4.12 面向对象系统分析

4.12.5 非功能需求概览

- Nonfunctional requirements: User visible aspects of the system not directly related to functional behavior.
  - Usability（可用性）
  - Reliability（可靠性）
  - Performance（性能）
  - Supportability（可支持性）
  - Implementation（实现条件）
  - Interface（接口）
  - Operations（运转）

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Usability（可用性）
  - User interface and human factors
    - *What type of user will be using the system?*
    - *Will more than one type of user be using the system?*
    - *What sort of training will be required for each type of user?*
    - *Is it particularly important that the system be easy to learn?*
    - *Is it particularly important that users be protected from making errors?*
    - *What sort of input/output devices for the human interface are available, and what are their characteristics?*

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Usability（可用性）
  - Documentation
    - *What kind of documentation is required?*
    - *What audience is to be addressed by each document?*
    - Example: 美军 **MIL-STD-498** 中的文档种类

美军 MIL-STD-498 中的文档种类

- **系统/子系统规格说明（SSS）**：系统和子系统的需求及满足每一需求采用的方法
- **系统/子系统设计说明（SSDD）**：系统与子系统的设计与体系结构设计
- **软件开发计划（SDP）**：描述开发者的软件开发过程
- **软件需求规格说明（SRS）**： CSCI（计算机软件配置项）的需求及满足需求所采用的方法
- **接口需求规格说明（IRS）**：系统、子系统、HWCI（硬件配置项）、CSCI、手工操作及其它部件的接口要求
- **软件设计说明（SDD）**：CSCI的设计，包括设计决策、体系结构设计、详细设计
- **软件测试计划（STP）**：CSCI与软件系统的合格化测试计划
- **软件安装计划（SIP）**：用户现场安装软件的计划，包括准备、用户培训及现有系统的转换

美军 MIL-STD-498 中的文档种类

- **软件移交计划（STrP）**：给出支持交付软件所需的硬件、软件及其资源
- **操作原理说明（OCD）**：从用户要求与现有系统（或过程）的关系及使用的方法等不同角度来描述系统
- **软件用户手册（SUM）**
- **接口设计说明（IDD）**：给出系统、子系统、HWCC、CSCI、手工操作及其它部件的接口特征
- **数据库设计说明（DBDD）**：描述数据库设计，用来作为实现数据库及相关软件单元的基础
- **软件测试说明（STD）**：描述测试准备、测试用例、测试过程，用于执行CSCI或软件系统或子系统的合格性测试
- **软件测试报告（STR）**：关于对CSCI、软件系统或子系统或其它软件项进行合格性测试的记录

美军 MIL-STD-498 中的文档种类

- **计算机操作手册（COM）**：为操作计算机及其外设提供必要信息
- **计算机程序设计手册（CPM）**：为程序员理解如何对计算机编程提供所需的信息
- **固件支持手册（FSM）**：为系统的固件设备的编程与再编程提供所需的信息
- **软件版本说明（SVD）**：识别与描述由一个或多个CSCI组成的软件版本
- **软件输入/输出手册（SIOM）**：对于批式或交互式软件系统，用户如何存取、提交输入，如何解释其输出
- **软件中心操作手册（SCOM）**：为计算机或其它集中式（或网络）软件安装中工作人员提供安装、操作软件系统的信息
- **软件产品规格说明（SPS）**：包含可执行软件、源文件及文件支持信息或注明资料来源

上述文档可根据任务的性质、大小及其使用来进行合理的取舍与合并

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Reliability（可靠性）
  - Error handling and extreme conditions
    - *How should the system respond to input errors?*
    - *How should the system respond to extreme conditions?*
  - Security issues
    - *Must access to any data or the system itself be controlled?*
    - *Is physical security an issue?*

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Reliability（可靠性）
  - Quality issues
    - *What are the requirements for reliability?*
    - *Must the system trap faults?*
    - *Is there a maximum acceptable time for restarting the system after a failure?*
    - *What is the acceptable system downtime per 24-hour period?*
    - *Is it important that the system be portable (able to move to different hardware or operating system environments)?*
    - *What sorts of modifications are expected?*

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Performance（性能）
  - System performance
    - *Are there any speed, throughput, or response time constraints on the system?*
    - *Are there size or capacity constraints on the data to be processed by the system?*

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Supportability（可支持性）
  - System modifications
    - *What parts of the system are likely candidates for later modification?*

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Implementation（实现条件）
  - Hardware considerations
    - *What hardware is the proposed system to be used on?*
    - *What are the characteristics of the target hardware, including memory size and auxiliary storage space?*
  - Physical environment
    - *Where will the target equipment operate?*
    - *Will the target equipment be in one or several locations?*
    - *Will the environmental conditions in any way be out of the ordinary (for example, unusual temperatures, vibrations, magnetic fields, ...)?*

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Interface（接口）
  - Input and output considerations
    - *Is input coming from systems outside the proposed system?*
    - *Is output going to systems outside the proposed system?*
    - *Are there restrictions on the format or medium that must be used for input or output?*

4.12 面向对象系统分析

4.12.5 非功能需求概览

- Operations（运转）
  - Resources and management issues
    - *How often will the system be backed up?*
    - *Who will be responsible for the back up?*
    - *Who is responsible for system installation?*
    - *Who will be responsible for system maintenance?*

4.13 面向对象系统设计

4.13.1 系统设计的必要性

- 系统分析是对目标信息系统的问题空间（问题域）的理解、分析、准确的表示、客观的反映，不应当涉及解空间（即目标信息系统可能的那些解）。
- 程序设计是具体地实现一个特定的解的每个部件，这时已经不可能考虑到整个解的结构是否可行与合理。
- 系统设计是根据系统分析的结果，构建对应的解空间，并从中得出可行与合理的解的结构，其结果能够指导程序设计，并为测试、维护、产品的系列化、设计与程序设计的重用等，提供相应的基础。

4.13 面向对象系统设计

4.13.2 面向对象系统设计的主要活动

- 6 项主要活动:
  - 定义具体的用例（*real use cases*）
  - 定义报表和人机界面
  - 定义/改进系统的体系结构
  - 对（*to refine*）交互图进行精制
  - 定义设计类图（*design class diagrams*）
  - 定义数据库模式

4.13 面向对象系统设计

4.13.2 面向对象系统设计的主要活动

- 课堂上将讨论其中的 3 项:
  - 定义具体的用例（*real use cases*）
  - 定义报表和人机界面
  - 定义/改进系统的体系结构
  - 对（*to refine*）交互图进行精制
  - 定义设计类图（*design class diagrams*）
  - 定义数据库模式

4.13 面向对象系统设计

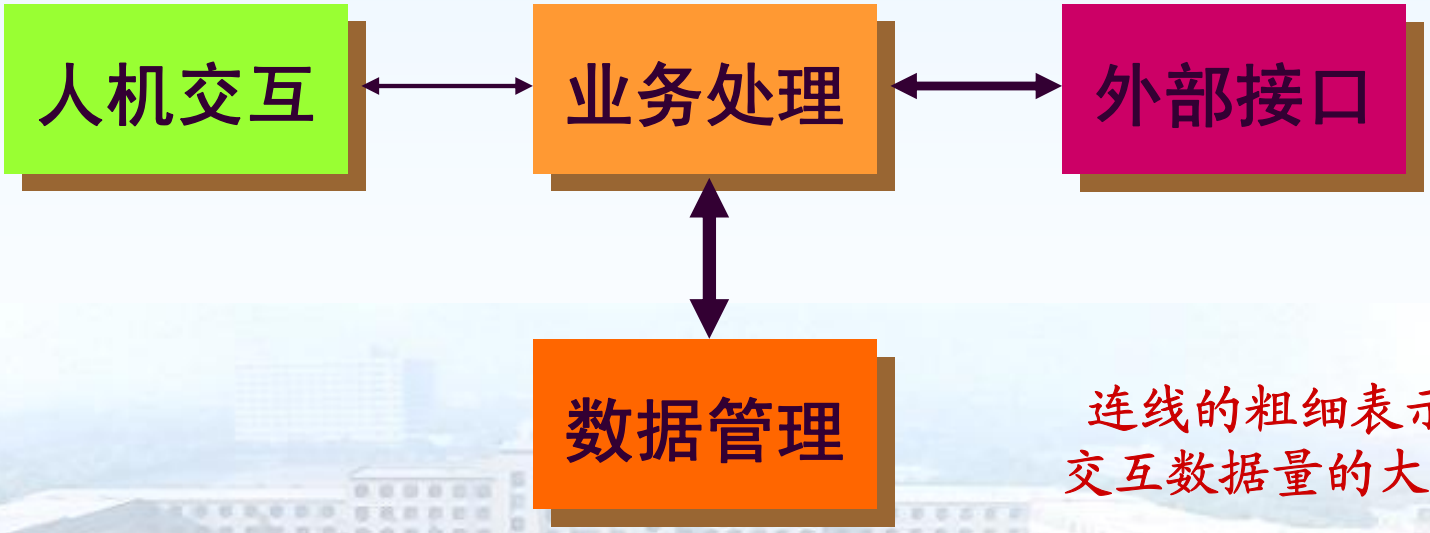
4.13.3 必须重视软件体系结构

- **Why?**
- 历史证明:
  - 影响大型信息系统生存能力的关键因素是应用软件。
  - 大型软件系统的成败，很大程度上取决于系统高层单元的组织形式与合作方式的设计。
  - 应用需求与领域特征都要求软件体系结构提供强有力的支持。

