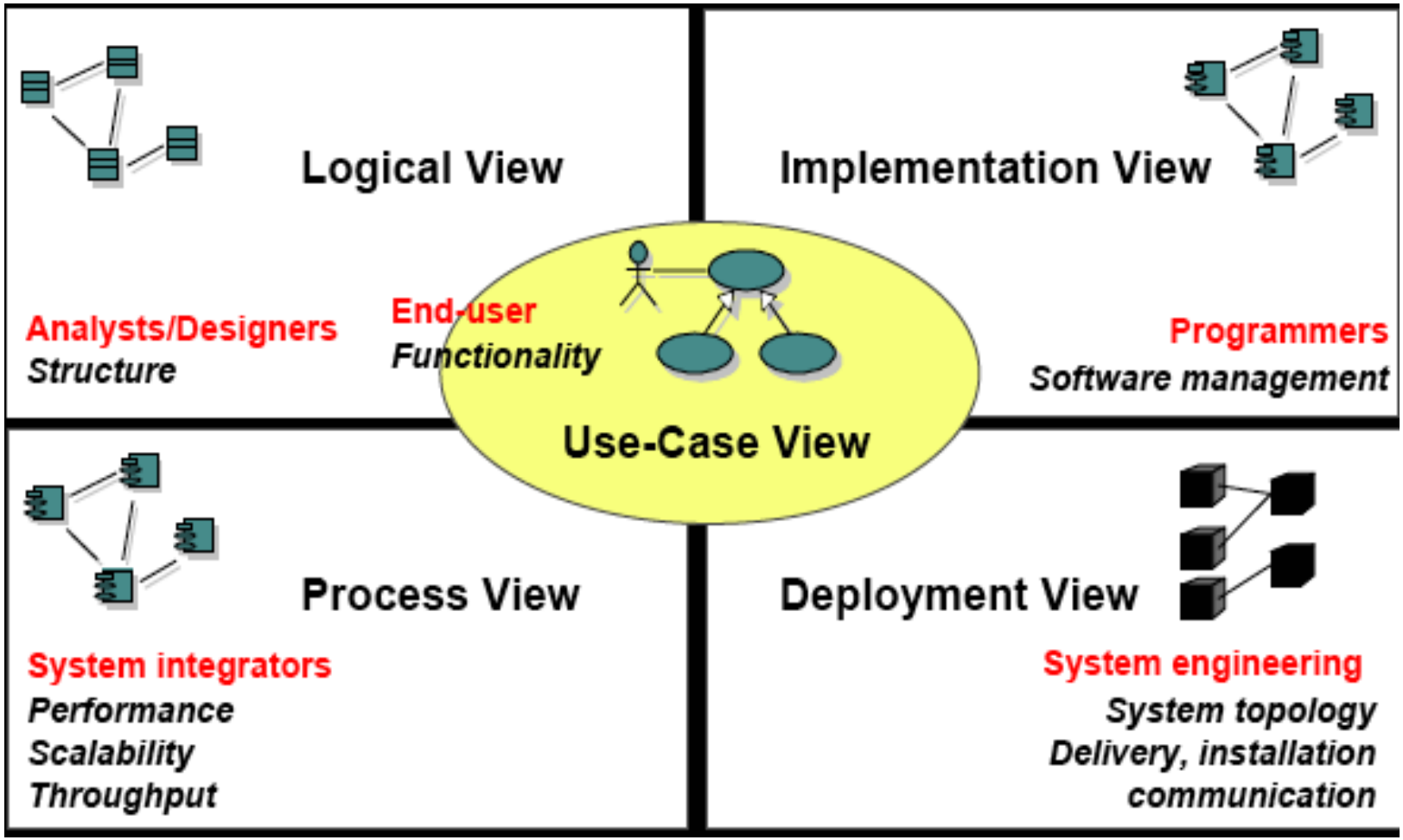


图 UML 的发展历程

UML 4+1 view



UML模型

- 静态结构模型
  - 类图
    Class diagram
  - 对象图
    Object diagram
  - 包图
    Package diagram
  - 构件图
    Component diagram
  - 部署图
    Deployment diagram
- 动态行为模型
  - 用例图
    Use Case Diagram
  - 活动图
    Activity Diagram
  - 顺(时)序图
    Sequence Diagram
  - 协作图
    Collaboration Diagram
  - 状态图
    Statechart Diagram

UML需求分析工具

- IBM Rational
  - Rose (UML)
  - RUP (Rational Unified Process)
  - SoDA (Document Process)
- 第三方UML工具
  - Violet (eclipse plugin)
  - Astah (Jude)
  - ArgoUML
  - StarUML
  - …

面向对象分析

- 运用面向对象的方法进行系统分析
  - 软件生命周期的一个阶段，具有与一般系统方法共同的目标、内容和策略
    - 深入描述软件的功能和性能
    - 确定软件设计的限制和软件同其它系统元素的接口细节
    - 定义软件的其它有效性需求
    - 给待开发的系统提供一个清晰的、一致的、精确的并且无二义性的模型
  - 强调用面向对象的概念和模型表示分析结果

任务

- What?
- 功能
  - 外部边界
  - 内部行为
- 动态行为
  - 表示类（对象）之间的关系
  - 为对象行为（消息传递、控制）建模
- 静态设计的基础
  - 标识类
  - 刻画类的层次结构

主要步骤

1. 功能建模。多方沟通，采用用例图获取、描述用户需求，采用文字或者活动图详细描述场景。
2. 行为建模。对于每一个用例图进行系统动态行为建模，采用时序图、协作图描述系统。
   （指导设计：选择类和对象、定义类的结构和层次）
3. 重复上述步骤，直至完成对象建模
4. 文档化，需求评审

