第八章 进行体系结构设计

王美红

- 体系结构设计需要做的事情：为你提供整体的视图并保证得到正确的理解
- “系统架构师”根据软件需求分析得到的需求为系统选择合适的体系结构风格。数据库或数据仓库设计者为系统创建数据体系结构。
- 工作产品：在体系结构设计过程中，要创建一个包含数据结构和程序结构的体系结构模型。同时，还需描述构件属性以及构件之间的相互关系（交互关系）。

主要内容

- 软件体系结构
- 数据设计
- 体系结构风格和模式
- 体系结构设计
- 评估可选的体系结构设计
- 映射数据流到软件体系结构

8.1 软件体系结构

- 什么是体系结构？
  - 建筑类的体系结构
    - 整体形状
    - 纹理、颜色和材料在一起创建外部门面和内部“居住环境”
    - 灯光设计、地板类型、壁挂布置等等
    - 可以是数千个“大大小小的决策”

8.1 软件体系结构（续）

- 软件体系结构定义一（P139）
  - 一个系统的“结构元素和其相应的接口”以及系统中各个构件和子系统的“行为”。

8.1 软件体系结构（续）

- 软件体系结构定义二：
  - 描述一个计算机软件的各个功能模块内部关系以及不同模块之间的关系的构造图（可以是文字的描述，也可以是图表的描述），类似于硬件设计中的原理框图，它是软件流程图的上层抽象。

8.1 软件体系结构（续）

- 软件体系结构定义三：
  - 一个程序和计算机系统软件体系结构是指系统的一个或者多个结构。结构中包括软件的构件，构件的外部可见属性以及他们之间的关系。
- 是一定程度上的抽象，非可运行的软件。

8.1 软件体系结构（续）

- 软件体系结构定义四：
  - 系统体系结构是一个综合模型，系统体系结构是由许多结构要素及各种视图（或观点）（View）所组成的，而各种视图主要是基于各组成要素之间的联系与互操作而形成的。所以，系统体系结构是一个综合各种观点的模型，用来完整描述整个系统。

表 4.3　主要的架构视角

视　角	说　明	模式示例
子系统和组件视图	这个视图确定了最大范围内的系统组织单元，并详细说明了它们的职责以及服务和数据的分配，定义了它们提供的和所需的接口	分层模式 微内核模式 递归包含模式 分层控制模式 端口模式
并发性和资源视图	这个视图确定了并发单元以及相关的并发元数据、调度策略和资源共享策略	循环执行模式 静态优先级模式 动态优先级模式 中断模式 保护调用模式 消息队列模式 集合点模式
部署视图	这个视图确定了所涉及的各个工程学科（如软件、电子、液压、气动、光学等），每个学科的职责，以及这些学科之间的接口	静态分配模式 硬件代理模式 硬件适配器模式 中介模式 去抖动模式
分布视图	这个视图确定了跨多个地址空间（单核或多核）分布软件的策略，元素如何发现通信基础设施的服务和协作，包括网络拓扑结构、中间件和通信协议	共享内存模式 观察者模式 代理模式 端口代理模式 数据总线模式 代理程序模式
可靠性视图	这个视图负责解决安全性、可靠性和安全保障性问题，以及正常和异常功能应该如何处理这些问题	受保护的单通道模式 同构冗余模式 异构冗余模式 CRC 模式 智能数据模式 基于代理的防火墙模式 安全通道模式

• 这些模式的详细信息可以在作者的《实时设计模式》[5]和《C 嵌入式系统设计模式》[2]或其他参考资料中找到。

8.1 软件体系结构（续）

- 凭借体系结构图，软件开发人员可以：
  - 分析设计能否完全满足需求
  - 为设计中某些方面的变更提供指导
  - 降低由于软件构造不合理带来的风险
- 是蓝图，是从架构师到设计师再到负责开发不同系统构件的软件工程师之间传递准确信息的工具。

8.1 软件体系结构（续）

- 一个系统的多个不同视图，每个视图从不同利益相关者不同关注点对整个系统进行的表示。
- 开发者角度：是从架构师到设计师再到负责开发不同系统构件的软件工程师之间传递准确信息的工具。
  - 可以分析设计能否完全满足需求；为设计中某些方面的变更提供指导；降低由于软件构造不合理带来的风险