4.13 面向对象系统设计

4.13.3 必须重视软件体系结构

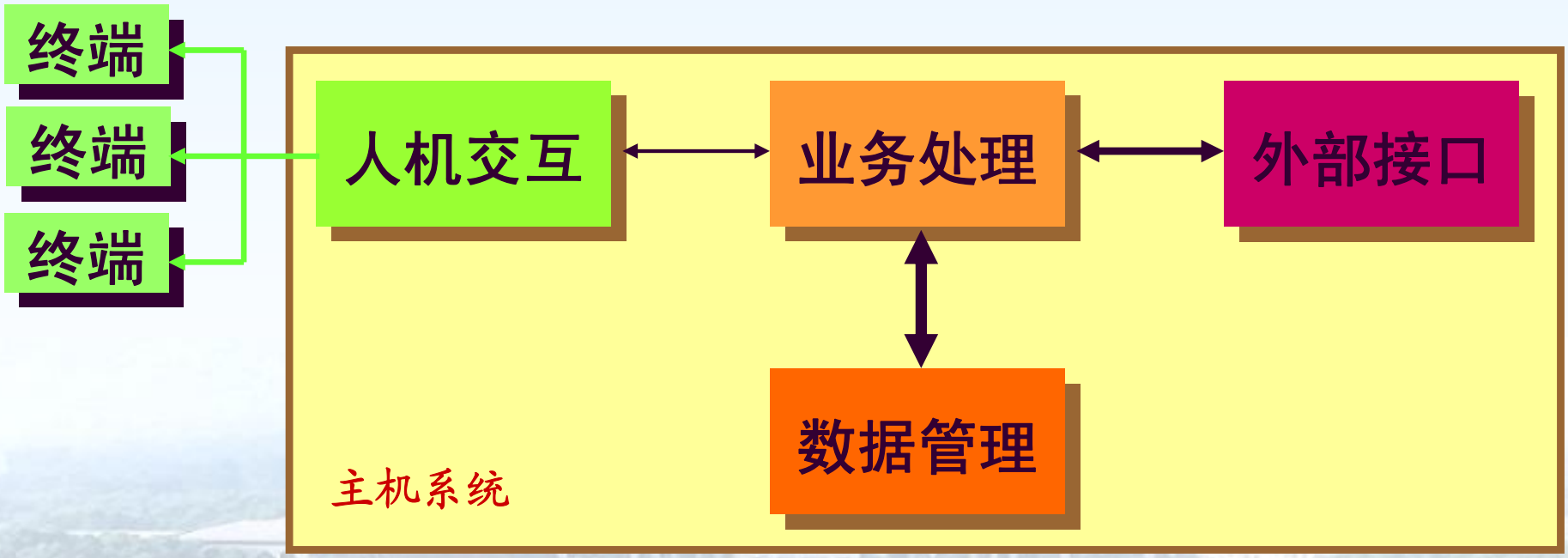
- 例：结构的能量－信息系统的 T 型结构要素



4.13 面向对象系统设计

4.13.3 必须重视软件体系结构

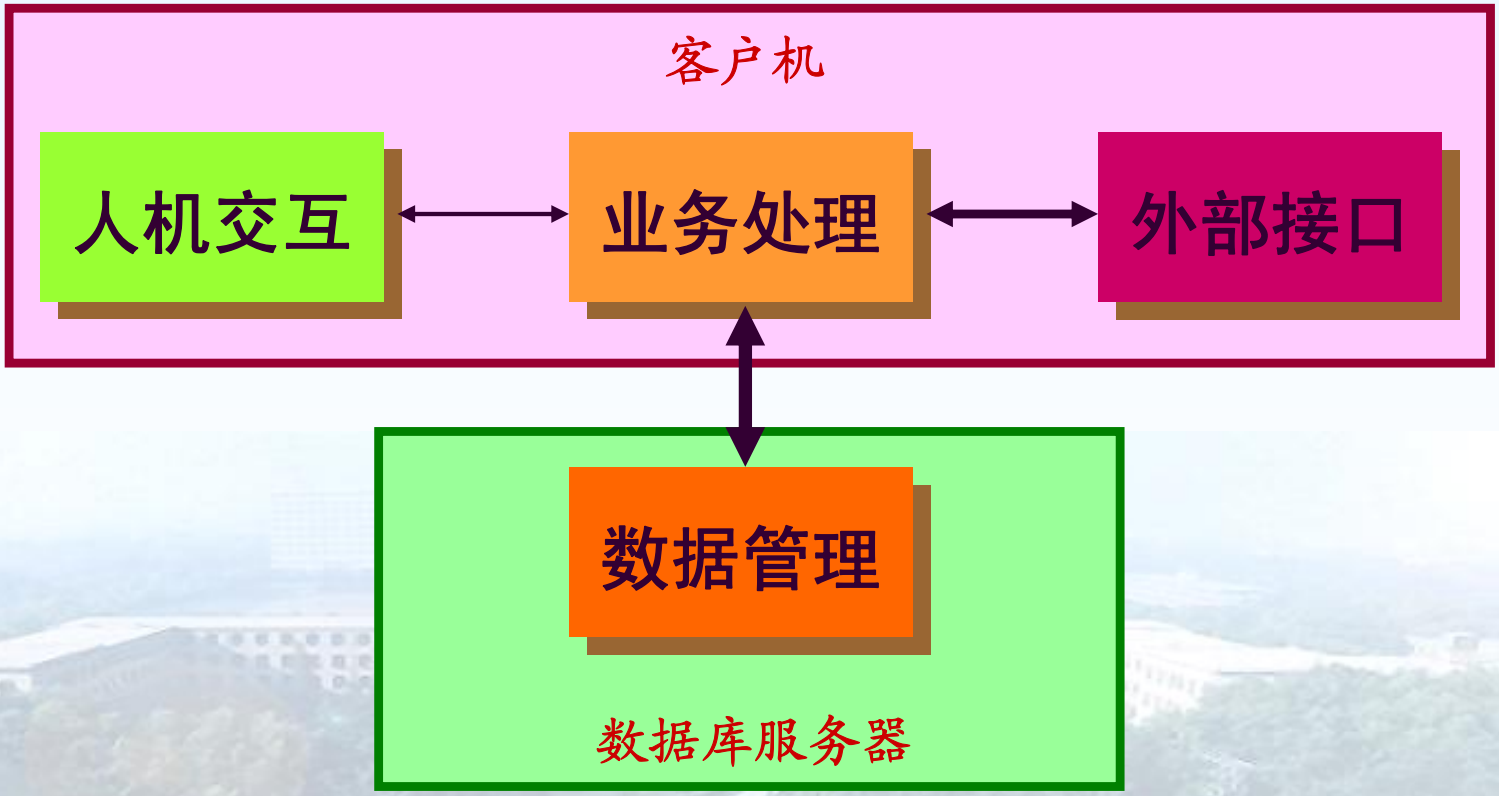
- 例：结构的能量－主机系统的体系结构要素



4.13 面向对象系统设计

4.13.3 必须重视软件体系结构

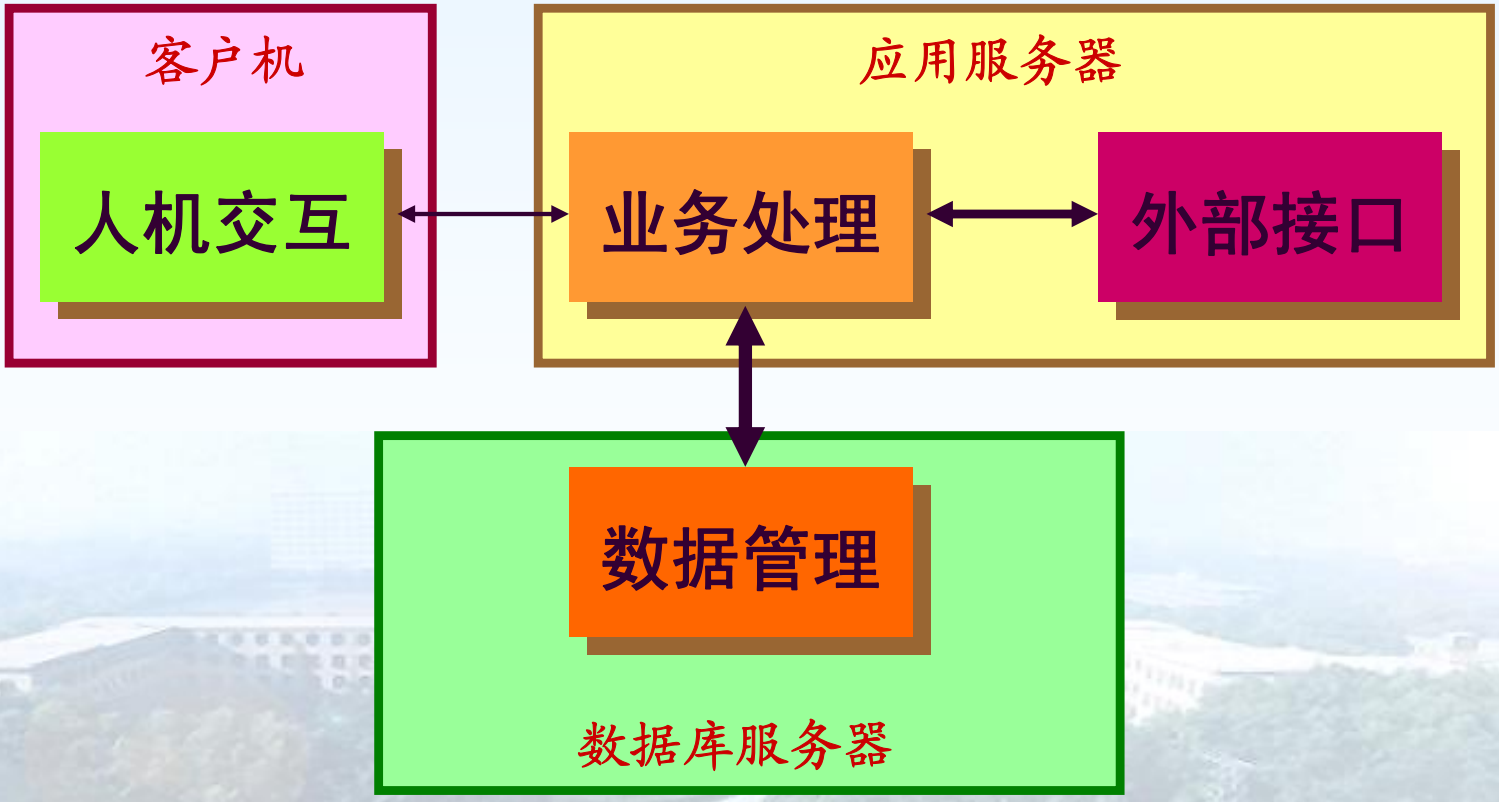
- 例：结构的能量－二层 C/S 节系统的体系结构要素



4.13 面向对象系统设计

4.13.3 必须重视软件体系结构

- 例：结构的能量－三层 C/S 节系统的体系结构要素



4.13 面向对象系统设计

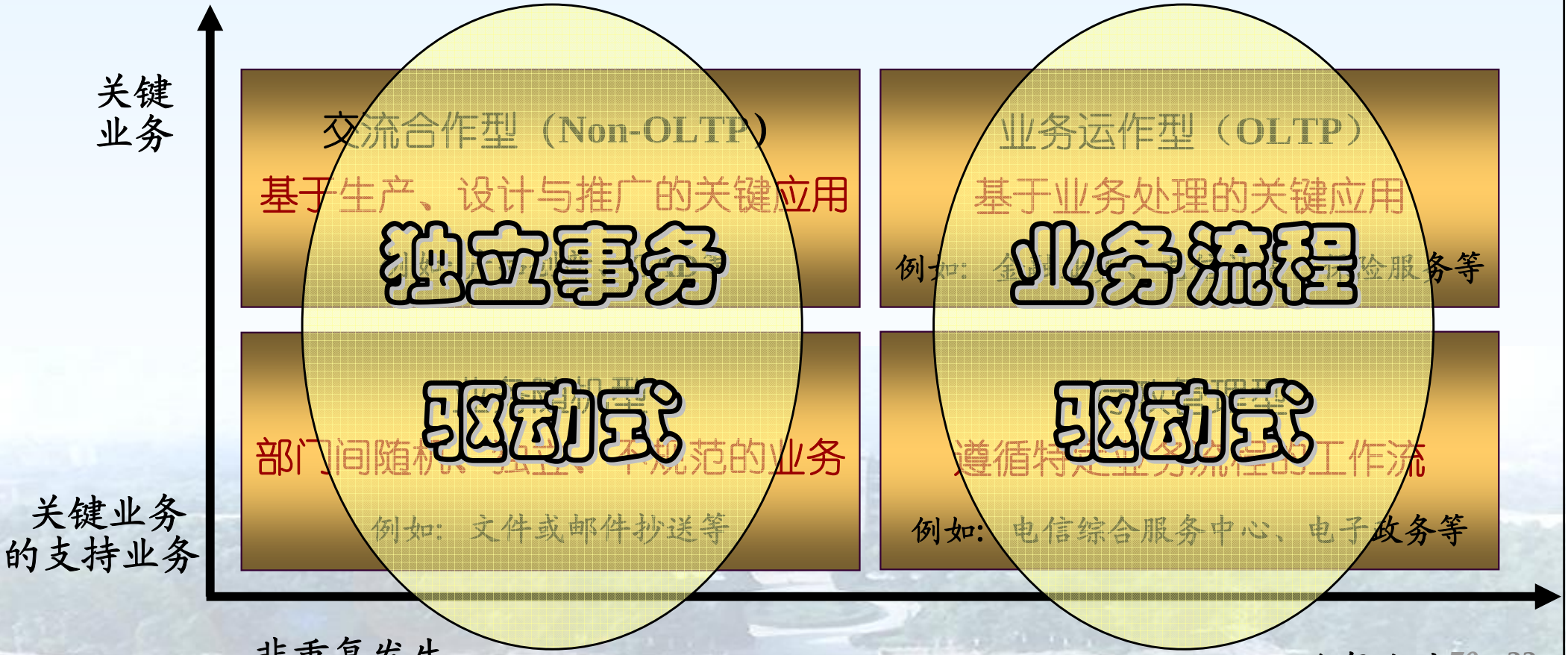
4.13.3 必须重视软件体系结构

- 信息系统的结构分类：

4.13 面向对象系统设计

4.13.3 必须重视软件体系结构

- 信息系统的结构分类：



4.13 面向对象系统设计

4.13.3 必须重视软件体系结构

- 我们看到了什么？
  - 同样的系统部件，组织成不同的结构（形成不同的软件体系结构），就呈现出不同的能力特征。
  - 不同种类的信息系统，其业务特征不同，从设计中重点解决的问题不同，所采用的软件体系结构也不同。
  - 软件体系结构的设计，决定了其后进行的设计的出发点。
  - 即使是很没有水平的用户，再也不会要求先确定硬件、后设计软件体系结构了。
    - 可能的例外是在必须抓紧时间把经费花出去的情况下这样做，不过已经很罕见了。

4.13 面向对象系统设计

4.13.4 特定域软件体系结构（DSSA）

- 在试图作出“放之四海而皆准”的通用应用软件模型遇到了很大困难之后，软件开发人员开始探索如何在特定领域之内，依靠可重用的、面向该领域的软件体系结构，来达到开发软件产品系列的目标。
- **1990**年代初，由美国国防部支持的项目提出了**DSSA**（*Domain Specific Software Architecture*）的概念，并成功地应用于航管、指挥控制、军用交通管制等大型系统的开发。

4.13 面向对象系统设计

4.13.4 特定域软件体系结构（DSSA）

- 第一种定义：DSSA是软件部件的一个集合，以标准的结构拓扑组合而成。它对于一个特定类型的应用领域是通用的，可以用来有效地、成功地建立一个应用系统。
- 第二种定义： DSSA是问题元素和解元素的模式空间，在该空间中定义了问题元素与解元素之间的映射关系。
- 第三种定义： DSSA是一个过程与设施的集合，它支持开发领域的领域模型、参考需求，支持特定领域内一系列软件应用的参考体系结构。
- 我们的定义：DSSA是软件构件的一个集合，它支持开发领域的领域模型，支持特定领域内一系列软件应用的参考需求与参考体系结构。

4.13 面向对象系统设计

4.13.4 特定域软件体系结构（DSSA）

- 领域模型：
  - 对特定领域内任何对应用系统有意义的元素以及元素间关系的表达。
- 参考需求：
  - 对领域内应用系统共同需求的公共表达，可分为功能需求、非功能需求、设计与实现约束，也可分为稳定需求和变化需求。
- 参考体系结构：
  - 适用于领域内一组应用软件系统的公共体系结构（一般用构件（components）、连接件（connectors）和约束（constraints）来表达）。

4.13 面向对象系统设计

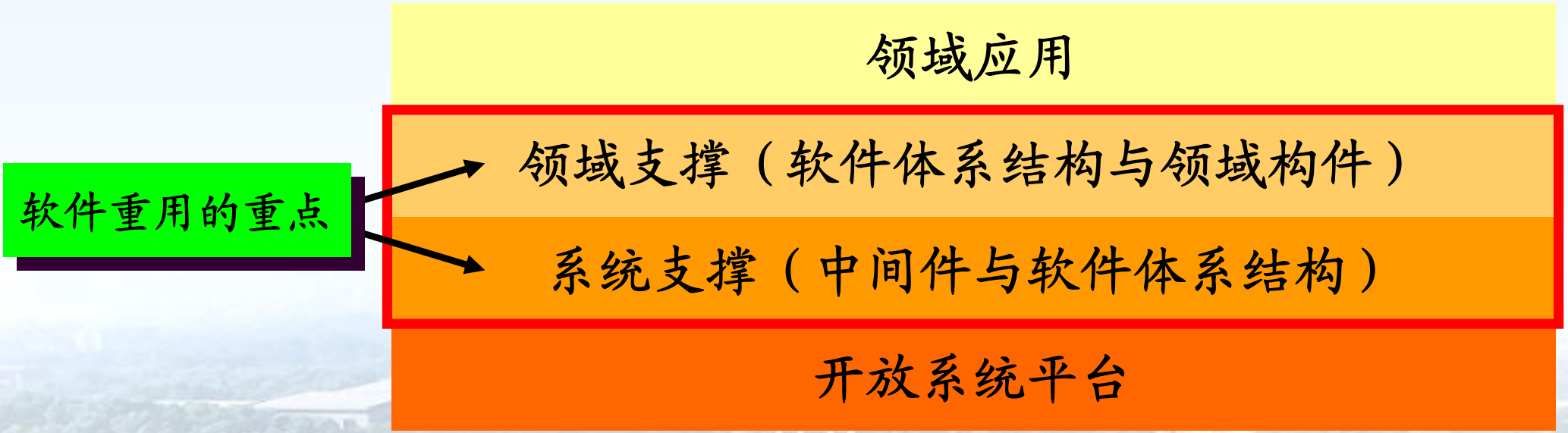
4.13.4 特定域软件体系结构（DSSA）

- 未采用DSSA技术的系统层次结构:

4.13 面向对象系统设计

4.13.4 特定域软件体系结构（DSSA）

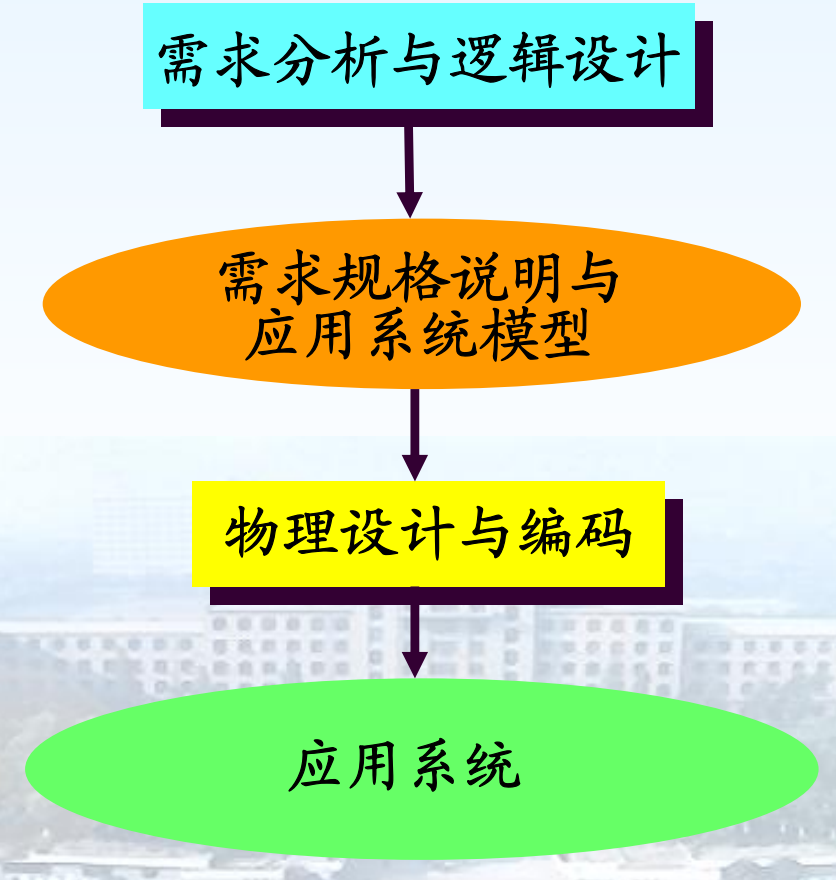
- 采用了DSSA技术的系统层次结构:



4.13 面向对象系统设计

4.13.4 特定域软件体系结构（DSSA）

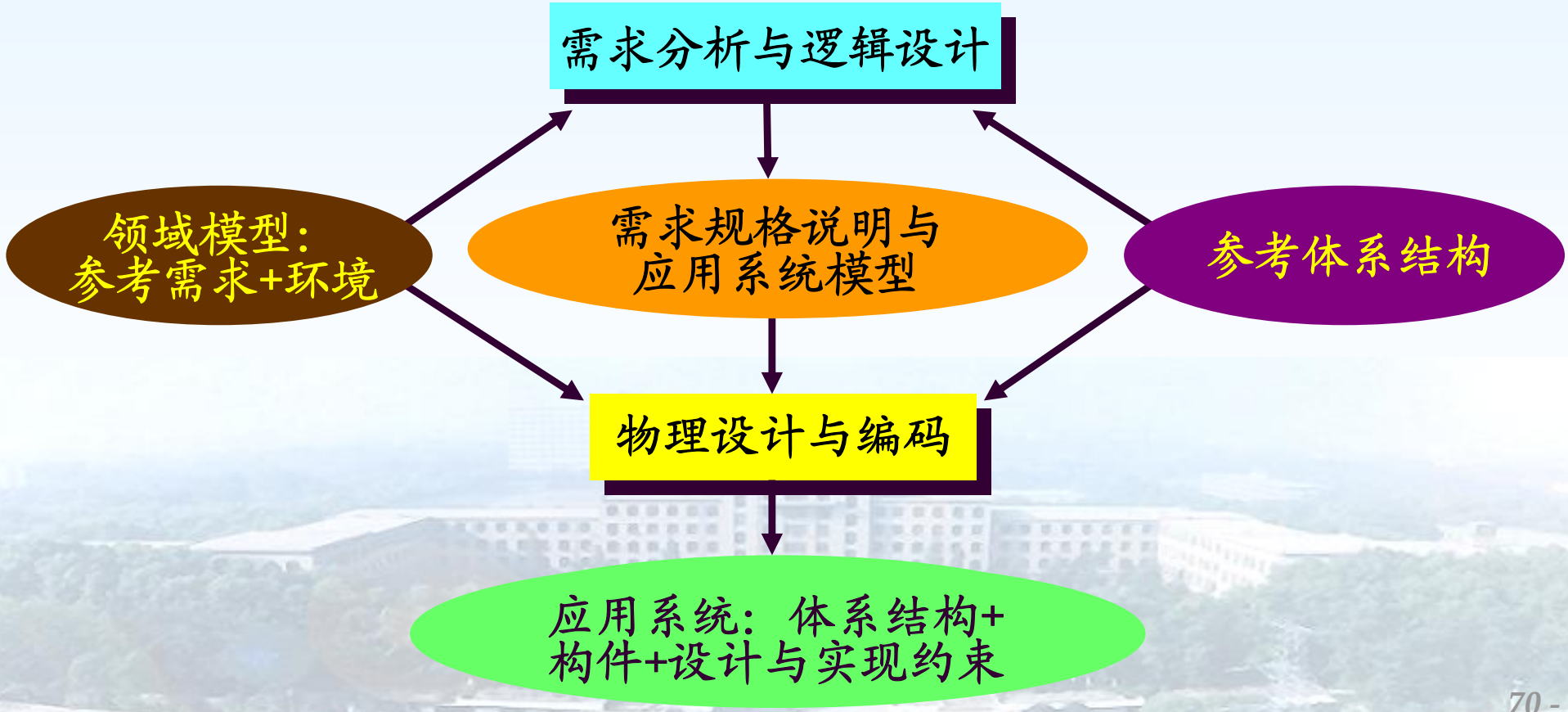
- 常规的系统开发模式:



4.13 面向对象系统设计

4.13.4 特定域软件体系结构（DSSA）

- 基于DSSA的系统开发模式:



4.13.4 特定域软件体系结构（DSSA）

- 特定领域的领域模型是相对稳定的;
- 特定领域内对应用系统的需求，可以提取出公共的部分;
- 在一段时间内，特定领域内的应用系统所采用的软件体系结构可以归纳出若干种;
- 在上述前提下，基于DSSA的开发对分析、设计和实现都有很好的重用度;
- 必须考虑建立DSSA所需的成本（约为同样应用系统的 3 倍）。

4.13 面向对象系统设计

4.13.4 特定域软件体系结构（DSSA）

- 对于软件重用（*software reuse*）的再认识：
  - 函数重用：可重用函数经链接或动态加载后的重用（程序重用）；
  - 继承重用：类库或构架的继承重用（程序重用与部分设计重用）；
  - 构件重用：基于通用支撑平台和标准接口的重用（与语言/平台无关的程序重用与部分设计重用）；
  - 基于统一对象模型的重用：对面向对象分析、设计与实现结果的重用（依赖于具体需求的软件开发全过程重用）；
  - 基于**DSSA**的重用：对领域模型和参考体系结构的重用（强调领域特征的、全面的需求与设计重用）。

4.13 面向对象系统设计

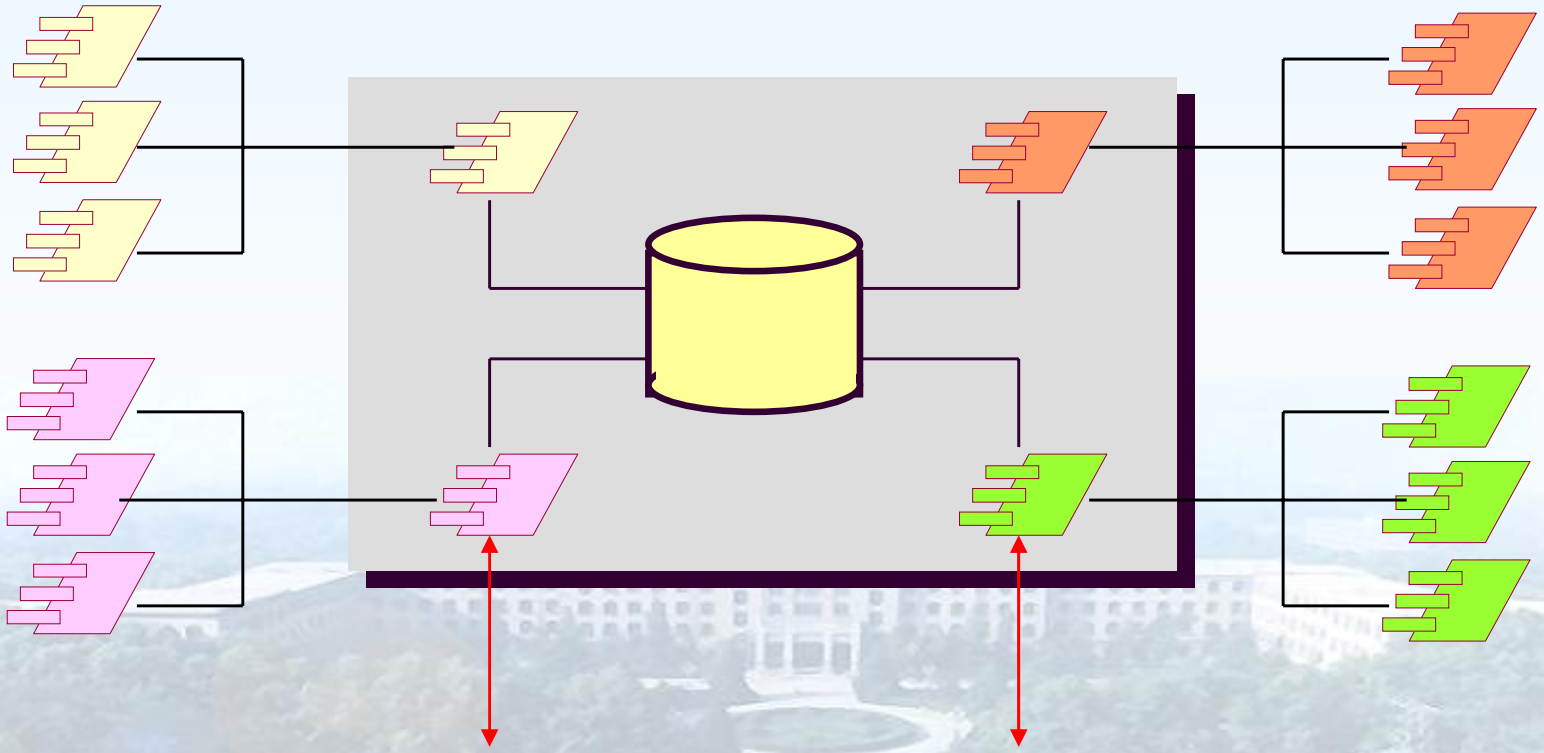
4.13.5 典型信息系统的软件体系结构

- 金融系统（领域特征）：
  - 交易完整性是金融系统最关键的问题之一。
  - 将数据和交易高度集中以形成数据处理中心是目前的趋势。
  - 非传统金融业务的不断扩充，需要设立中间业务转换和隔离。
  - 中间业务平台可以集中在传统业务服务一端，也可以分布在区域中心一端。
  - 要求支持面向用户的业务逻辑设计支持。