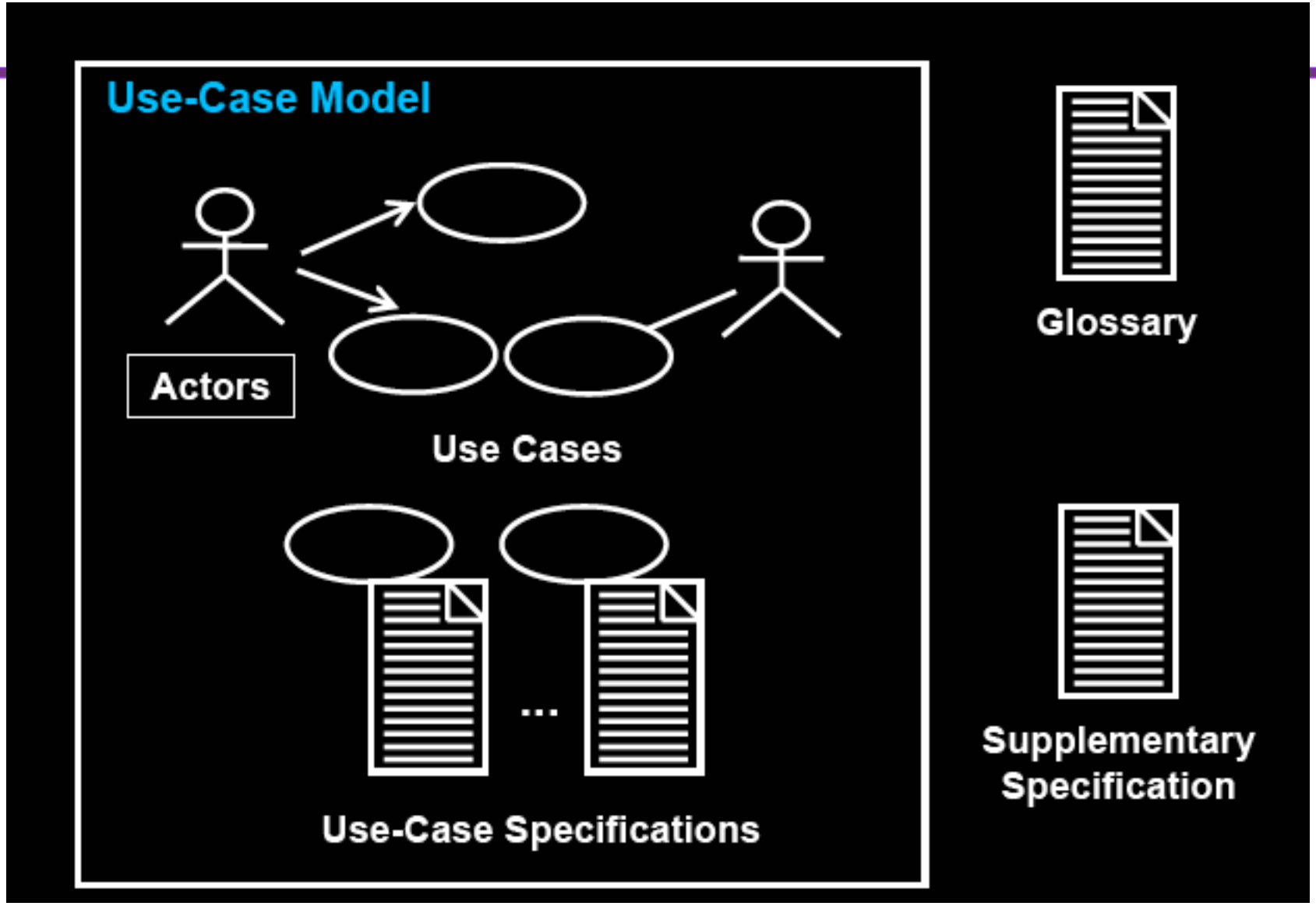
系统行为（功能）建模

- 系统行为（功能）
  - 一个系统如何执行（act）和交互（interact）
    - 内部&外部

- 系统行为用use case描述
  - 用例描述系统，环境以及二者之间的关系

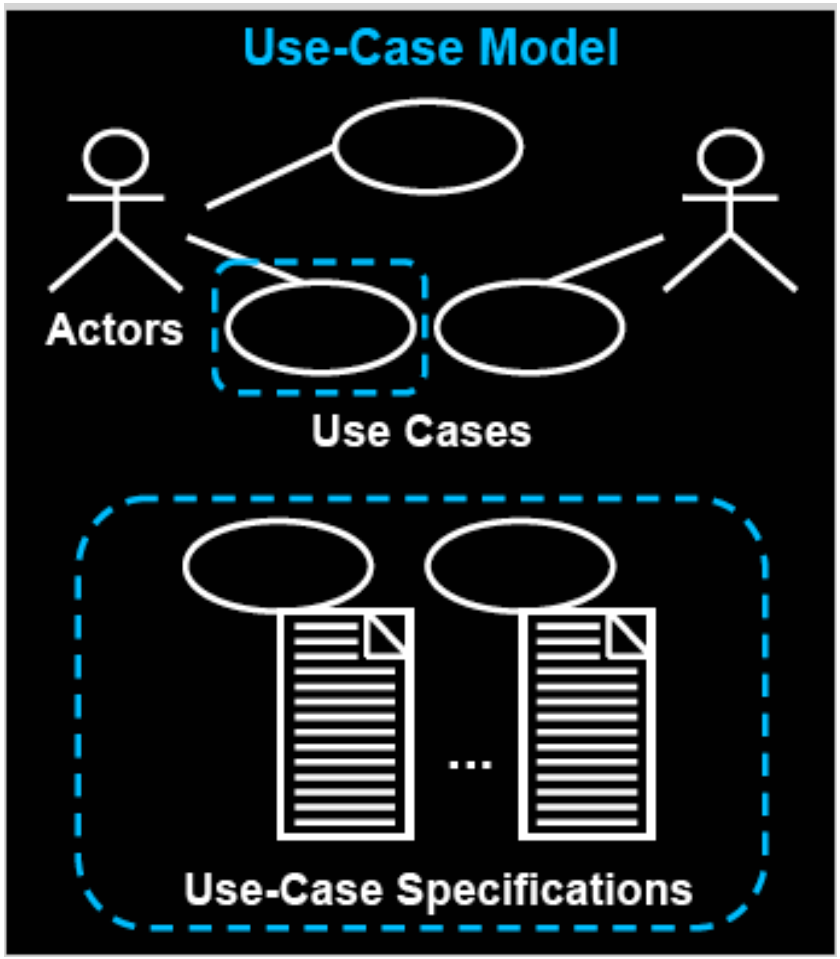
UML用例图（功能建模）

- 用例
  - 描述系统的功能（功能建模）
  - 软件开发以后所有活动都将追溯到用例
- 行为者（Actor）
  - 与系统交互的人或其他系统
- 系统
  - 提供用例的黑盒



用例描述（Specification）

- 名字
- 简介
- 角色
- 前置条件
- 后置条件
- 基本流
- 活动图
- 时序图
- 特殊需求
- 其它图



Glossary

Glossary

Course Registration System Glossary

1. Introduction

This document is used to define terminology specific to the problem domain, explaining terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal *data dictionary*, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information.

2. Definitions

The glossary contains the working definitions for the key concepts in the Course Registration System.

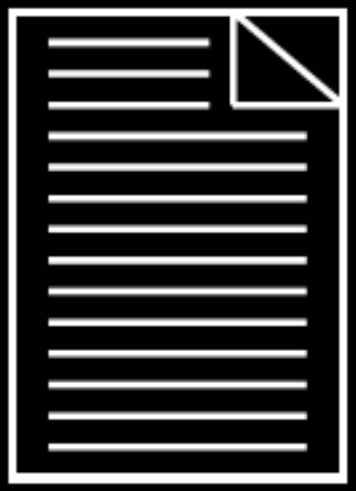
2.1 Course: A class offered by the university.

2.2 Course Offering: A specific delivery of the course for a specific semester – you could run the same course in parallel sessions in the semester. Includes the days of the week and times it is offered.

2.3 Course Catalog: The unabridged catalog of all courses offered by the university.

Supplementary Specification

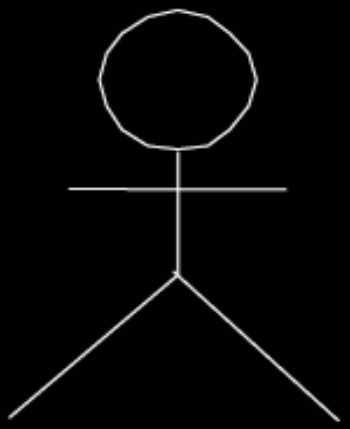
- Functionality
- Usability
- Reliability
- Performance
- Supportability
- Design constraints



Supplementary Specification

What Is an Actor?

- Actors represent roles a user of the system can play.
- They can represent a human, a machine, or another system.
- They can actively interchange information with the system.
- They can be a giver of information.
- They can be a passive recipient of information.
- Actors are not part of the system.
  - Actors are EXTERNAL.



Actor

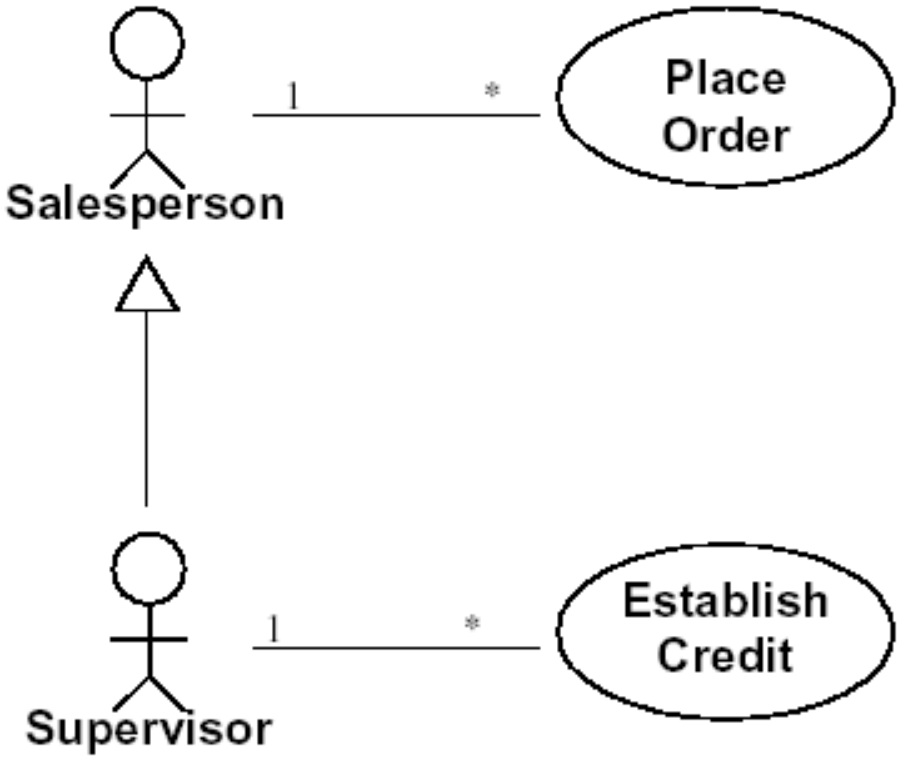
Actor就是边界！

确定行为者

- 行为者代表一种角色，而非一个人
- 系统外部的参与者
- 主动行为者（启动用例）/被动行为者（参与用例）
- 寻找行为者
  - 谁使用系统的主要功能？
  - 谁需要系统的支持以完成日常工作？
  - 系统需要与哪些其他系统交互？
  - 哪些人或哪些系统对该系统所产生的结果感兴趣？

行为者之间的关系

- 关联 association
- 泛化 generalization



What Is a Use Case?

- Defines a set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable result of value to a particular actor.
  - A use case models a dialogue between one or more actors and the system
  - A use case describes the actions the system takes to deliver something of value to the actor

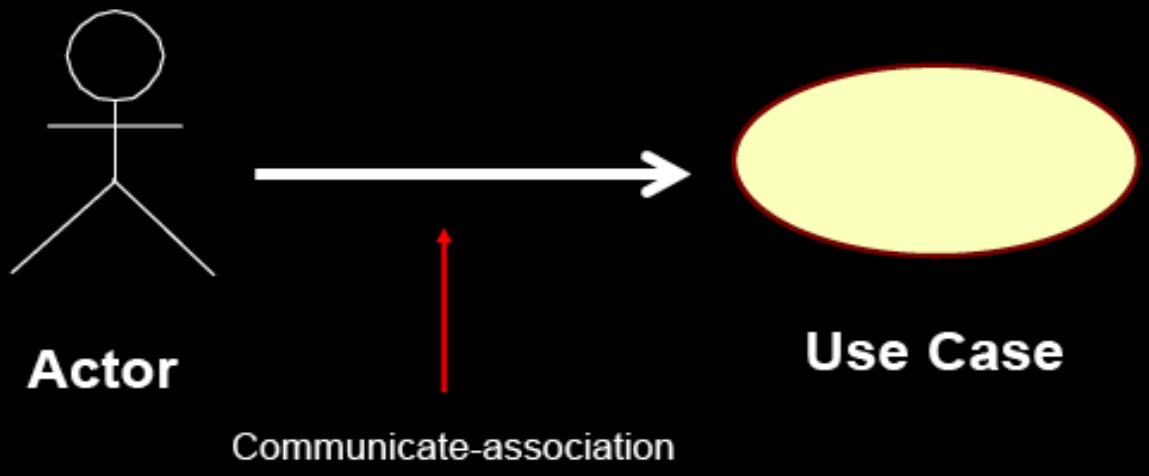
Use Case

确定用例

- 用例的特征
  - 被行为者启动
  - 向行为者提供可识别的处理结果
  - 完整的描述
- 寻找用例
  - 行为者需要系统提供哪些功能？行为者需要做什么？
- 用例描述
  - 着眼于系统的外部行为，忽略系统内部的实现
  - 描述内容：
    - 目的；启动的前提条件；交互的信息流；选择执行；结束条件