8.1 软件体系结构（续）

- 不同利益者相关者：交流的语言
- 可看作对诸如成本、可用性、可维护性以及性能等多个会对社会系统产生重大影响的属性进行权衡取舍之后做出的决策。
- 是记录过去体系结构解决方案等文档资料

体系结构决策描述模版

每个关键的体系结构决策都可以被记录，以便以后想要理解体系结构揭述的利益相关者进行评审。这里给出的是 Tyree 和 Ackerman [Tyr05] 提出模版的修改简化版本。

设计问题：描述将要解决的体系结构设计问题。

解决方案：陈述所选择的解决设计问题的方法。

分类：指定设计问题和解决方案陈述的设计类别（例如，数据设计、内容结构、构件结构、集成、简要说明）

假设：指出任何有助于制定决策的假设。

约束：指定任何有助于制定决策的环境约束（例如，技术标准、可用模式、项目相关问题）。

候选方案：简要描述考虑过的体系结构设计方案，并描述为什么放弃这些方案。

决策理由：陈述选择该决策而不是其他决策的理由。

决策结果：指出做出该决策的影响这个决策方案会对其他体系结构设计问题产生什么影响？这个决策方案是否会对设计产生约束？

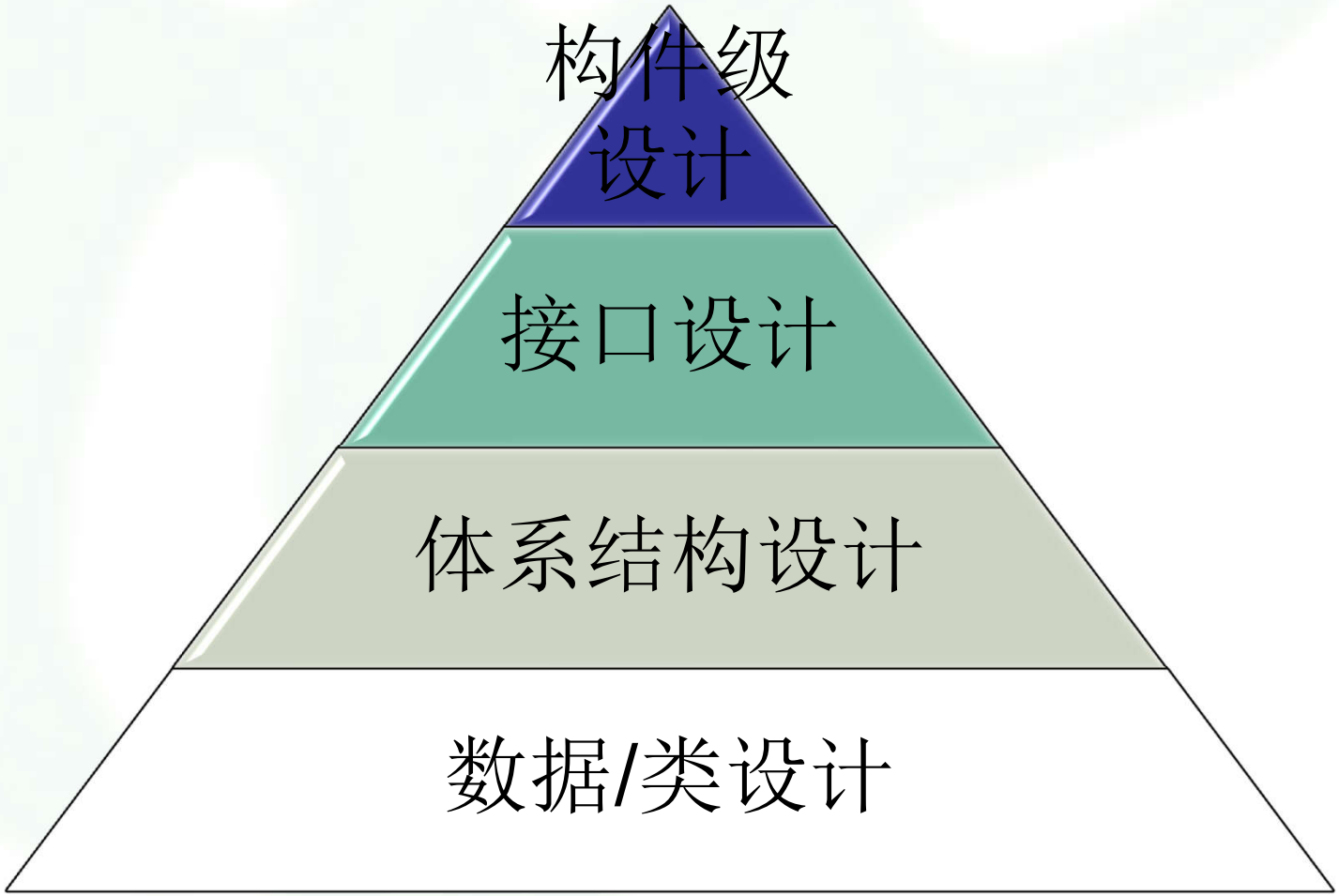
相关决策：其他哪些已记录的决策与此决策相关？

相关关注点：其他哪些需求与该决策相关？

工作产品：指出该决策会在何处对整个体系结构描述产生影响。