4.13 面向对象系统设计

4.13.5 典型信息系统的软件体系结构

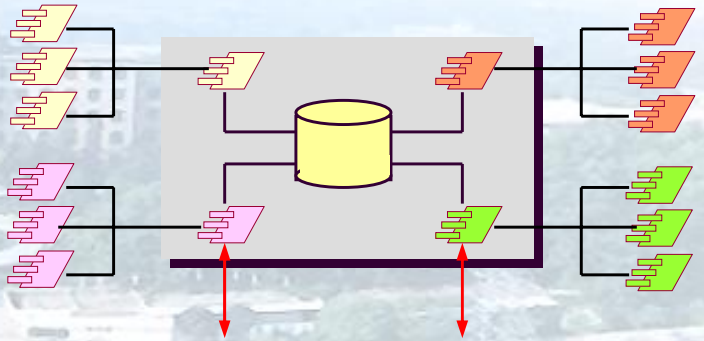
- 金融系统（一般结构）：



4.13 面向对象系统设计

4.13.5 典型信息系统的软件体系结构

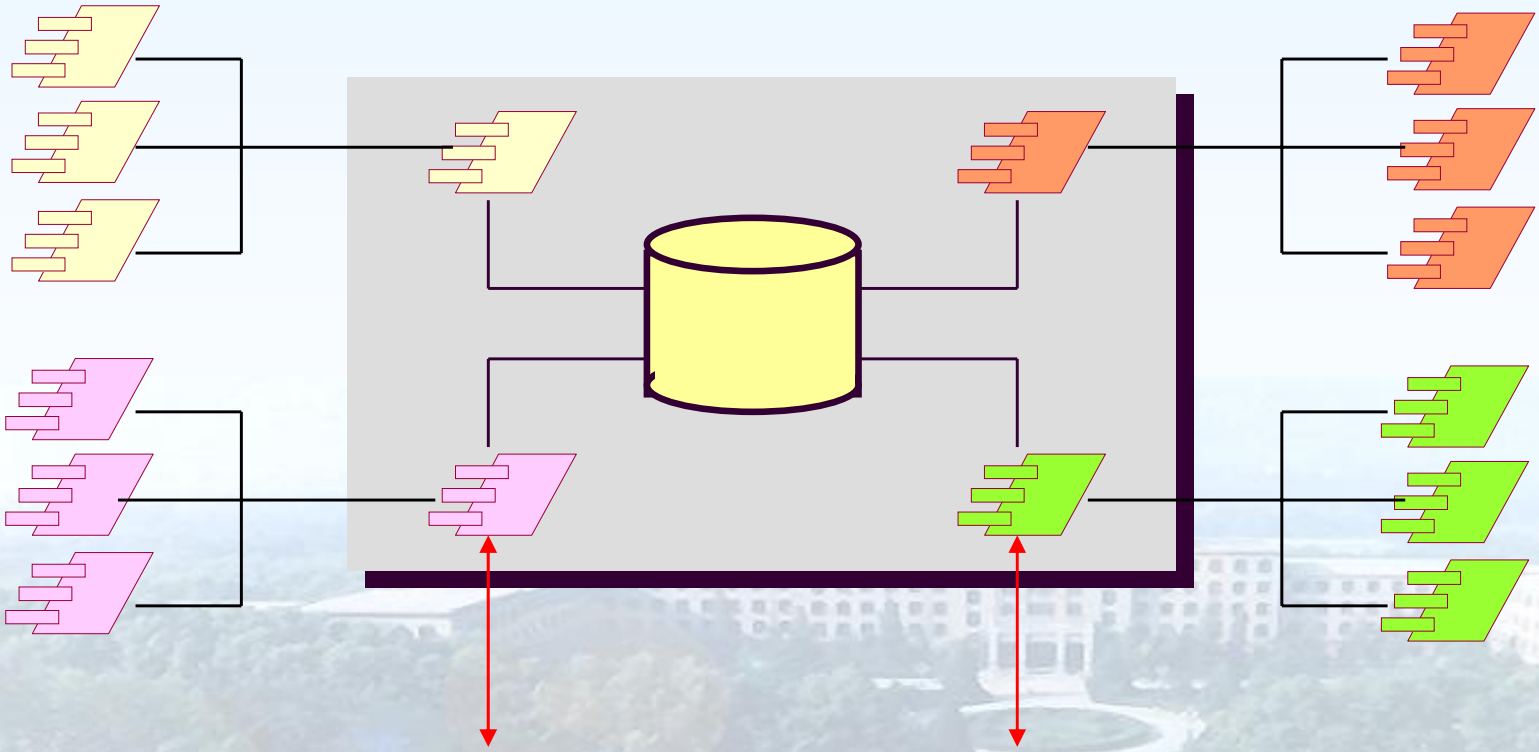
- 金融系统（结构特征）：
  - 系统的中心是数据库。数据库是保持、协调和传递系统状态的中心，服务进程之间很少有直接的协同；
  - 交易由客户进程或外部系统激发，服务进程根据交易请求和相关的数据库状态，决定应用逻辑的具体处理；
  - 服务进程一般是分类的，如储蓄、对公、帐务、会计、清算等，有的系统还按业务种类（如开、存、取、清）对服务进程进行分类。



4.13 面向对象系统设计

4.13.5 典型信息系统的软件体系结构

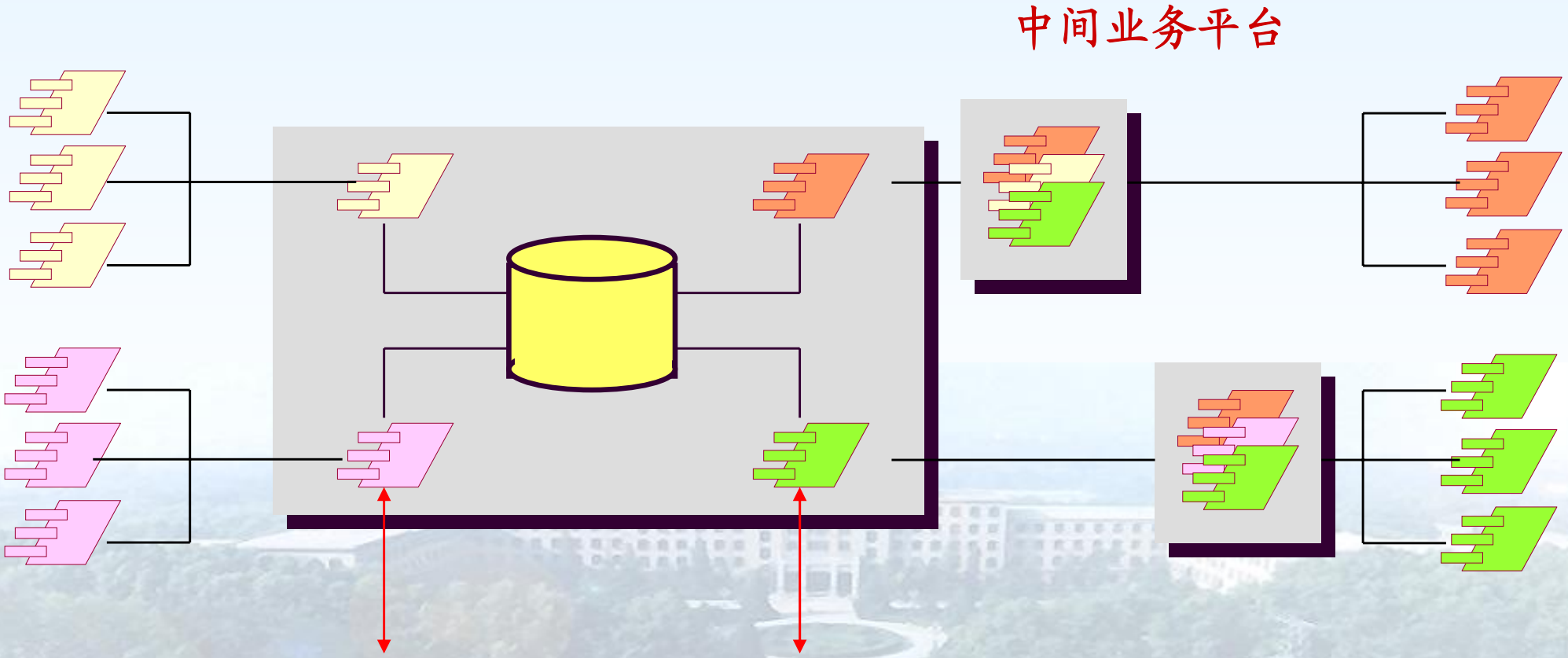
- 金融系统（结构演化）：



4.13 面向对象系统设计

4.13.5 典型信息系统的软件体系结构

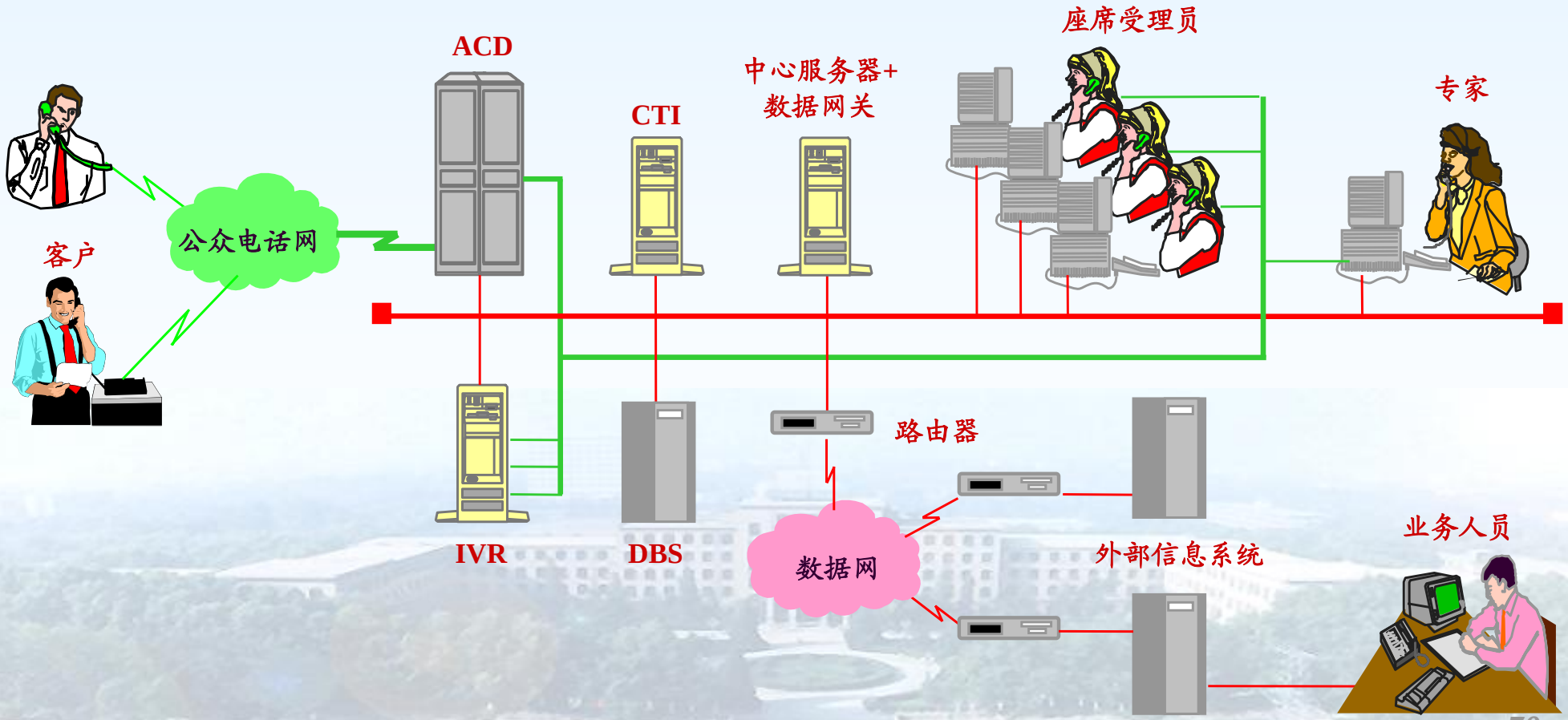
- 金融系统（结构演化）：



4.13 面向对象系统设计

4.13.5 典型信息系统的软件体系结构

- 呼叫中心（一般结构）：



4.13 面向对象系统设计

4.13.5 典型信息系统的软件体系结构

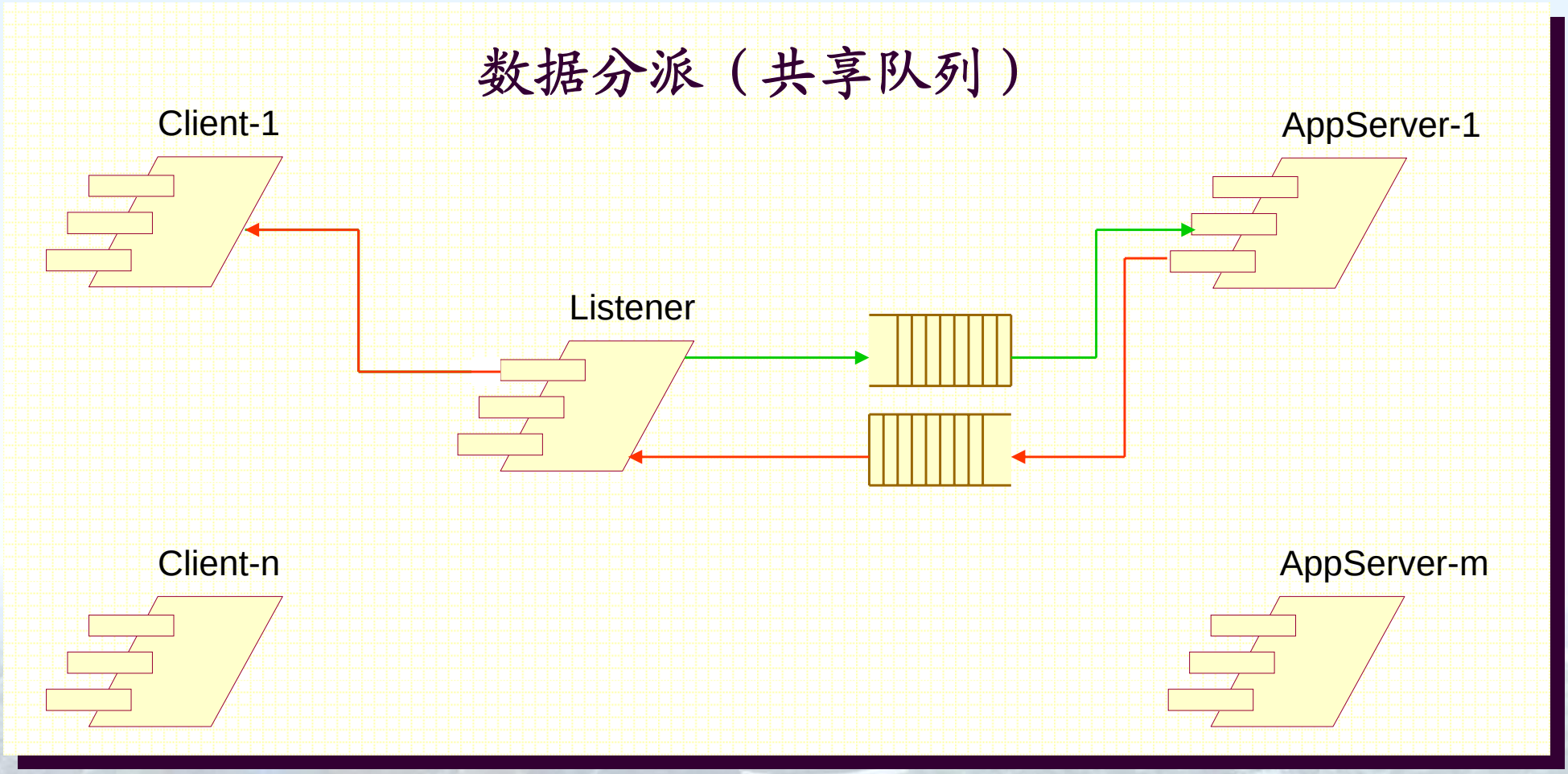
- 呼叫中心（结构特征）：
  - 系统依靠一组服务进程（按照请求的种类）动态构成的处理链进行处理，并不是所有的操作都与数据库有关。
  - 请求的载体是多样化的，但必须转换成统一的协议，否则一旦请求载体扩充将导致本系统的大面积变更。
  - 外部系统多样化，必须在体系结构设计上就考虑隔离外部系统的变更对本系统的影响。
  - 相当一部分业务操作属于工作流（*Working Flows*）类型。
  - 在处理中必须积累相关信息，以利于客户关系管理（*CRM*）和数据挖掘。