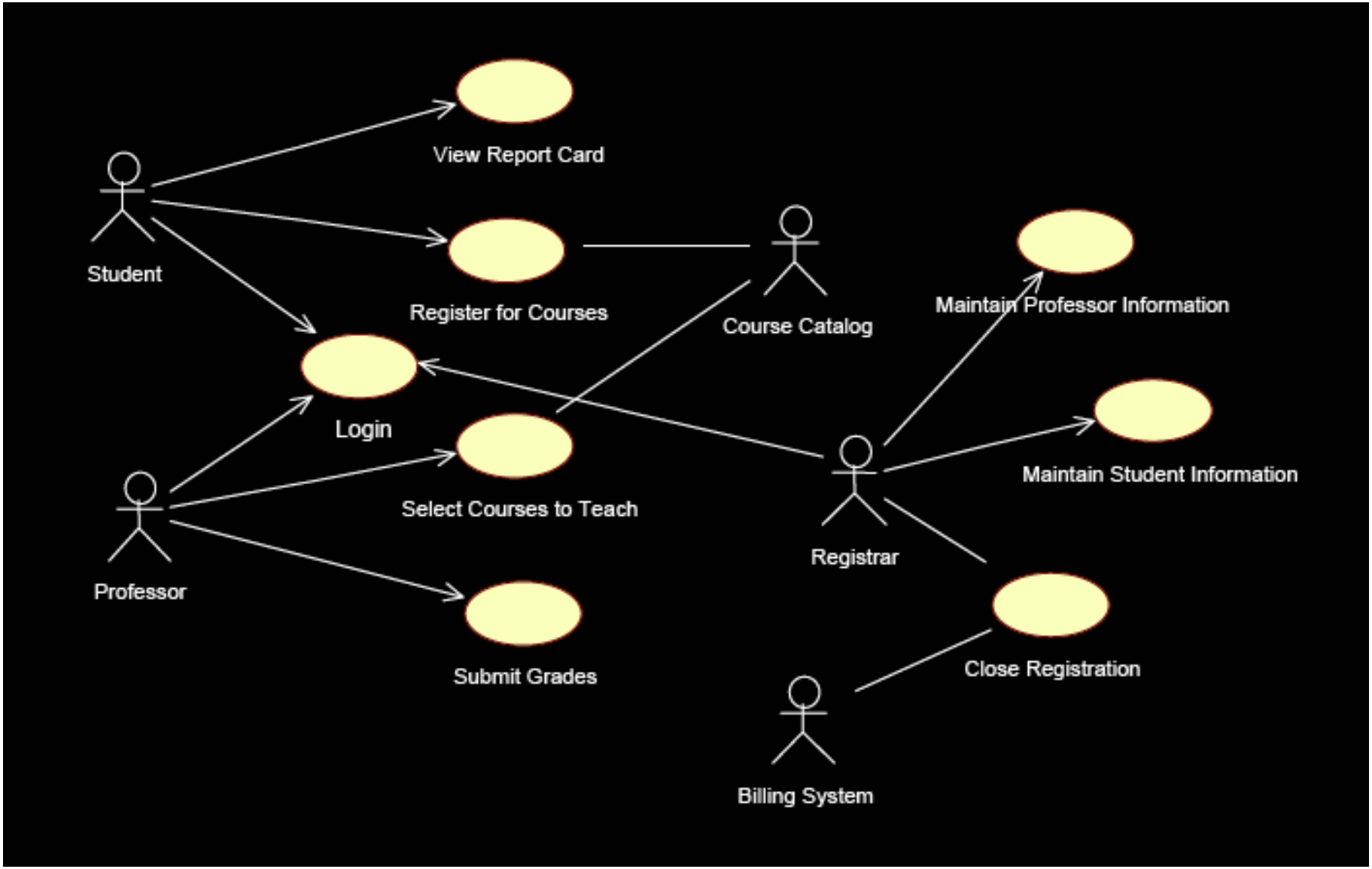
Use Cases and Actors

- A use case models a dialog between actors and the system.
- A use case is initiated by an actor to invoke a certain functionality in the system.



用例与行为者之间的关系

- 关联 association
  - 表明行为者与用例的参与者关系
  - 一个用例可能与系统的一个或多个参与者交互
    - 主动行为者：对系统服务的请求
    - 被动行为者：系统要求参与的行为
  - 用例和行为者之间的唯一关系

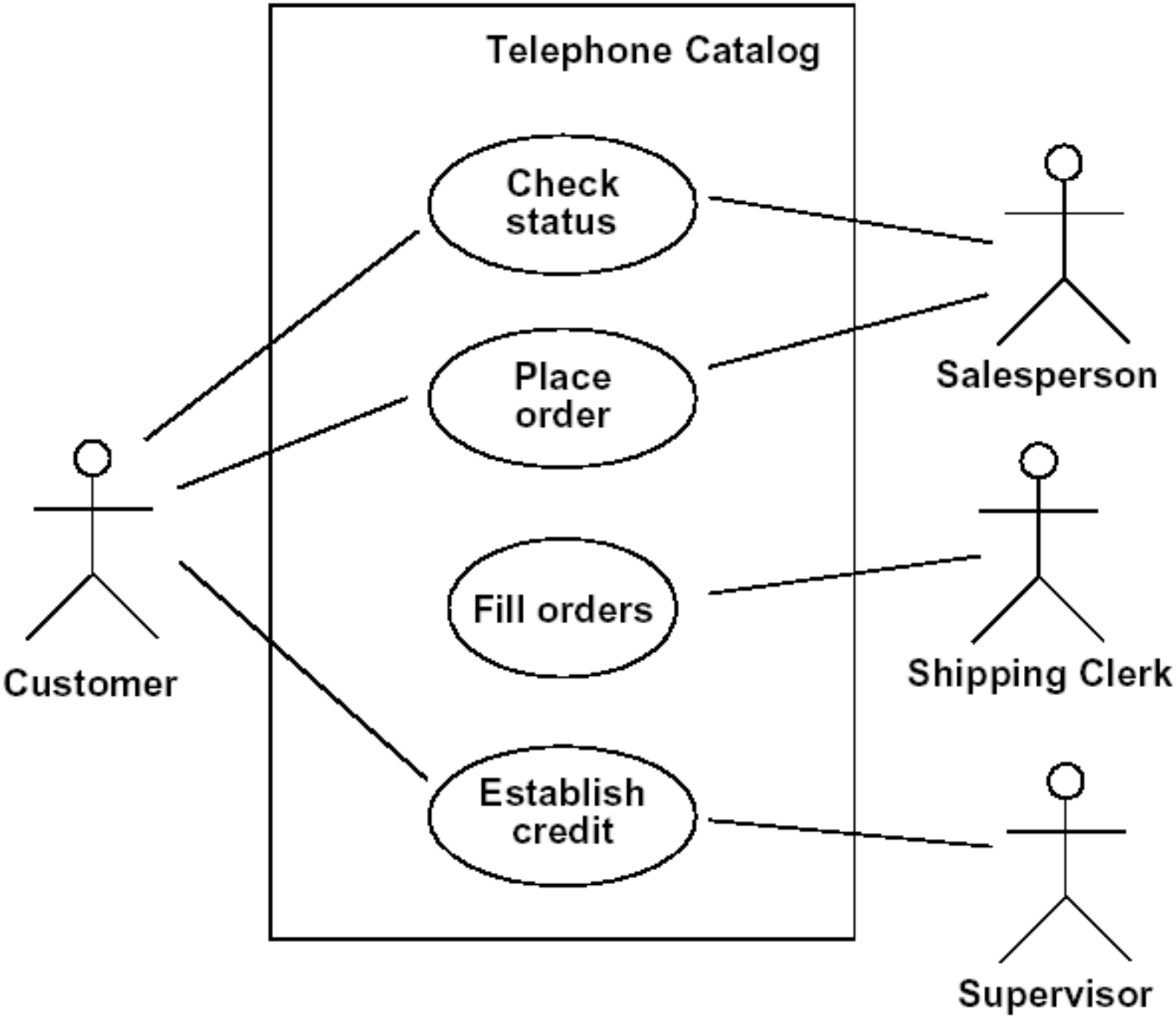


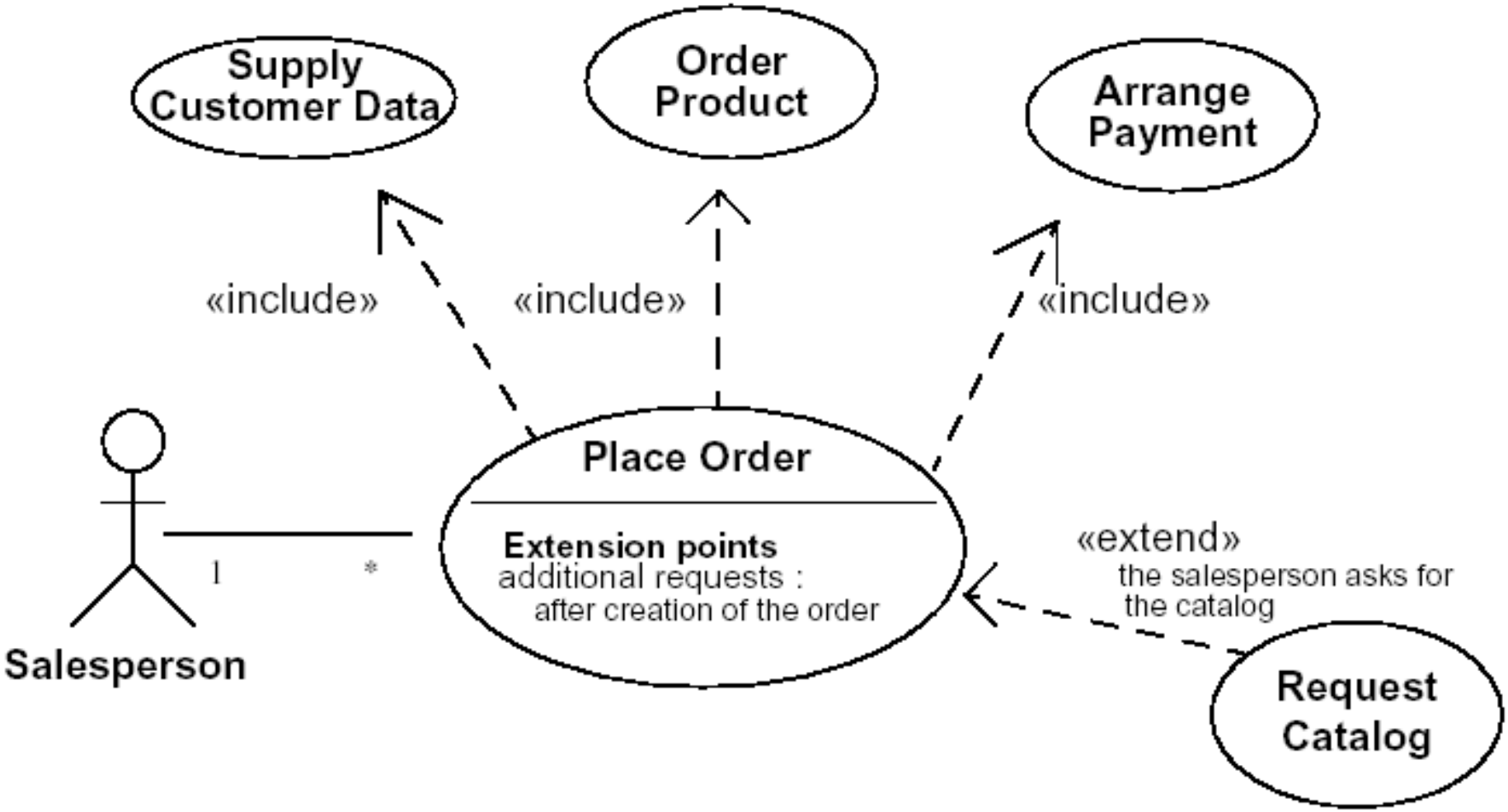
用例之间的关系

- 扩展 extend
  - 从用例A到用例B的扩展关系是指，用例B的实例是可以被用例A指定的行为扩充（服从与在扩展中指定的特定条件）。行为被插入到由B中的扩展点定义的位置。
  - “扩展点”：标识被扩展用例调用扩展用例的位置
    - 名字唯一
    - 采用文本描述