注释：参考可用来制定决策的其他团队的备忘录或文档。

设计模型



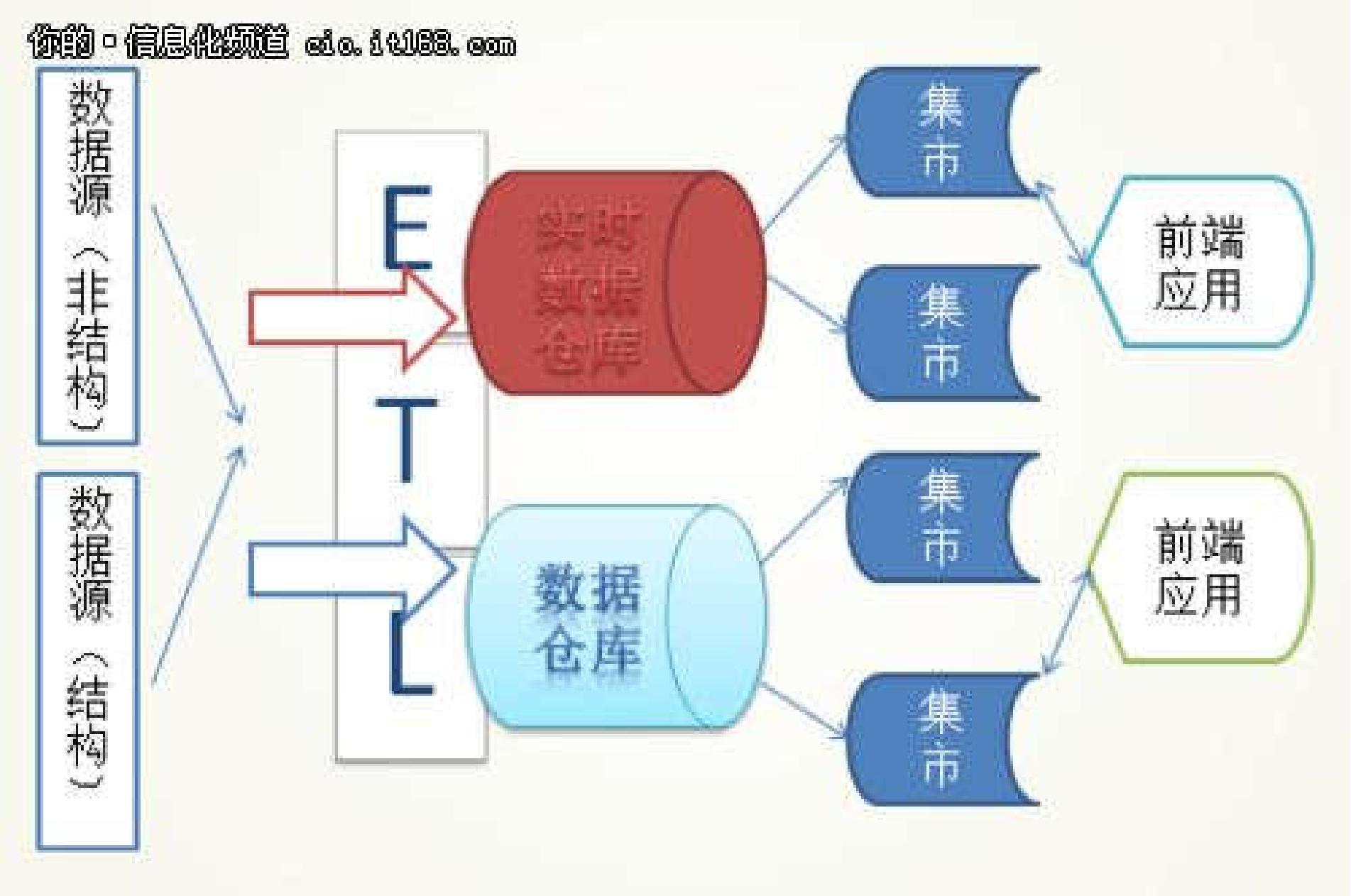
8.2 数据设计

- 数据设计：
  - 把分析模型中定义的数据对象转化成软件构件级的数据结构，或整个程序级的数据库结构。

8.2.1 体系结构级的数据设计

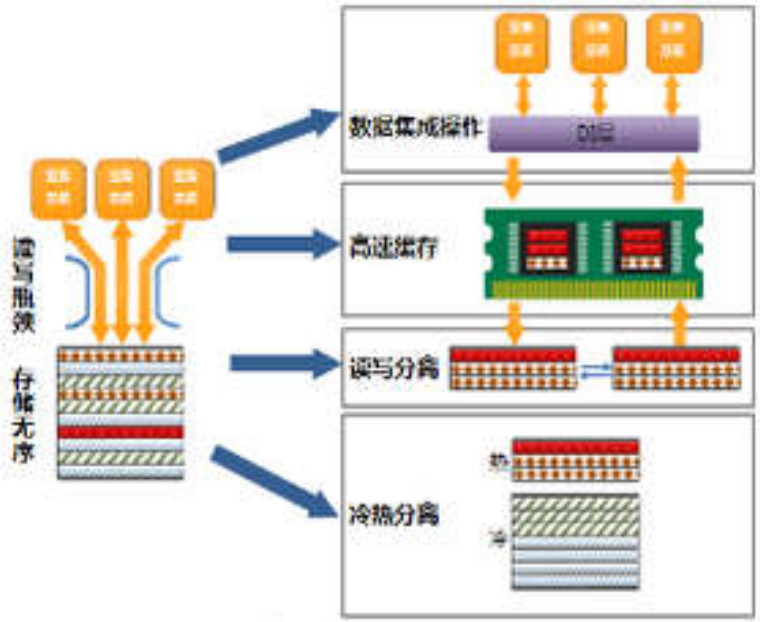
- 体系结构级（系统结构级，或程序级）的数据设计：
  - 数据挖掘技术
    - 数据库中的知识发现
  - 数据仓库
    - 庞大的、独立的数据库，访问存于数据库中的数据，为某项业务所需要的一组应用系统服务。