4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构

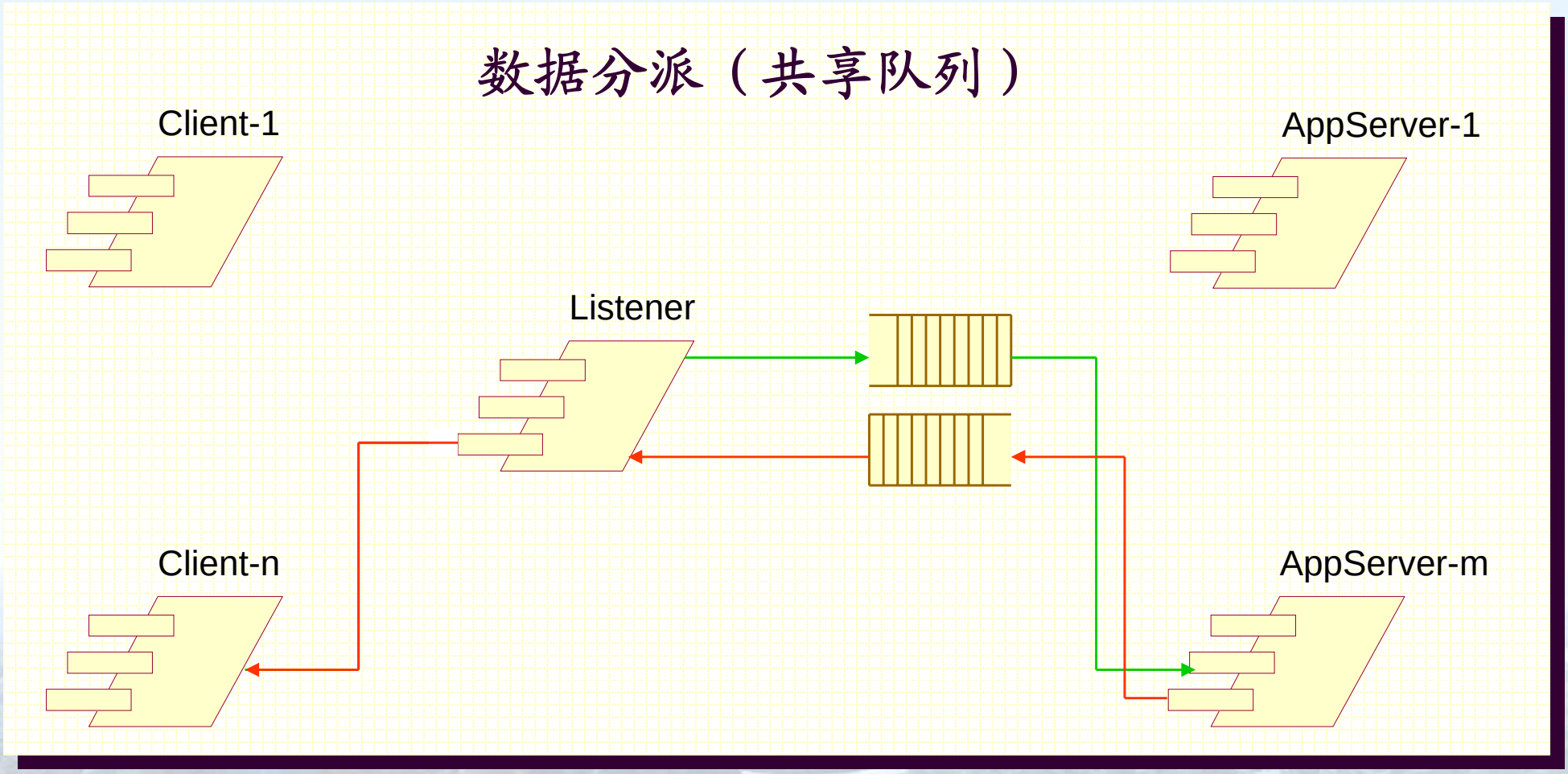
4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构



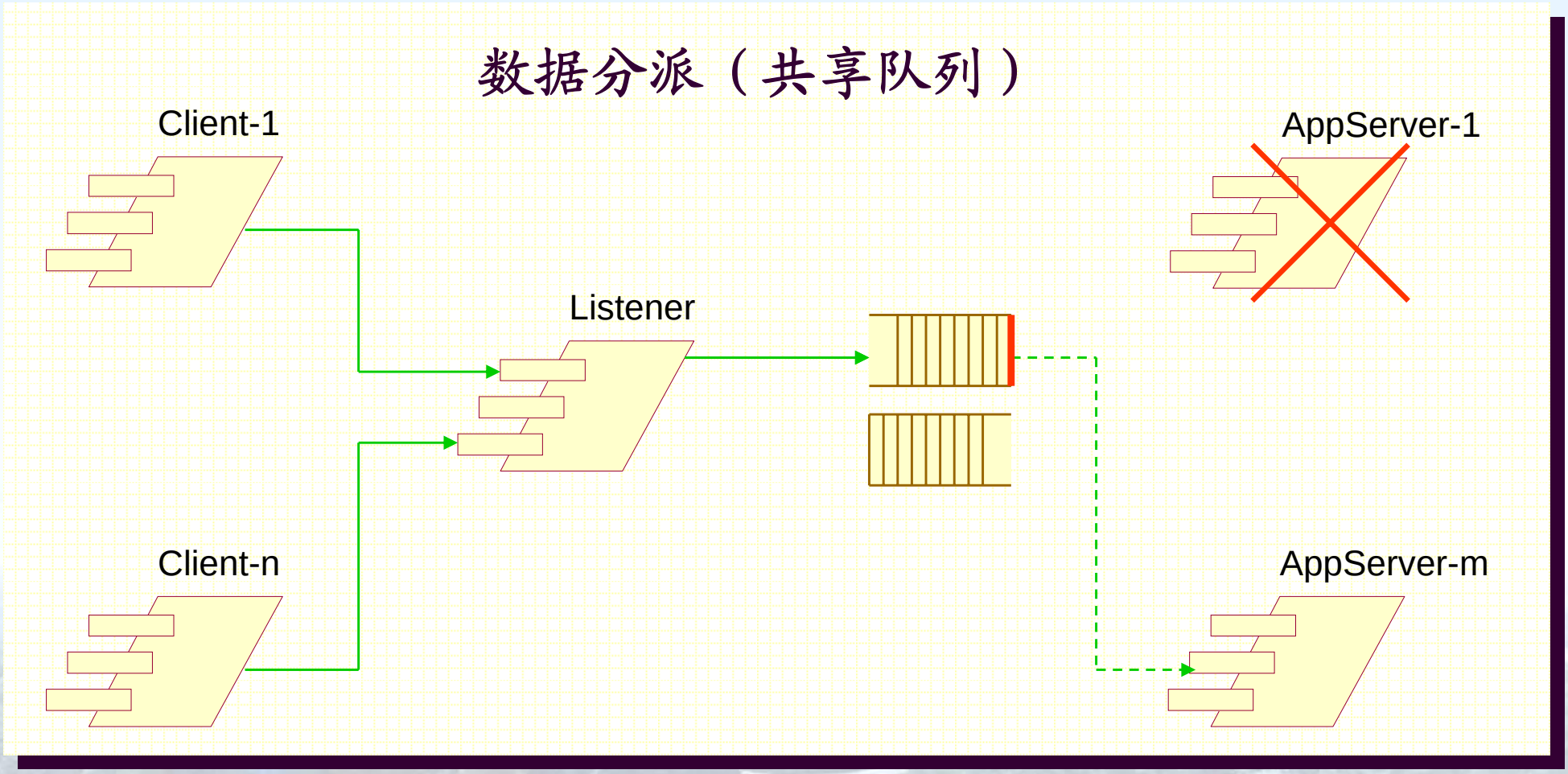
4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构



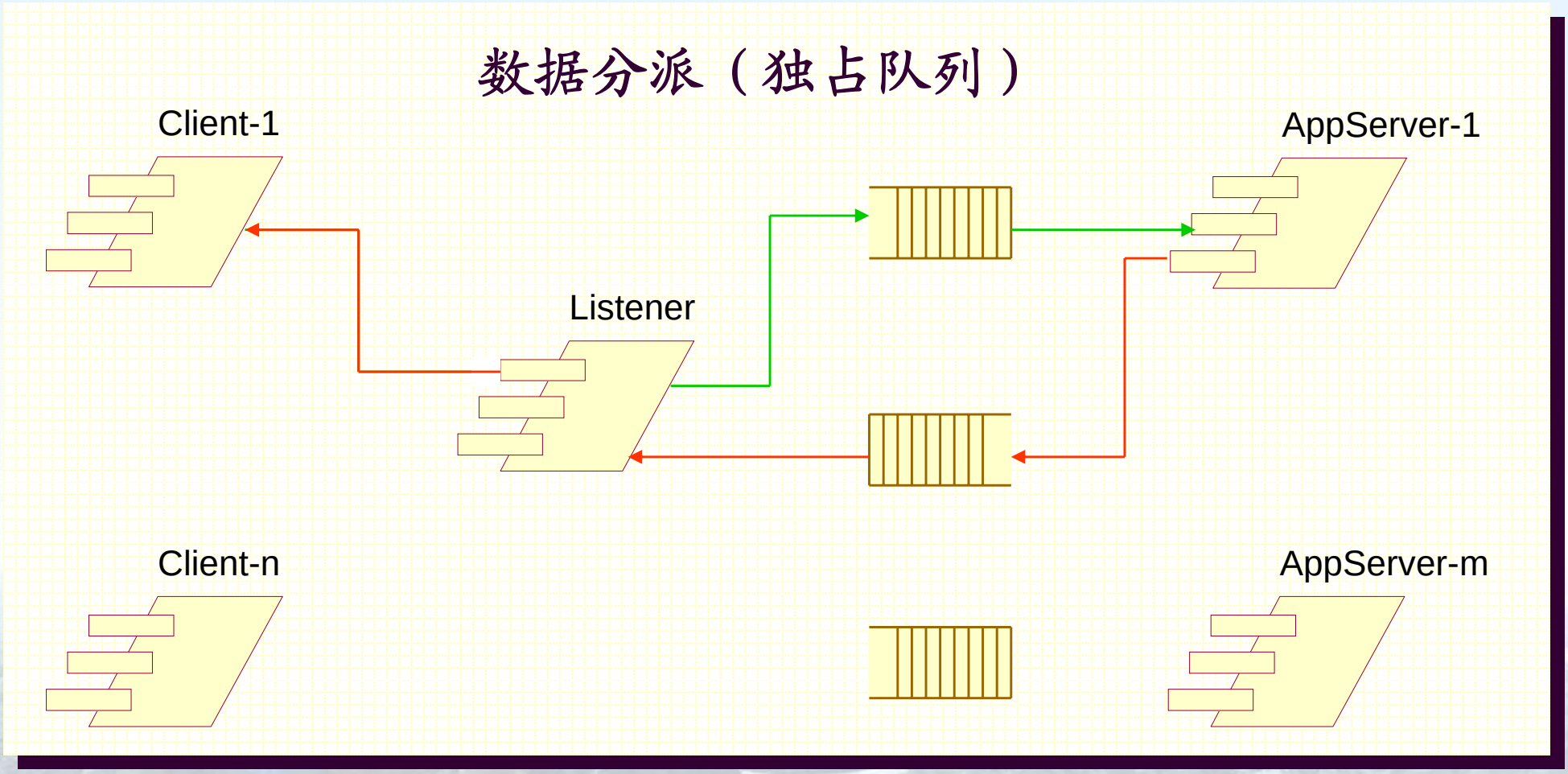
4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构