用例之间的关系

- 泛化 generalization
  - 子用例继承父用例的行为和含义
  - 子用例可以增加或覆盖父用例的行为
- 包含 include（use）
  - 用例A包含用例B说明，A的实例将包含B所声明的行为
  - “共性分析”：多个用例中共用的活动序列



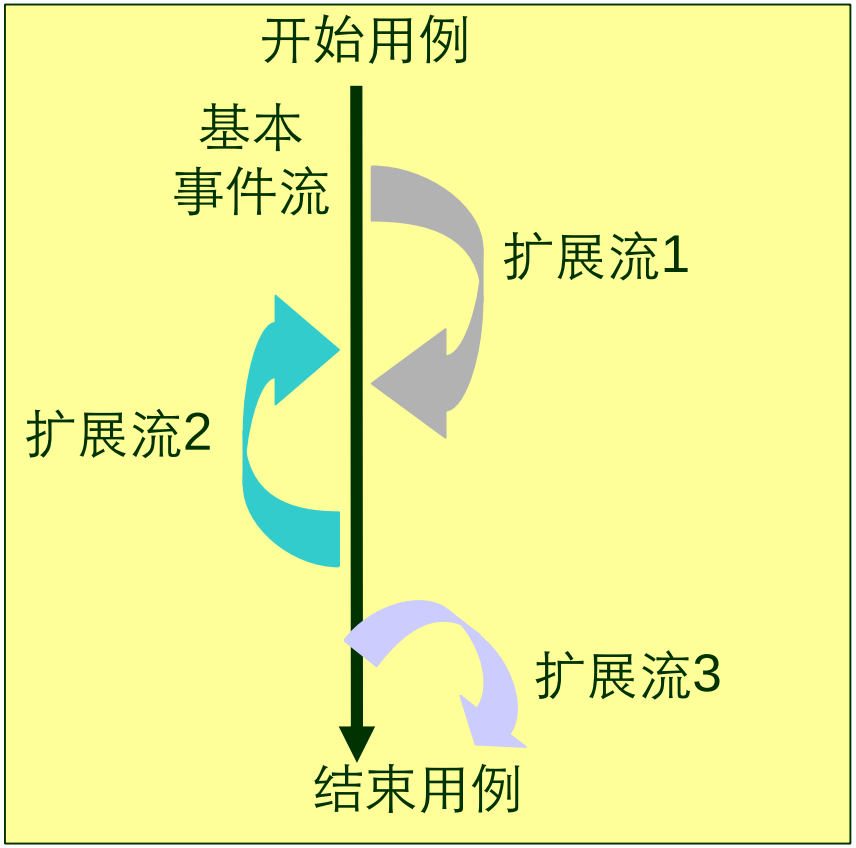


用例描述

- 简要描述：目的、作用
- 事件流：动作序列
  - 前置条件：指在用例启动时，参与者（Actor）与系统应置于什么状态，这个状态应该是系统能够检测到的、可观测的
  - 后置条件：用例结束时，系统应置于什么状态，这个状态也应该是系统能够检测得到的、可观测的
  - 基本事件流：基本事件流是对用例中常规、预期路径的描述，也被称为Happy day场景，这时大部分时间所遇到的场景；它将体现系统的核心价值
  - 扩展事件流：主要是对一些异常情况、选择分支进行描述

事件流

- 用例的实例，特定事件流的路径



场景1：基本流
场景2：基本流 扩展流1
场景3：基本流 扩展流1 扩展流3
场景4：基本流 扩展流2
场景5：基本流 扩展流2 扩展流3
场景6：基本流 扩展流3

活动图 Activity Diagram

- 目的
  - 用来描述完成一个操作所需活动，或一个用例实例的活动
- 组成
  - 采用状态图的符号表示，状态图的变形
    - 状态：动作状态
    - 状态迁移：可附加警戒条件、发送子句、动作表达式等
    - 判定 decision：有两个或两个以上携带警戒条件的输出迁移，当其中某个警戒条件为真时，该条件被触发
    - 泳道 swimlane：根据责任，将活动组织到不同的泳道中

What Is an Activity Diagram?

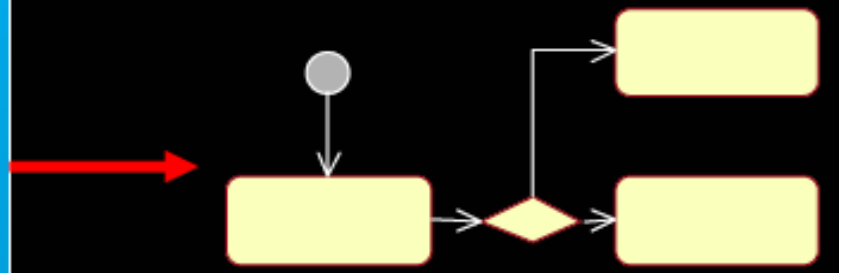
- An activity diagram in the use-case model can be used to capture the activities in a use case.
- It is essentially a flow chart, showing flow of control from activity to activity.

Flow of Events

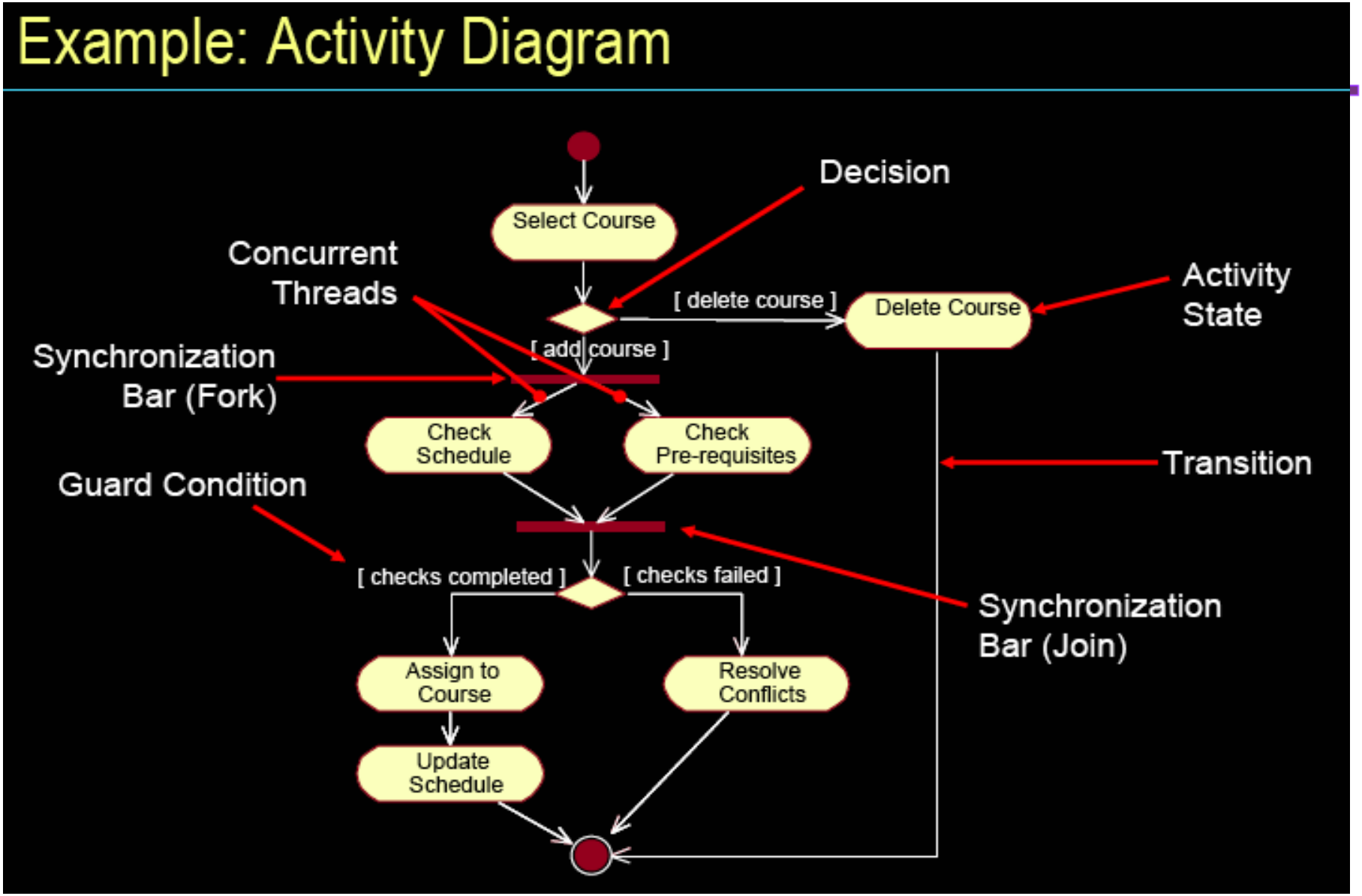
This use case starts when the Registrar requests that the system close registration.

1. The system checks to see if registration is in progress. If it is, then a message is displayed to the Registrar and the use case terminates. The Close Registration processing cannot be performed if registration is in progress.

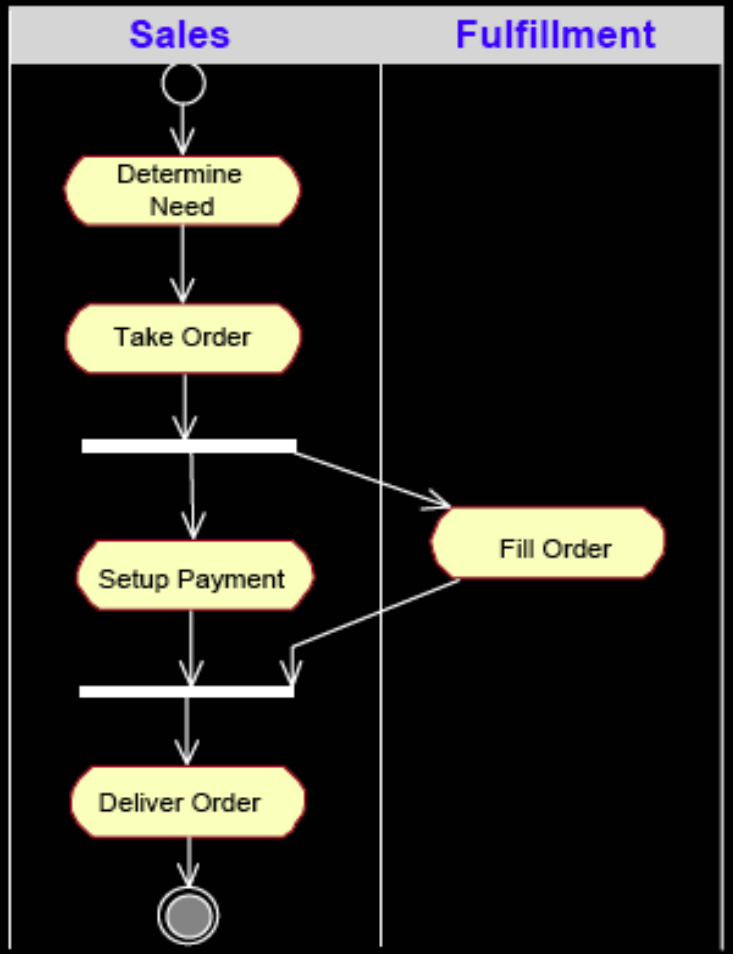
2. For each course offering, the system checks if a professor has signed up to teach the course offering and at least three students have registered. If so, the system commits the course offering for each schedule that contains it.