数据架构举例


你的·信息化频道 cio.it168.com
数据源（非结构）
数据源（结构）
E
T
L
数据仓库
数据仓库
集市
集市
集市
集市
前端应用
前端应用

数据架构 - 高效数据操作

农银人寿 ABC LIFE

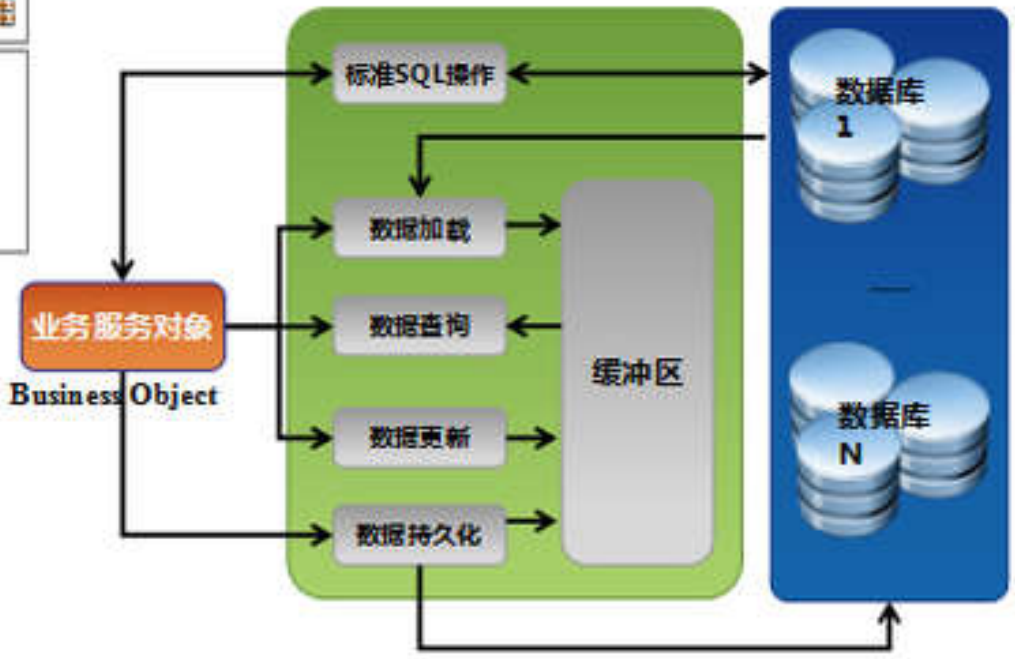


数据集成层（DI）
Powered by DEVELA

- ◆ 高效、透明、可独立部署
- ◆ 降低开发难度，提升开发效率
- ◆ 解决重复查询导致的性能问题

高速缓存
Powered by CACHE

- ◆ 存放访问频率很高、但数据量较少的数据。如：码表、用户机构权限、费率相关。



* 使用表分区、拆表、拆库 >> 数据分片（Oracle Data Sharding）

8.2.2 构件级的数据设计

- 构件级的数据设计原则：
  - 应用于功能和行为的系统分析原则也可用于数据。
  - 标识所有数据结构及其完成的操作。
  - 应该建立定义数据对象内容的机制，并且用于定义数据及其操作
  - 低层的数据设计应放到设计过程的后期—逐步求精
  - 数据的隐蔽性—只有使用的模块才可以看到数据结构的内部表示
  - 开发由有用的数据结构及其操作组成的库。--类库
  - 选用合适的编程工具以支持抽象数据类型的规格说明和实现。

8.3 体系结构风格和模式

- 建筑风格指导了建筑师的工作
- 体系结构风格：
  - 对完成同一种或同一类工作，不同的设计人员在体系结构设计的方式各不一样，这种方式的一定程度上的抽象
- 体系结构模式：
  - 就是风格的具体体现，或者体系结构设计的一个框架。
  - 定义了处理系统某些行为特征的方法