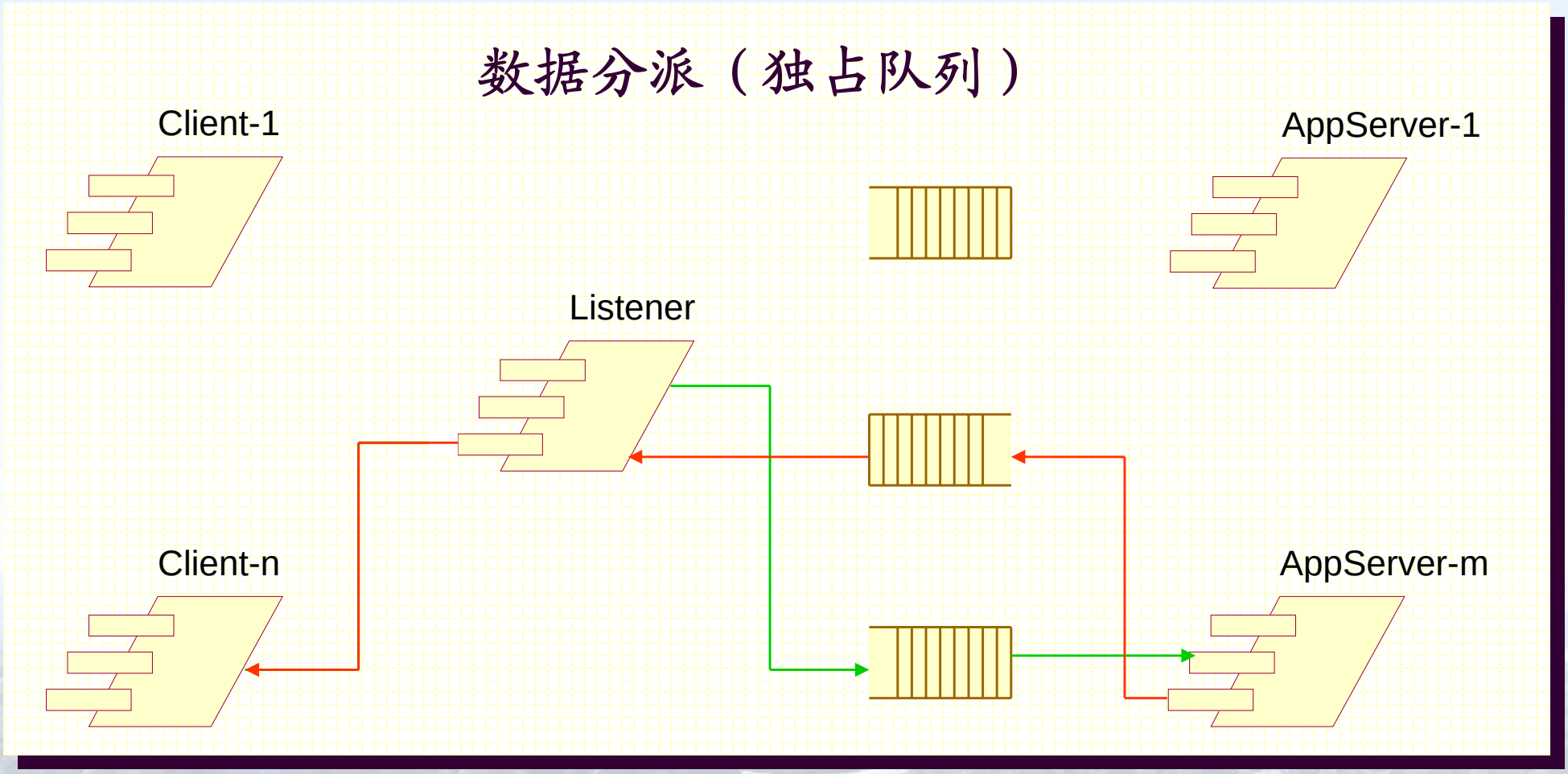
4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构



4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构

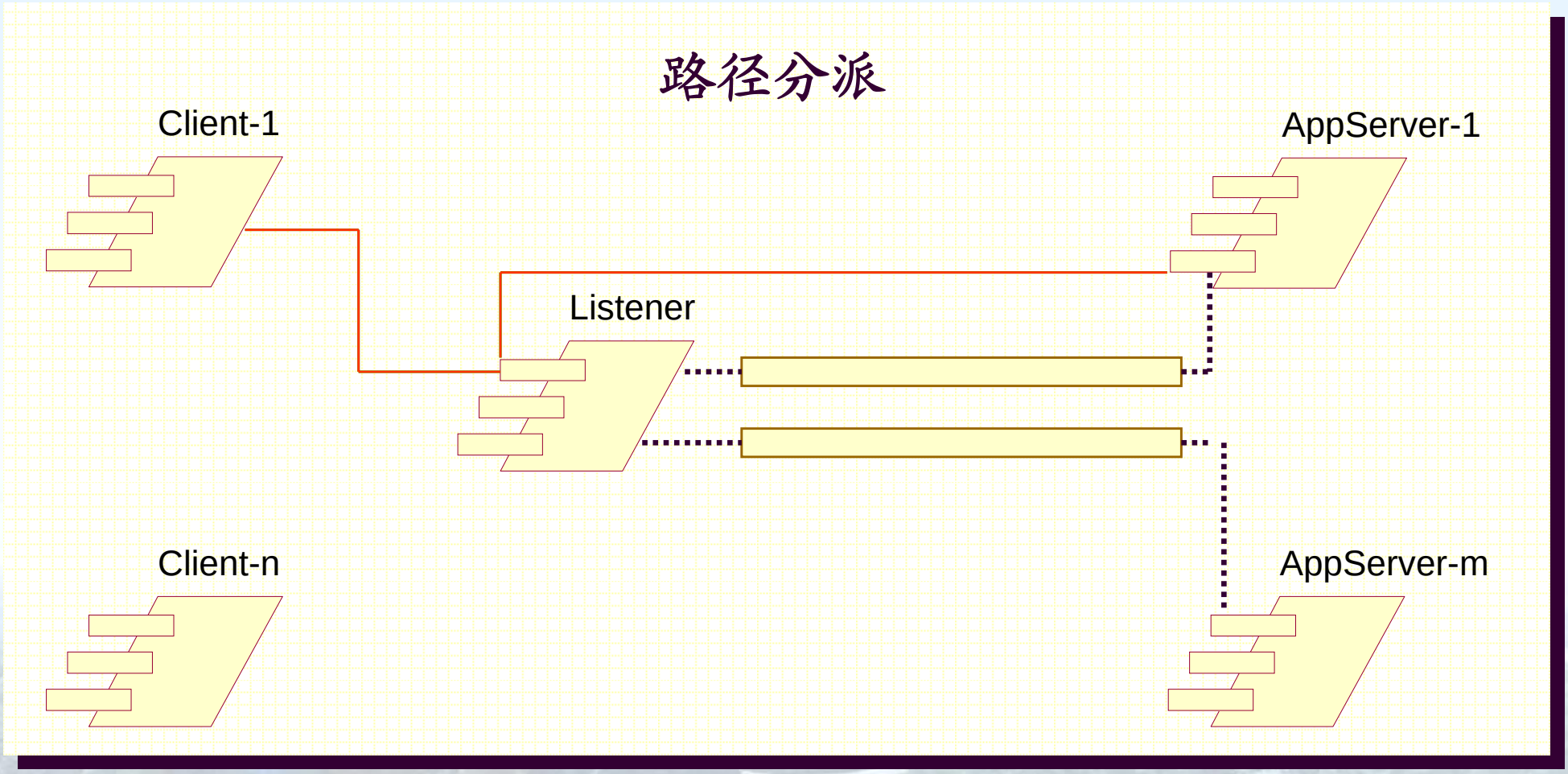


4.13.6 典型的分布、并发控制结构

- 数据分派的特点：
  - 服务器与客户机的通信集中在监听进程，其他服务进程不含与客户机的通信机制，只需要规定与监听进程的交互协议；
  - 监听进程要串行完成接收客户机请求、分配请求、获取响应、回发响应、管理路径等任务，当网络速度和服务速度提高后，这个进程可能成为系统性能与可靠性的一个瓶颈；
  - 服务进程共享队列可能因个别服务进程故障而锁死队列。

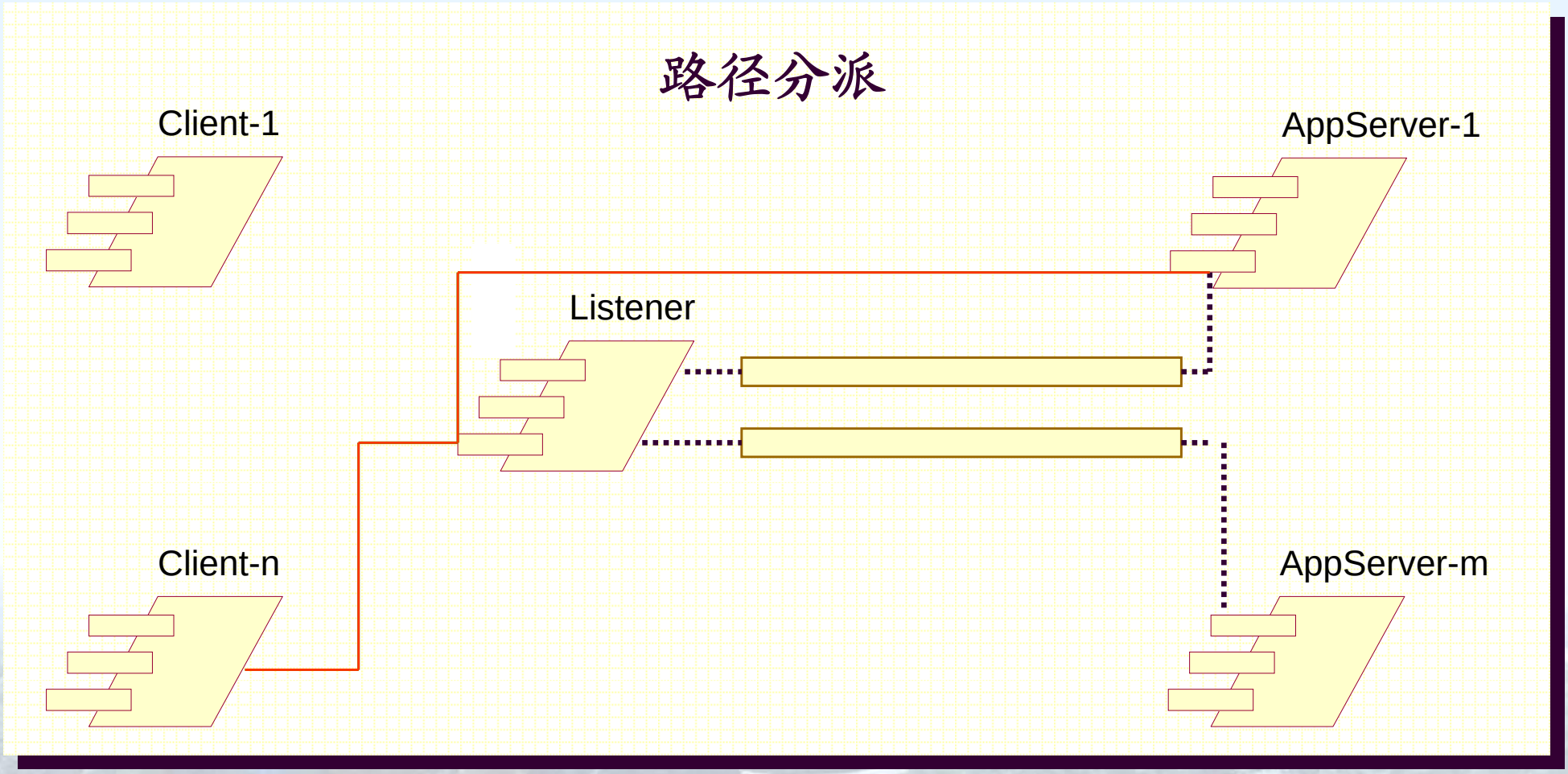
4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构