Example: Activity Diagram



Swimlanes



用例检查

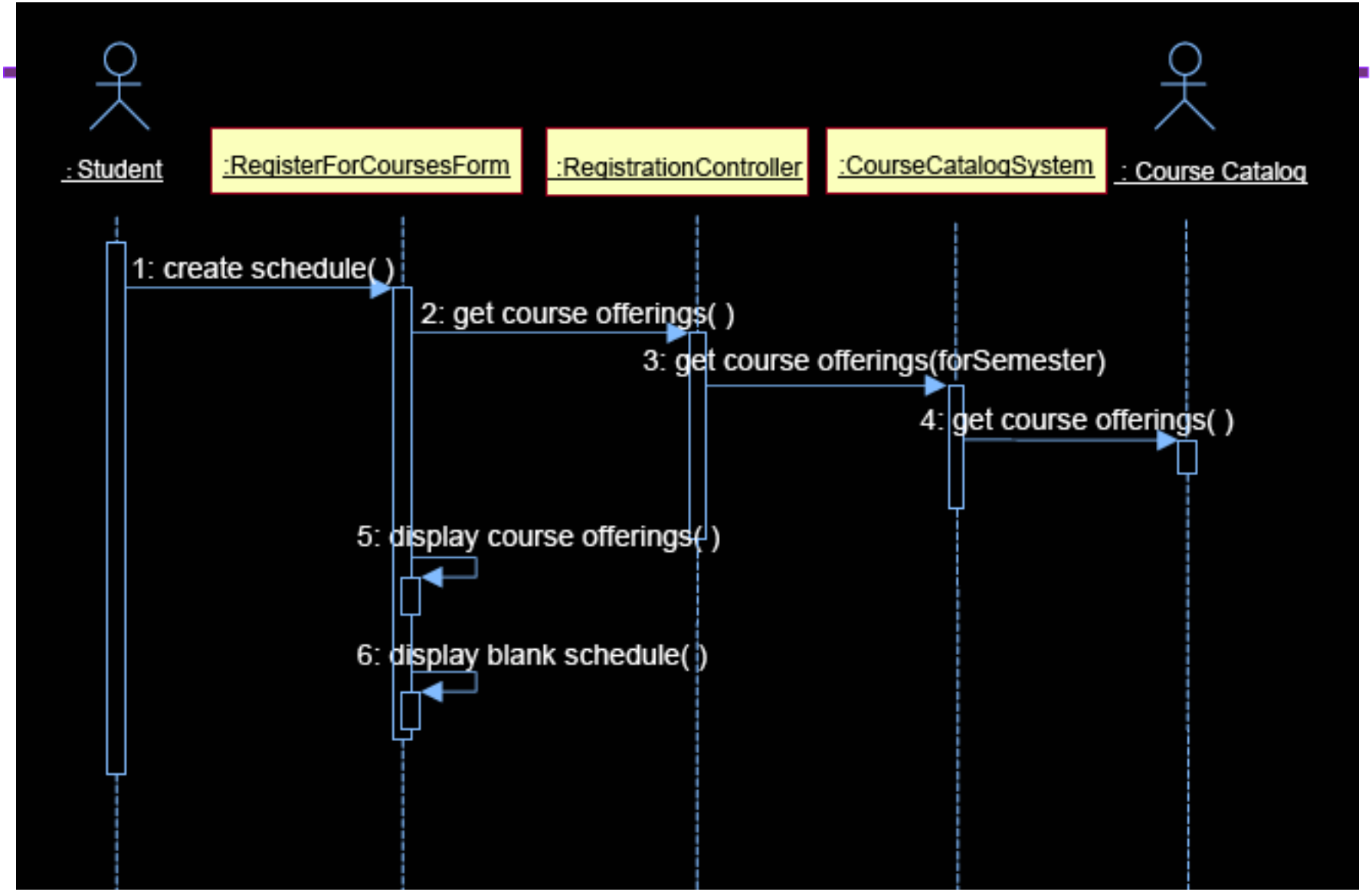
- 所有**干系人（stakeholder）**的利益都得到体现
- 分析用例
  - **正交**。不重，任何两个用例之间没有重复功能
  - **完备**。不落，所有用户期望的功能都得到描述
- 分析**事件流**
  - 平衡各用例的事件流步骤，10步之内
  - 如果某用例事件流步骤很少，重新分析，合并
  - 如果某用例事件流步骤很多，重新分析，扩展
  - 如果某几个 用例事件流步骤类似，重新分析、抽象

主要步骤 -- 2

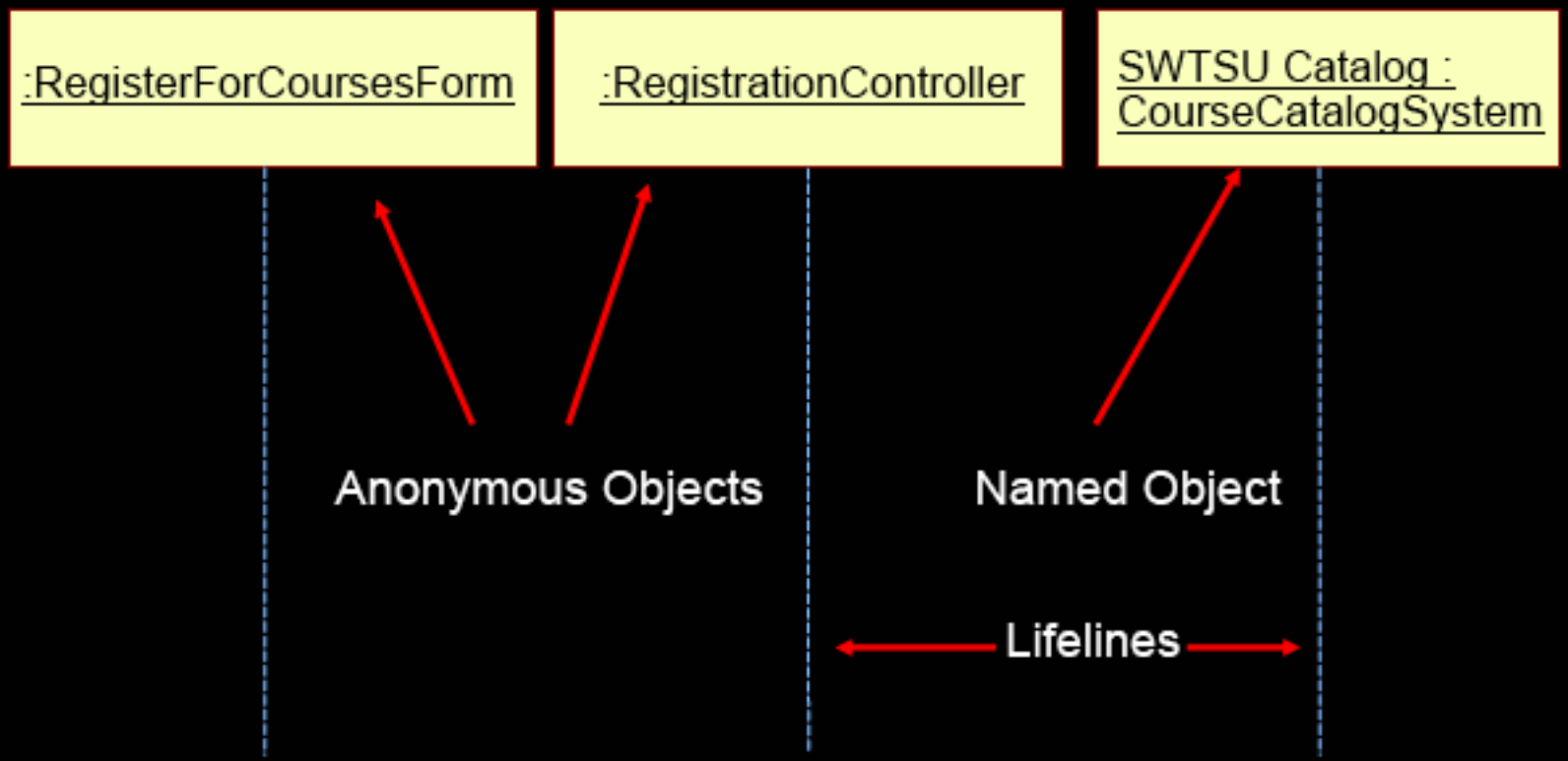
- 深入分析每个用例的内在行为
  - 定义系统的功能的具体操作步骤和参与者
  - 确定各参与者的责任与分工
  - 行为建模（Interactive Diagram）
    - 时序图 (Sequence Diagram)
    - 协作图 (Collaboration Diagram)
    - CRC (Class Responsibility Class) 进一步细化确定类