8.3.1 体系结构风格的分类

- 每个体系结构风格都描述了一种系统类别，包括：

(1) 一组执行系统所需功能的构件（例如，数据库、计算模块）；

(2) 一组实现构件间“通信、合作和协调”的连接件；

(3) 定义构件如何集成为系统的约束；

(4) 能够使设计者通过分析系统组成元素的已知属性来理解系统整体性质的语义模型。

体系结构图风格参考：https://zhuanlan.zhihu.com/p/644132764?utm_id=0

8.3.1 体系结构风格的分类

1. 以数据为中心的体系结构

- 数据存储驻留在这种体系结构的中心，其他构件经常访问（增删改查）数据存储
- 提高了可集成性，便于现有构件修改和新构件加入

8.3.1 体系结构风格的分类

1. 以数据为中心的体系结构

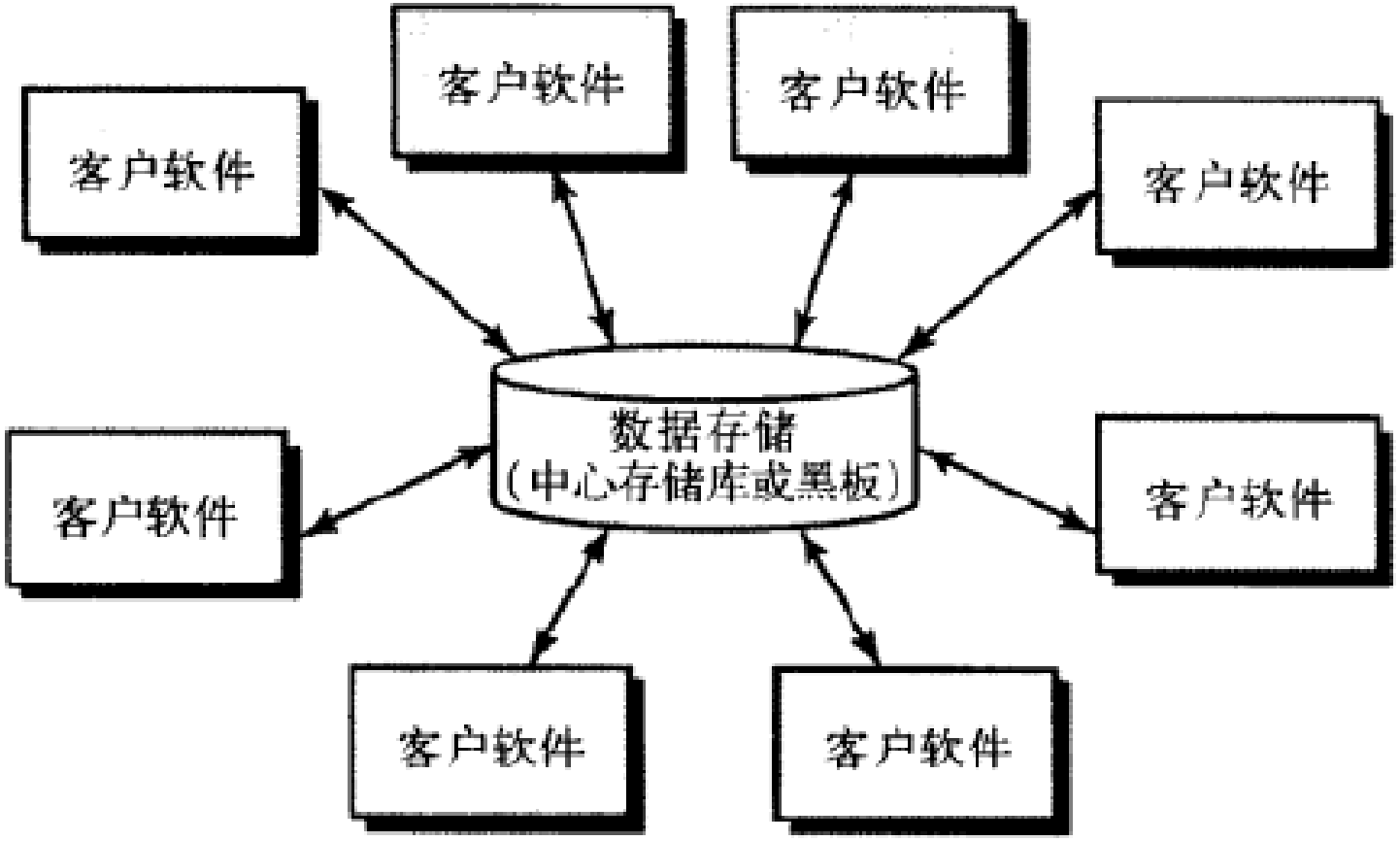


图　　以数据为中心的体系结构

8.3.1 体系结构风格的分类

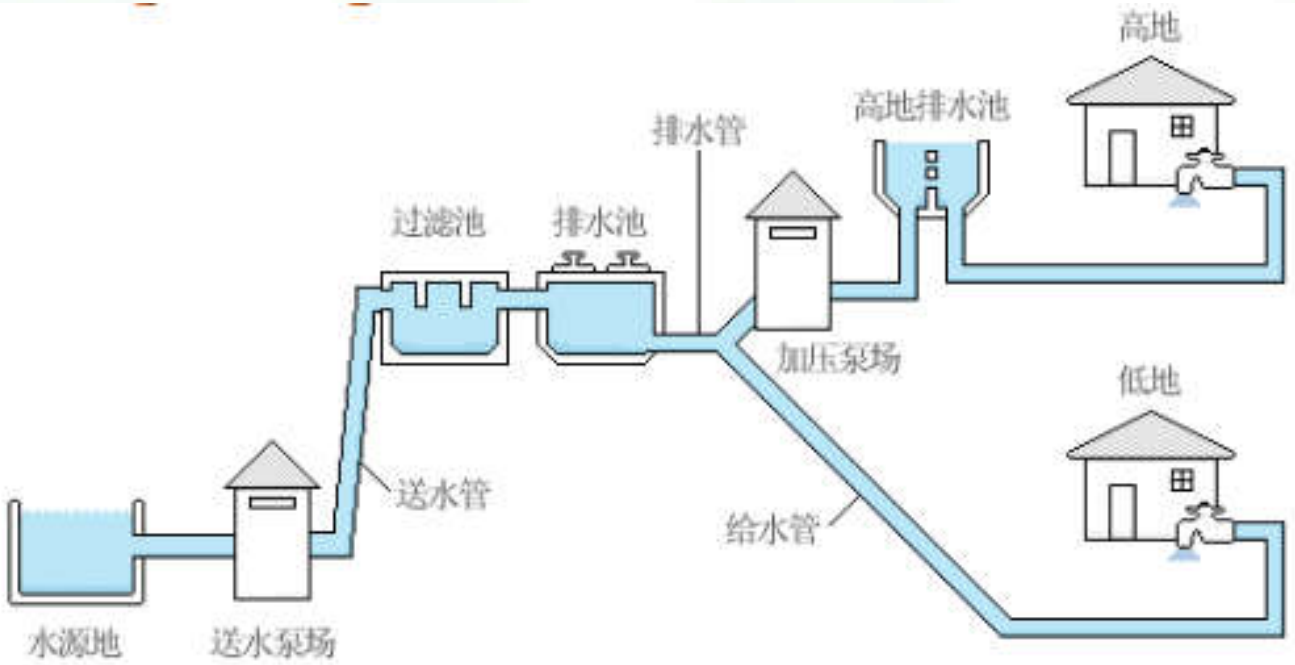
2. 数据流体系结构

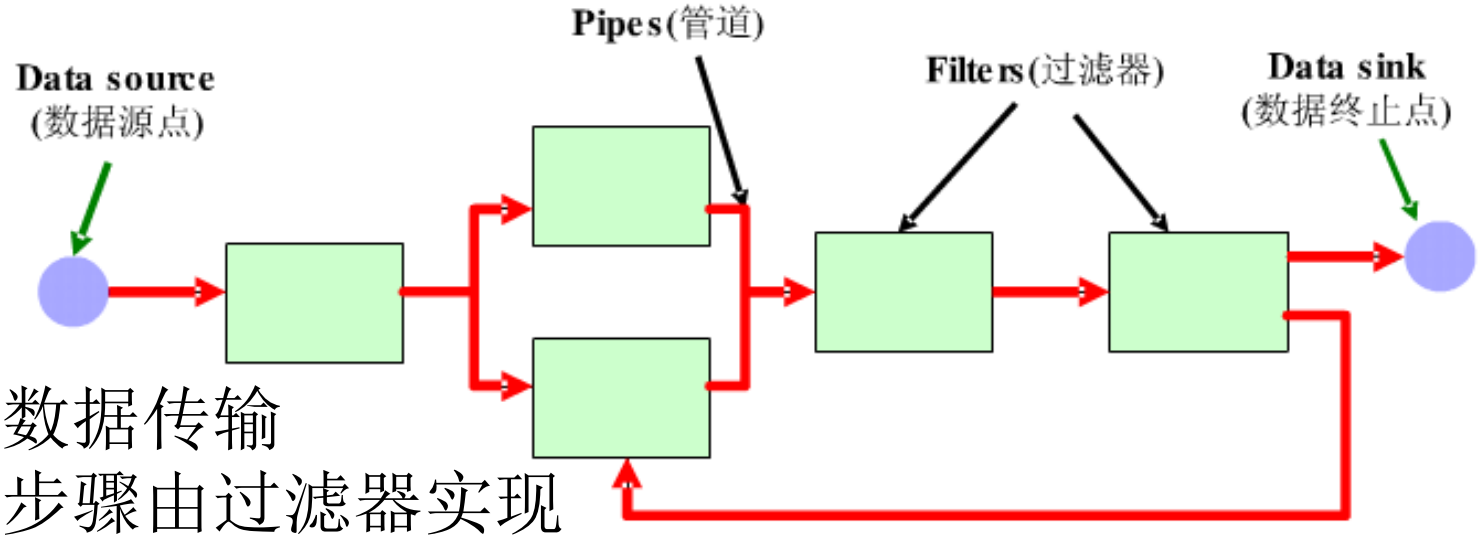
- 数据服从输入—变换---输出的简单流程
- 管道和过滤器结构拥有一组被称为过滤器的构件，每个构件独立于上游和下游而工作。