4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构



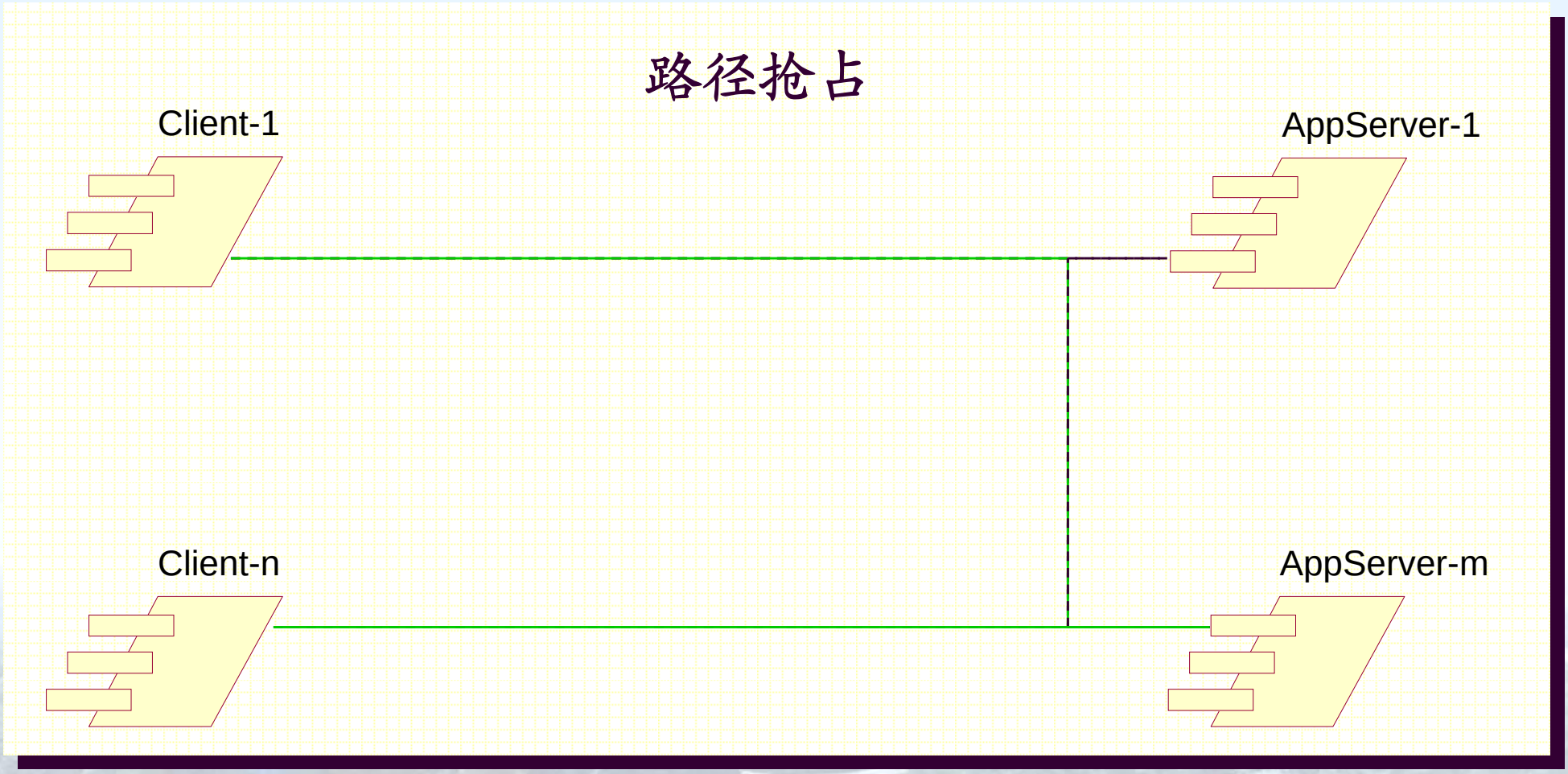
4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构

- 路径分派的特点：
  - 监听进程只负责监听请求、分派路径和管理路径，不承担接收和发送，负载比数据分派轻，所构成的并发系统效率高；服务进程与客户机进程之间的应用层协议可以很灵活；
  - 各个服务进程需要支持与客户机的通信协议；需要有在进程间传送路径的机制支持。

4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构



4.13 面向对象系统设计

4.13.6 典型的分布、并发控制结构

- 路径抢占的特点:
  - 每个服务进程都有监听功能，没有专门的监听进程，并发系统的性能和可靠性较高。这是目前Web Server所使用的并发结构;
  - 由于所有服务进程监听同一个连接表，因此在抢占时要对它进行互斥，以免一个对话被多个服务进程同时抢到，这需要付出开销；由于没有起控制作用的专用监听进程，因此进行负载均衡需要附加机制。

4.13 面向对象系统设计

4.13.7 模式（Patterns）

- 问题的提出：面向对象技术要求使用者有正确的观念，熟练掌握分析与设计技能，才能充分发挥对扩充和重用的作用，但这对于一般技术人员而言，有一段较长的过程。
- 与软件技术面临的问题相近、可以为软件技术领域所借鉴的一个传统工业领域：建筑。
  - 与环境融为一体（如中山陵）；
  - 造型与风格（如悉尼歌剧院、陕西省历史博物馆）；
  - 适应现有条件（如卢浮宫金字塔）；
  - 结构（如香港中国银行）；
  - 质量（反例如北京西客站、重庆彩虹桥）；
  - 造价与工期;
  - ……

4.13 面向对象系统设计

4.13.7 模式

- 1964 年，著名建筑学家 Christopher Alexander 出版了一本书：*Notes on the Synthesis of Form*。在此书中，他提出了 Form（建筑形式）的概念，认为设计师可创造 Form 来化解环境中互相冲突的需求，使冲突变成为和谐的景象。同时，他也提出了 Pattern（模式）的概念，可引导设计师逐步创造出 Form。1971 年，该书再版上市，此时正是结构化设计方法的萌芽阶段，该书对当时 Ed Yourdon 和Larry Constantine 的结构化观念的诞生具有相当大的影响力。

4.13 面向对象系统设计

4.13.7 模式

- **1972~1985** 年，**Alexander** 任教于 **UC Berkeley**，他和其同事共同研究模式，出版了四本书：
  - *The Timeless Way of Building*：完整地介绍了模式及模式语言的概念。
  - *A Pattern Language*：列举了 **253** 个建筑方面的模式。
  - *The Oregon Experiment*：叙述了在 **Oregon** 大学的实验过程。
  - *The Production of Houses*：叙述了在墨西哥的实验，也详述了这实验并未成功的原因。
- **Alexander** 发明的模式语言，用来描述一系列建筑方面的问题、这些问题的约束条件以及解决这些问题的方法，不仅为建筑设计提供了一种通用的语言，可用于描述以往的设计经验，以利于沟通，而且使得以往的设计经验可以被重复使用。

4.13 面向对象系统设计

4.13.7 模式

- 受到这一思想的启发，人们从 **1980** 年代末期开始尝试将模式这一概念用到软件工程中去。到 **1990** 年代之后，设计模式便渐渐成为软件界（尤其是面向对象领域）的一个热门话题。
- **1991** 年，**Gamma** 完成了他的博士论文－《*Object-Oriented Software Development based on ET++: Design Patterns, Class Library, Tools*》。
- **1995**年，**Gamma**、**Helm**、**Johnson** 和 **Vlissides**（俗称四人帮）撰写了一部名著：《*Design Patterns: Elements of Reusable Object-Oriented Software*》，奠定了这个领域的基础。

4.13 面向对象系统设计

4.13.7 模式

- 模式在设计领域的成功，使之推广到分析领域 – 分析模式。实际上，二者只有细致程度的区别。
- 即使是在设计领域，模式也被细分为体系结构模式和设计模式:
  - 前者主要体现了体系结构的设计经验;
  - 后者则主要体现了类之间关联结构的设计经验。

4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

- 典型的体系结构模式（*Architectural Patterns*）
  - Distributed
  - Event-driven
  - Frame-based
  - Batch
  - Pipes and filters
  - Repository-centric
  - Blackboard
  - Interpreter
  - Rule-based
  - Layered
  - MVC
  - IR-centric（信息查询为中心）
  - Subsumption（包容式）

4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

- 体系结构模式的分类:

类别	特征	典型模式
Data flow	dominated by motion of data through the system	◆ batch sequential ◆ data flow network ◆ pipes and filters
Call and return	dominated by order of computation	◆ main program/subroutines ◆ abstract data types ◆ object ◆ call based ◆ client/server ◆ layered

4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

- 体系结构模式的分类:

类别	特征	典型模式
Independent components	dominated by communication patterns	• event systems • communicating processes
Data-centered	dominated by a complex central data store, manipulated by independent computations	• repository • blackboard
Virtual machine	characterized by translation of one instruction set into another	• interpreter

4.13 面向对象系统设计

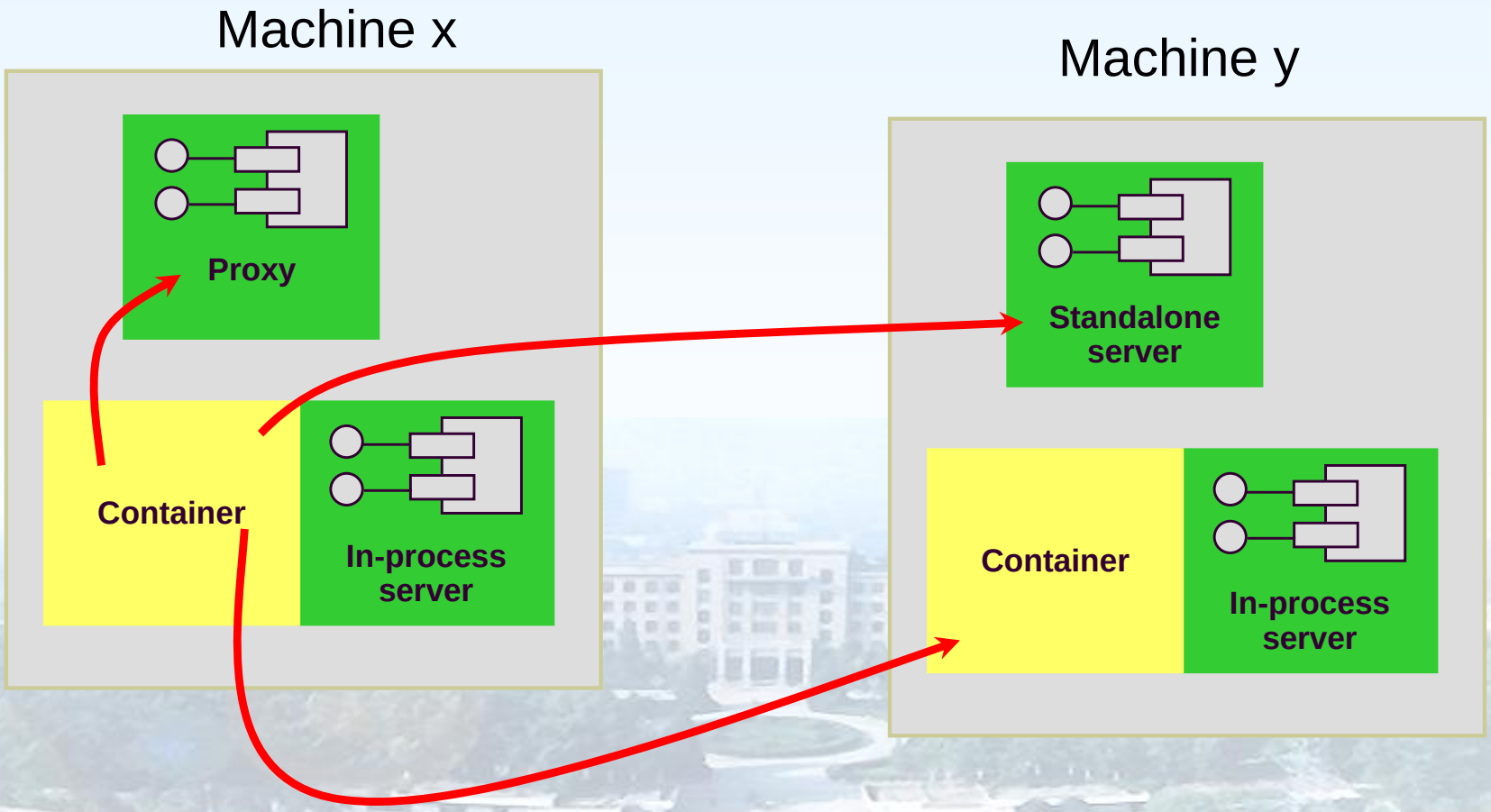
4.13.7 模式– 体系结构模式

- 一些商品化平台 / 构架已经提供了基本的体系结构支撑:
  - MTS / MSQS (Microsoft Transaction Server / Microsoft Message Queue)
  - CORBA
  - Enterprise Java Beans (EJB)
  - Domino
  - SAP R/3（**SAP** 公司的 **ERP** 产品）
  - Delphi
  - Forte（**Sun** 公司的 **Java** 开发平台）
  - Visual Basic

4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

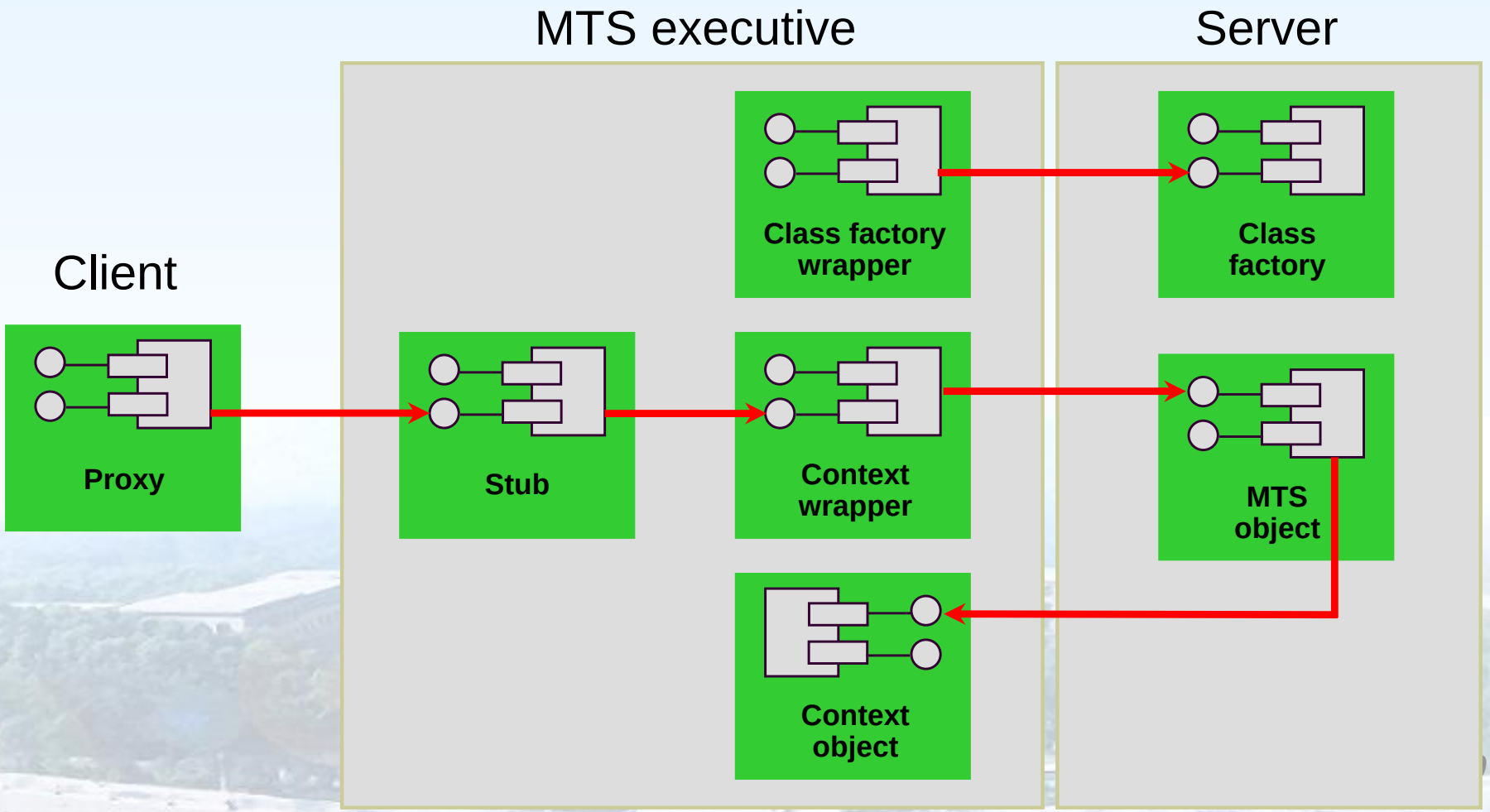
- DCOM的体系结构:



4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

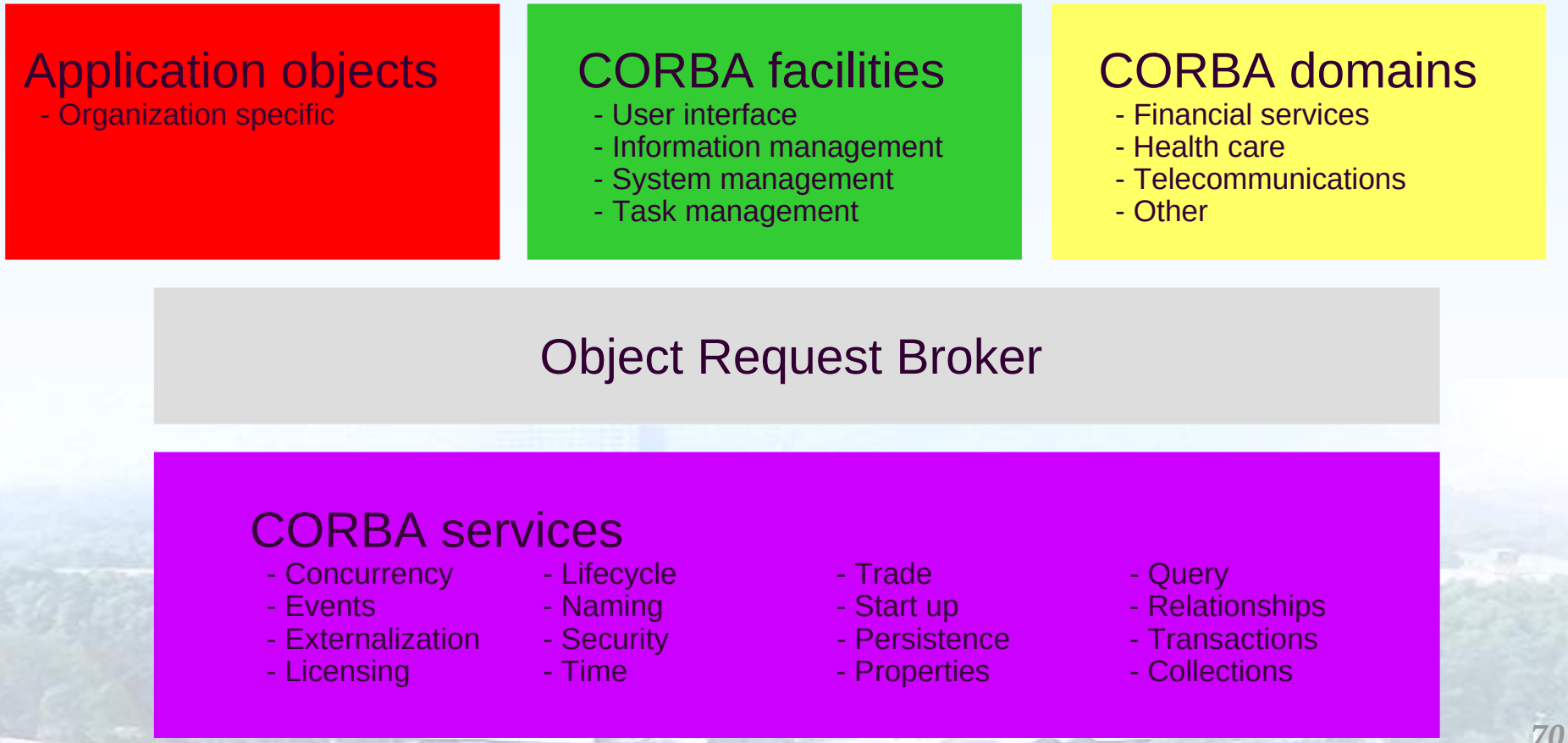
- MTS 的体系结构:



4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

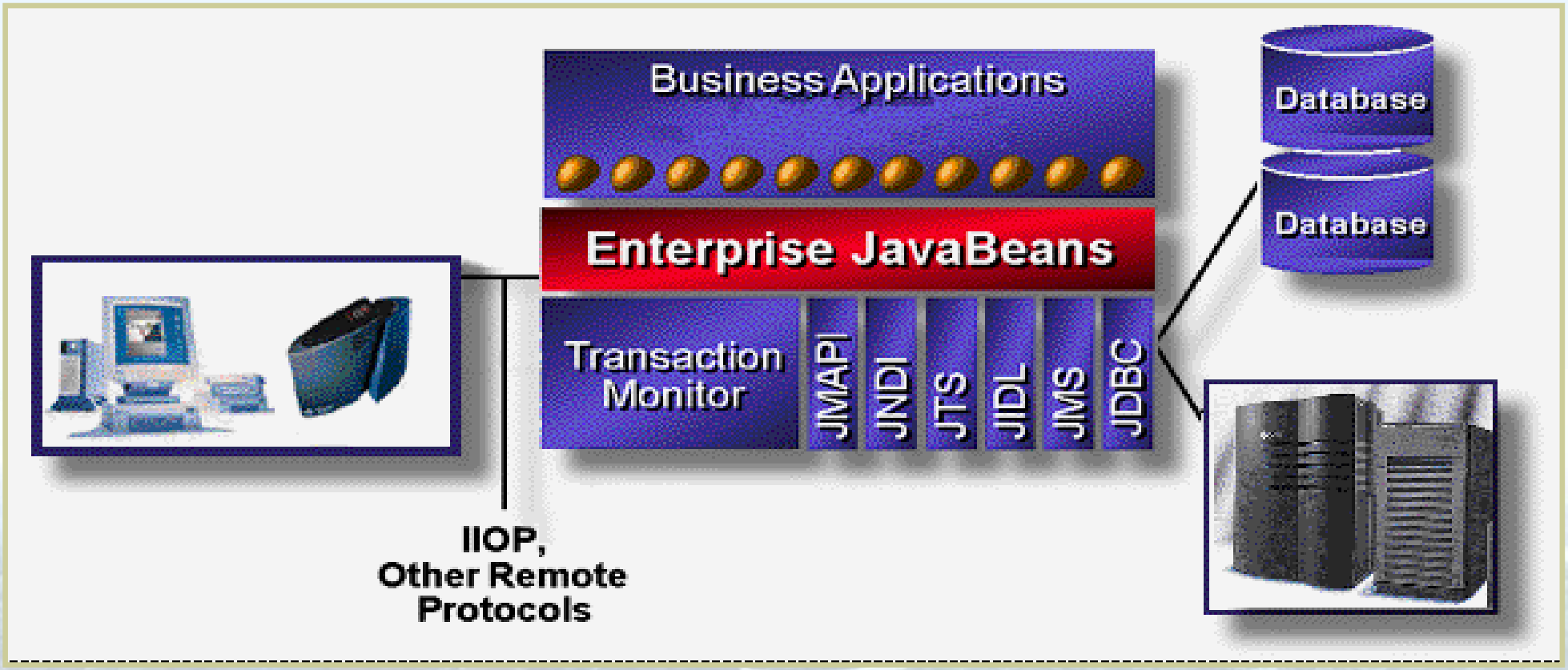
- **CORBA** 的体系结构:



4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

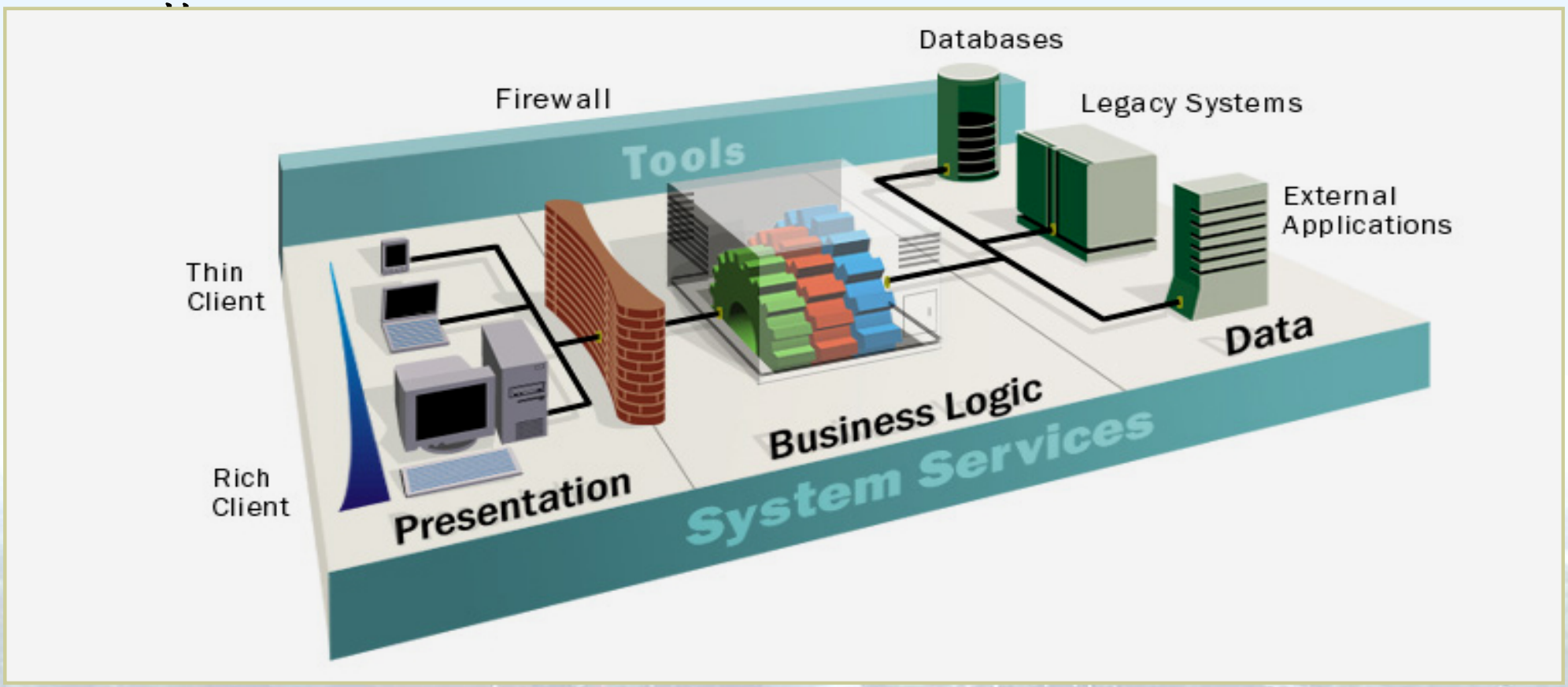
- EJB 的体系结构:



4.13 面向对象系统设计

4.13.7 模式– 体系结构模式

- **Microsoft DNA (Distributed interNet Application) 的体系结**



4.13 面向对象系统设计

4.13.7 模式– 设计模式

- 设计模式是一个抽象的方案，它可以解决一类问题。设计模式往往首先描述一类相似的问题，然后提出一个抽象而通用的解决方案，使用这一解决方案可以解决所描述的那些问题。
- 设计模式一般由三个主要部分组成:
  - 类的结构及对象交互的抽象描述;
  - 所涉及的问题（这决定了该设计模式的应用范围）;
  - 应用的后果（这决定了在某些约束下是否应使用该设计模式）。
- 设计模式比构架的规模小得多，易于组合与理解。
- 设计模式的应用难点在于如何将实际问题加以分解，使之与既定的设计模式匹配。

要点与引伸

- 特定领域指的是应用领域，而不是技术领域。
- 在特定领域中，不同应用系统的需求和逻辑设计实际上是具有共性的，抽象后得出的参考需求和参考体系结构可以被重用，而且如果重用成功，其效果将远远超过程序的重用。
- 参考体系结构是目标体系结构的模板和基础，其构件化有利于适应目标系统的组装、剪裁和配置要求；构件化的目标系统则可以为参考体系结构提供了扩充的基础。
- 对象技术是构件化的基础，因而是实现参考体系结构的基本要素；模式技术则是表示参考需求和参考体系结构、体现其模板作用的有效手段。

要点与引伸（续）

- 模式是系统分析和设计的“套路”，应当重视前人的经验。

下一次课的内容

- 面向对象系统分析与设计－面向对象系统设计
  - 对交互图进行精制－基于模式的设计
  - 对象间关联方式的考虑
  - 产生设计类图
  - 设计模式及其应用
    - 生成模式
    - 结构模式
    - 行为模式
- 分析模式简介
- 布置第三次作业