时序图Sequence Diagram

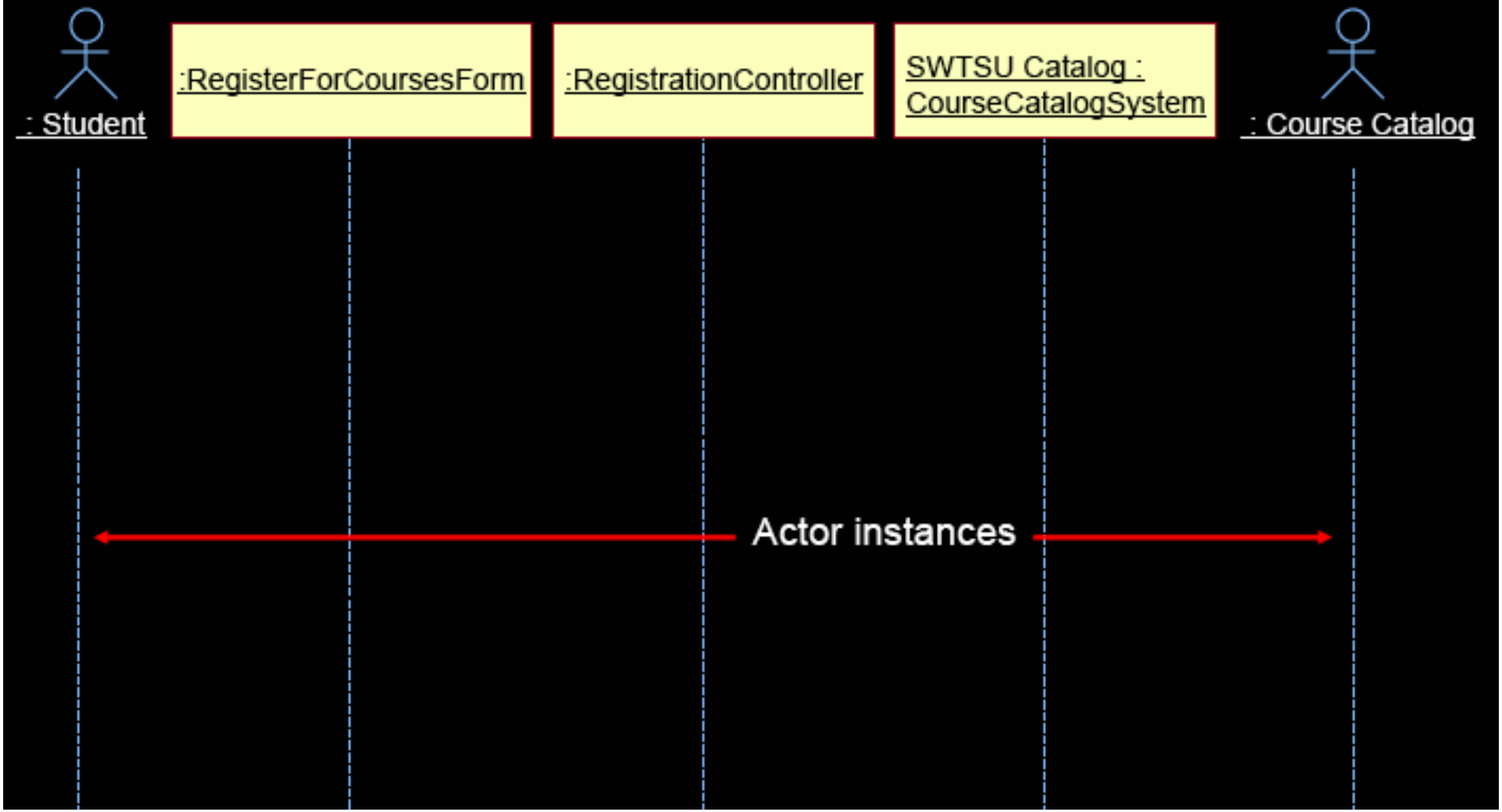
- 目的
  - 描述对象间的动态交互关系，体现对象间消息传递的时间顺序
- 组成
  - 垂直坐标轴：时间
  - 水平坐标轴：一组交互的对象
    - 对象框：对象名/类名
    - 生存线 lifeline：对象执行期间的时序
  - 消息：信号，操作调用等
  - 激活：对象收到消息后，开始执行活动



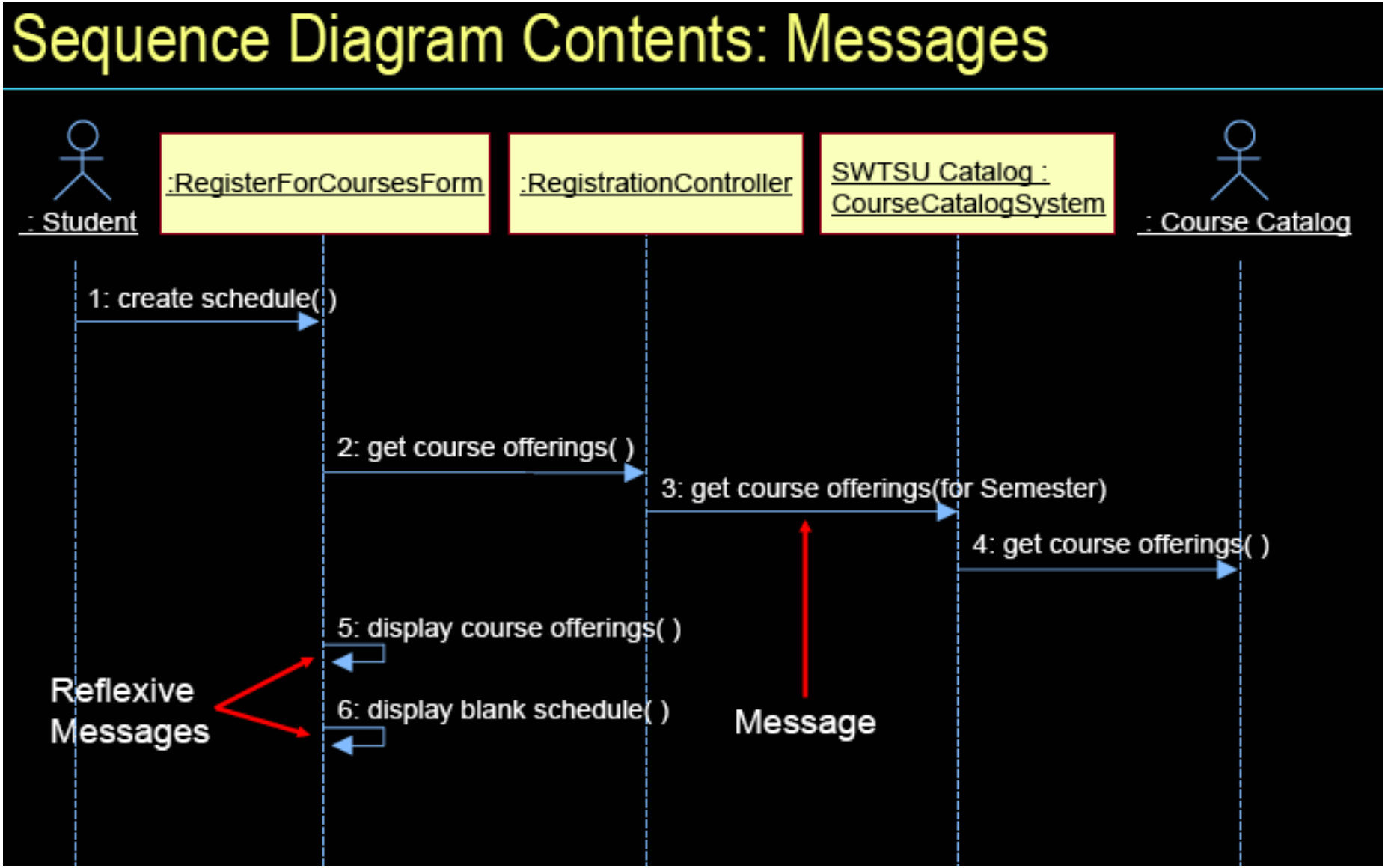
Sequence Diagram Contents: Objects



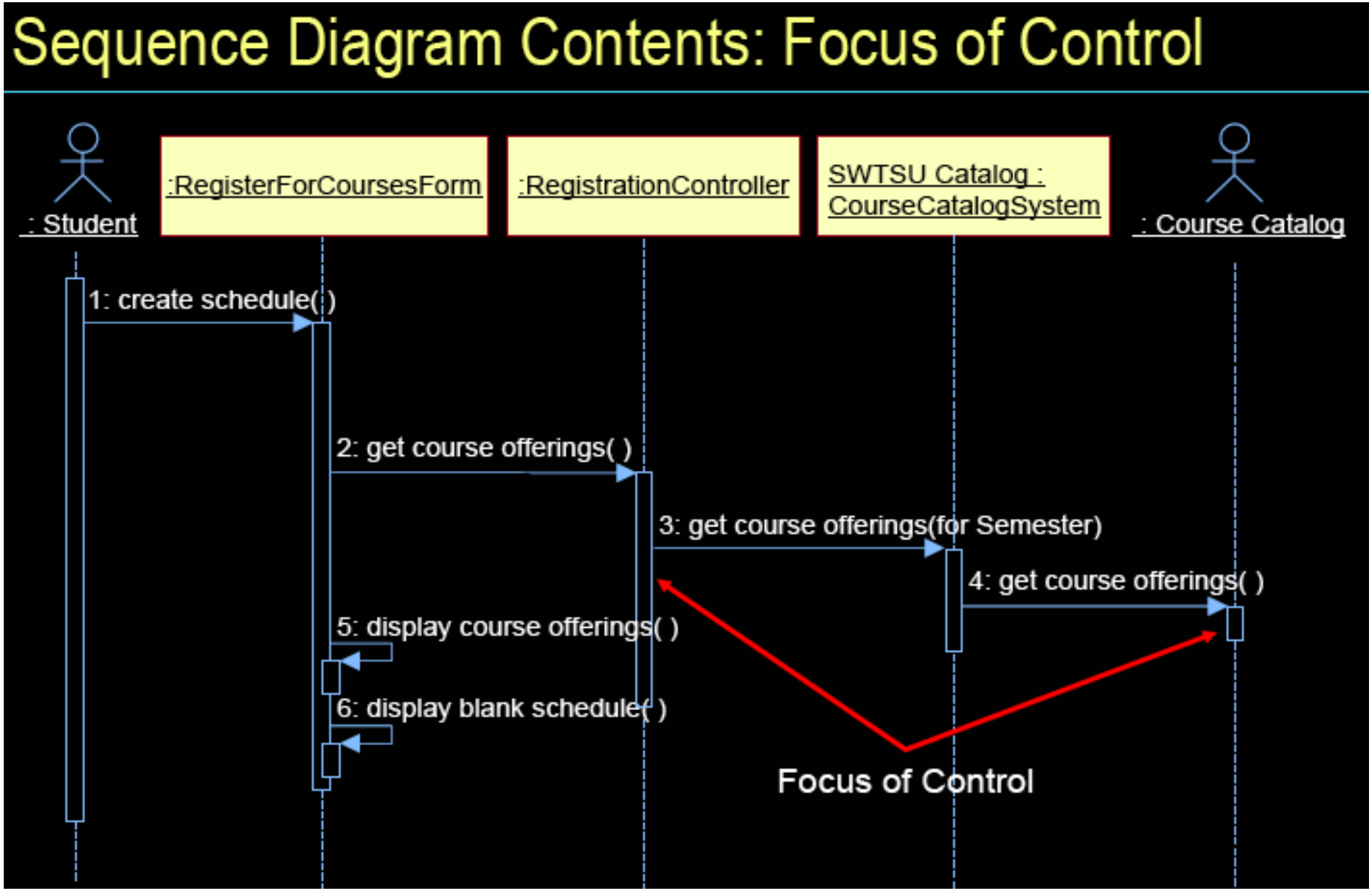
Sequence Diagram Contents: Actor



Sequence Diagram Contents: Messages

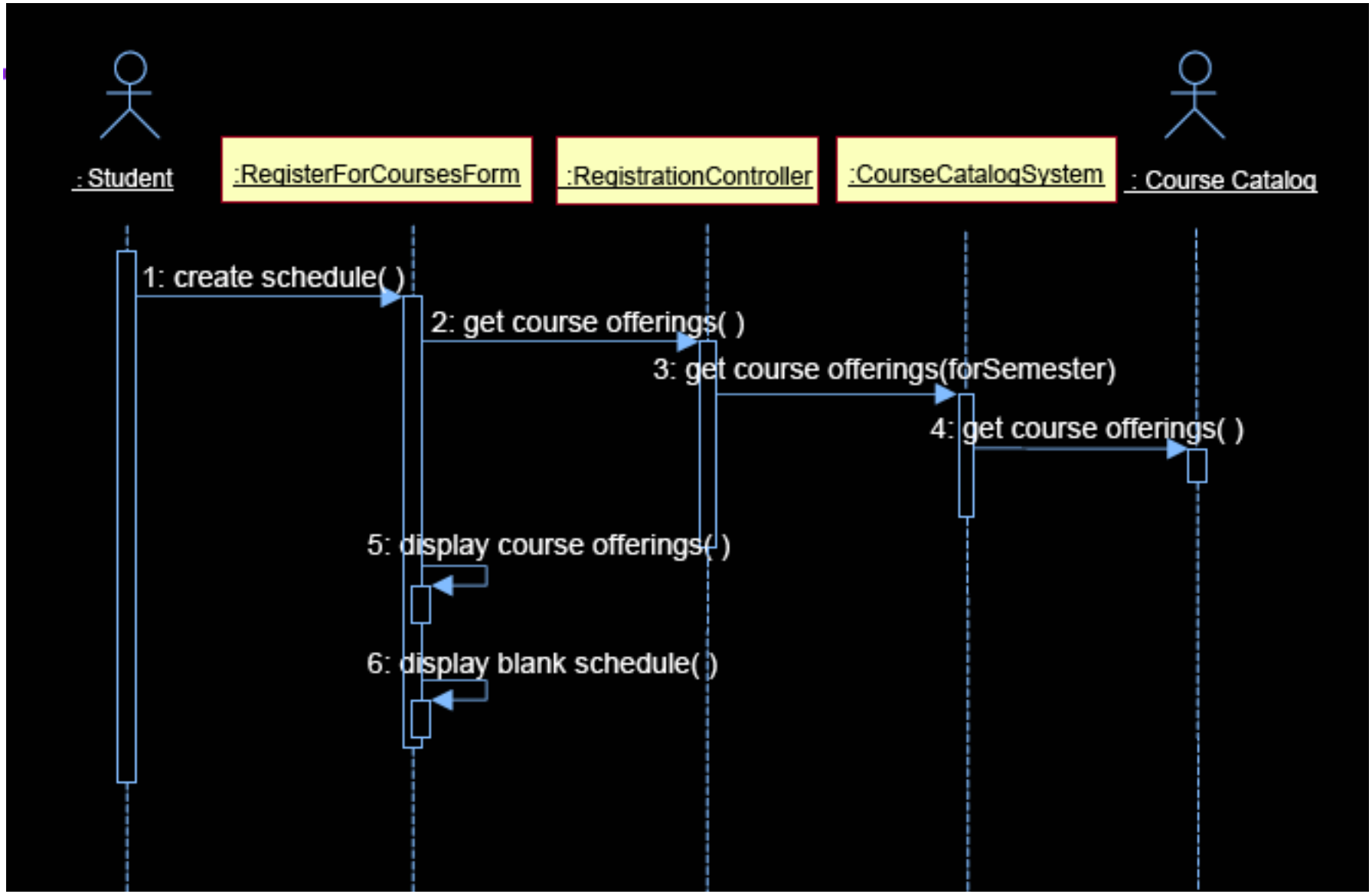


Sequence Diagram Contents: Focus of Control

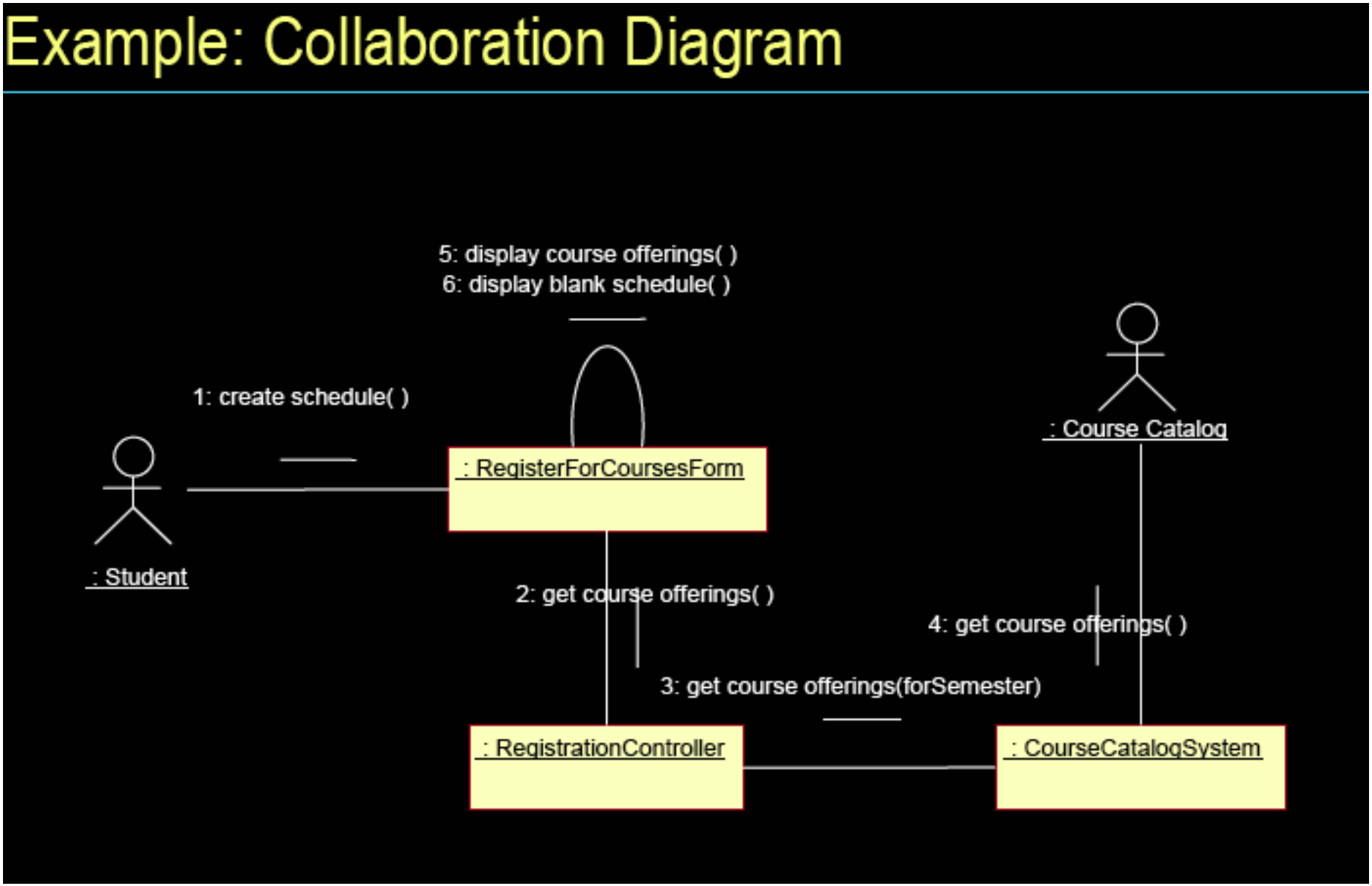


协作图 Collaboration Diagram

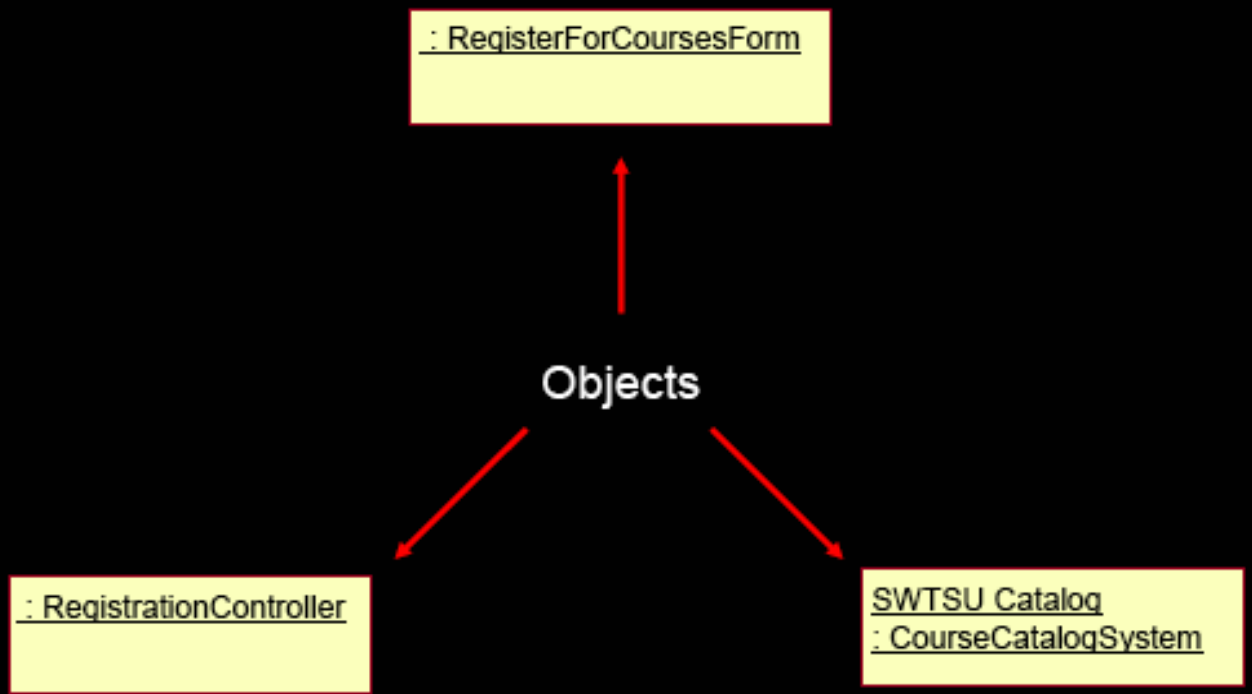
- 目的
  - 协作对象间的交互和链接
- 组成
  - 链接
    - 两个对象之间的连接，说明对象间的关系如聚集、关联等
    - 链上可标明：角色名及约束（需与类图一致）
  - 消息流
    - 在链接线上，用带消息串的消息描述对象间的交互
    - 消息串说明发送的顺序、条件、重复、回送值等



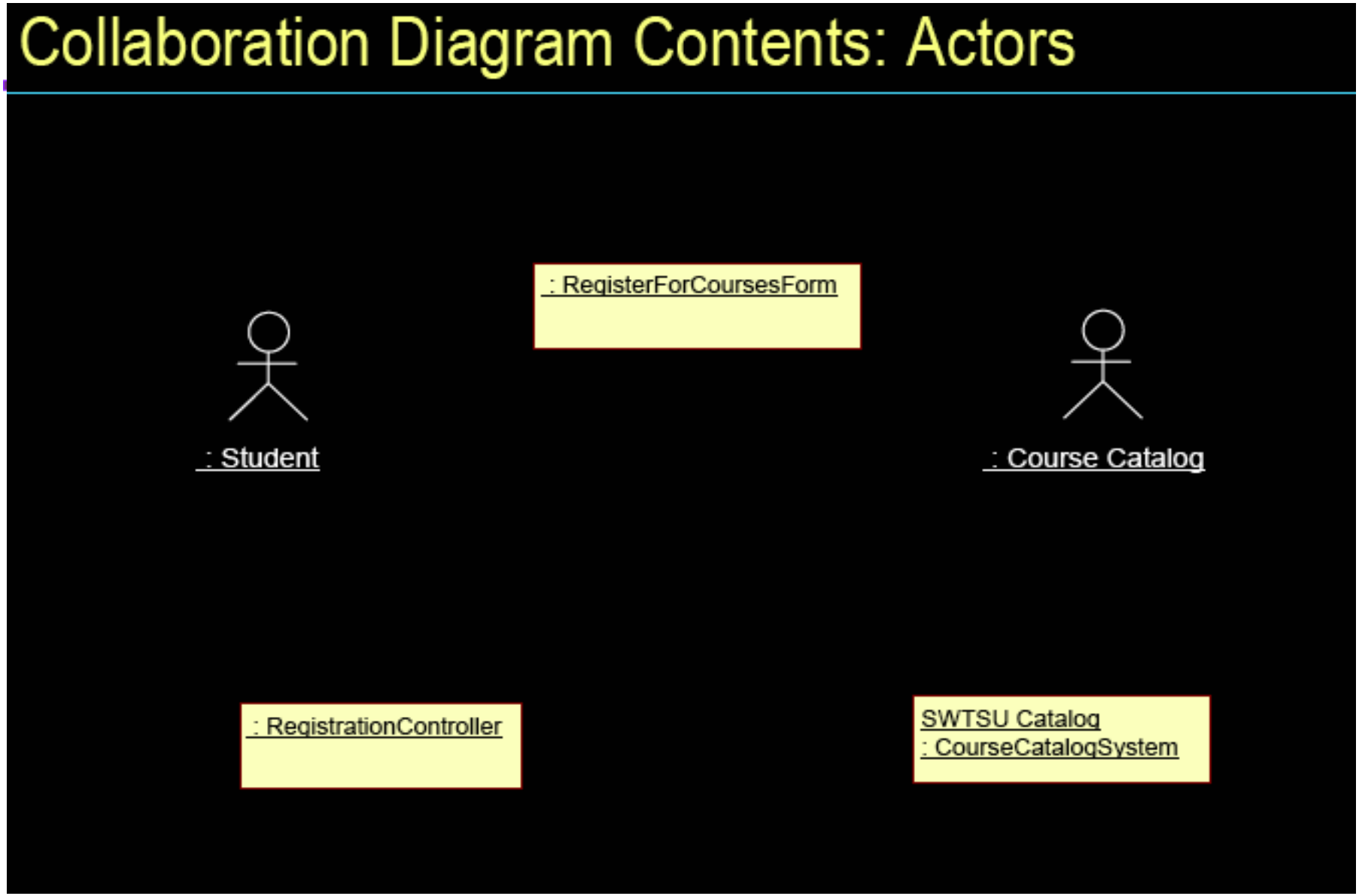
Example: Collaboration Diagram



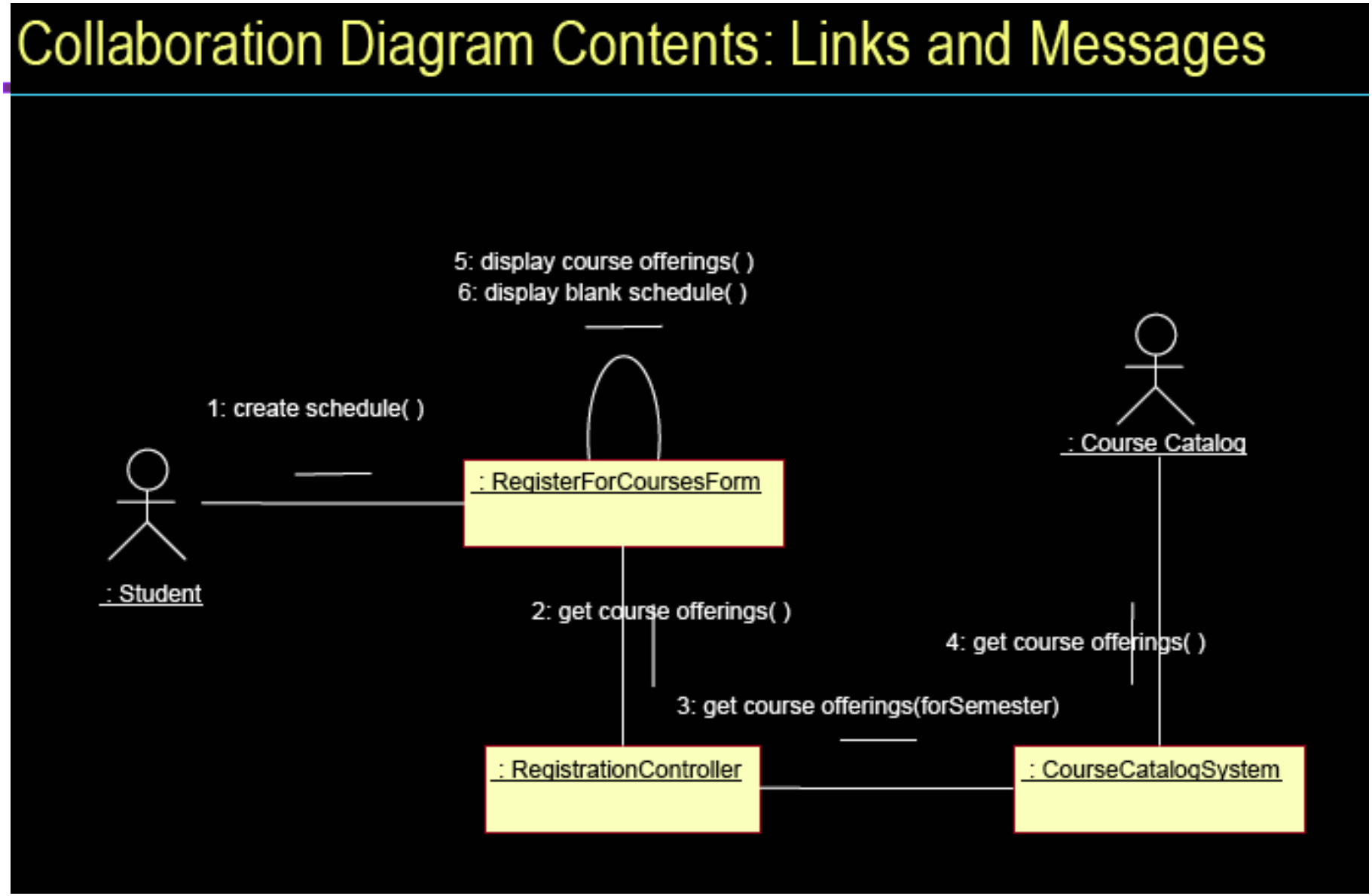
Collaboration Diagrams Contents: Objects



Collaboration Diagram Contents: Actors



Collaboration Diagram Contents: Links and Messages



Sequence and Collaboration Diagram Similarities

- Semantically equivalent
  - Can convert one diagram to the other without losing any information
- Model the dynamic aspects of a system
- Model a use-case scenario

Sequence and Collaboration Diagram Differences

- Collaboration diagrams
  - Show relationships in addition to interactions
  - Better for visualizing patterns of collaboration
  - Better for visualizing all of the effects on a given object
  - Easier to use for brainstorming sessions
- Sequence diagrams
  - Show the explicit sequence of messages
  - Show focus of control
  - Better for visualizing overall flow
  - Better for real-time specifications and for complex scenarios

小结

- 面向对象的基本概念
- UML 是基于面向对象技术的标准建模语言
  - Specifying, Visualizing, Constructing, Documenting
- Use Case Diagram：通过行为者和功能用例描述系统的主要外部表现，刻画其边界
- Interactive Diagram：时序图及协作图描述对象间的交互关系，时序图侧重于描述时间约束；协作图侧重于表现交互结构
- 识别类及其属性、操作、协作关系

需求分析总结

- 需求分析
  - 概念
  - 目标
- 需求分析方法
  - 一般流程
    - 需求获取
    - 需求建模
    - 需求验证
    - 需求评审
    - 需求管理
  - 结构化分析方法
  - 非结构化方法
    - **UML**（面向对象、形式化）

案例分析

- 研究生小张组了一个小开发团队，接项目赚钱。最近，他接了一个项目，时间紧，客户要求很高，但报酬丰厚。小张学过软件工程，知道需求的重要性，决定这一次好好从需求开始落实，提高软件质量，不容有失。
- 于是，团队很快就把需求写好了，也定稿了，小张觉得差不多了，后边任务很紧张，无论瀑布还是迭代，都得往下走了，于是赶紧开始了开发。投入4个小伙伴，翘了无数次课、熬了无数次夜，历时三个月，终于开发完了。拿给客户看的时候，客户对于界面、功能、使用流程无一个满意，提出终止合作。
- 小张傻眼了。。。

案例分析

- 研究生小张的团队，经历过上次失败后，大家并不气馁，好歹也没有花钱，只是浪费了时间，而且还涨了经验值。这次，小张又接了一个项目，朋友介绍的一个税务系统。
- 吸收上次的教训，这次需求经过反复和客户的主管沟通，并且最后获得了他的确认，签了合同，拿到了预付款10%，开工！小张的团队很兴奋，软件工程终于形成了生产力！于是，团队按照需求，一步步地开发、测试，版本迭代，历经无数次翘课、熬夜，三个月后，终于做完了！
- 小张忐忑地拿去给客户，对方主管一看，对于增值税发票流程有疑问，其中一部分功能并不是自己想的那样，需要大修改；拿给业务员测试，业务员说问题很多，给出结论：“根本不能用！”，要求退钱！
- 小张又傻眼了。。。

谢谢大家！