数据流体系结构

- 两种典型的数据流风格：
  - 管道-过滤器（Pipe-And-Filter）
  - 批处理（Batch Sequential)

从”自来水管道系统”看管道-过滤器风格





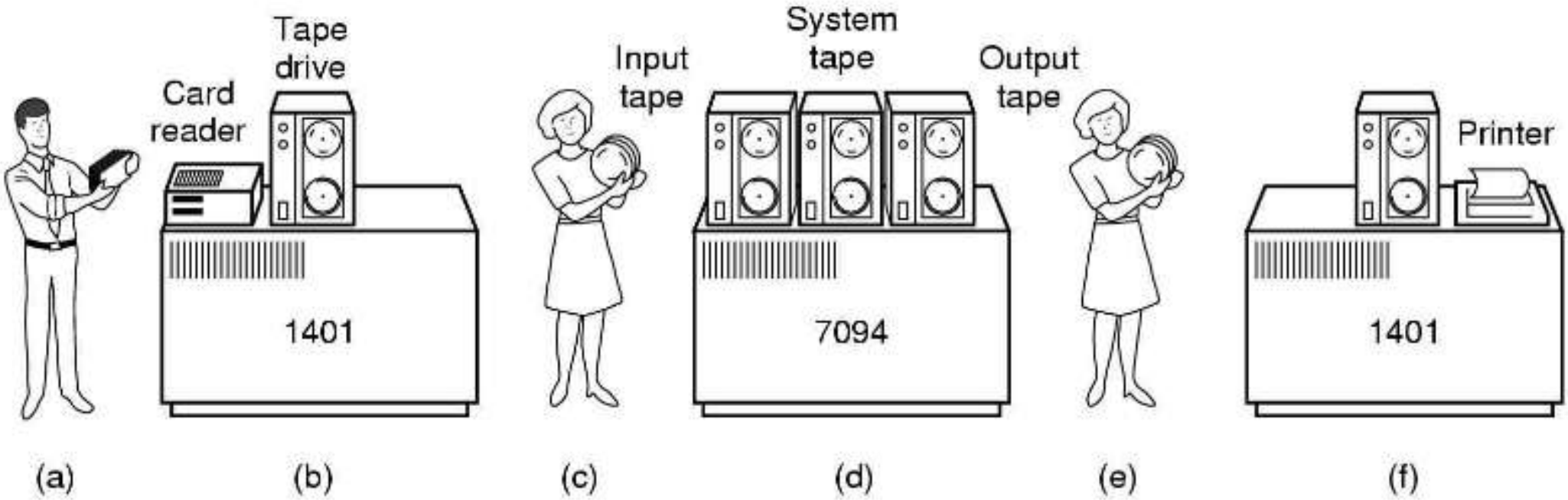
管道负责数据传输

每个处理步骤由过滤器实现

- 管道-过滤器风格优点：
  - 使得软构件具有良好的隐蔽性和高内聚、低耦合的特点；
  - 允许设计者将整个系统的输入/输出行为看成是多个过滤器的行为的简单合成；
  - 支持软件重用。只要提供适合在两个过滤器之间传递的数据，任何两个过滤器都可被连接起来；
  - 系统维护和增强系统性能简单。新的过滤器可以添加到现有系统中来；旧的可以被改进的过滤器替换掉；
  - 允许对一些如吞吐量、死锁等属性的分析；
  - 支持并行执行。每个过滤器是作为一个单独的任务完成，因此可与其他任务并行执行。

- 管道-过滤器风格缺点：
  - 通常导致进程成为批处理的结构。这是因为虽然过滤器可增量式地处理数据，但它们是独立的，所以设计者必须将每个过滤器看成一个完整的输入到输出的转换；
  - 不适合处理交互的应用。当需要增量地显示改变时，这个问题尤为严重；
  - 因为在数据传输上没有通用的标准，每个过滤器都增加了解析和合成数据的工作，这样就导致了系统性能下降，并增加了编写过滤器的复杂性。

批处理风格的直观结构



将用户输入的纸带上的数据写入磁带

将磁带作为计算设备的输入，进行计算，得到输出结果

打印计算结果

每一步必须在前一步结束后完成；每个处理步骤是一个独立的程序；数据必须是完整的，以整体的方式传递

8.3.1 体系结构风格的分类

3. 调用和返回体系结构

1. 主程序/子程序体系结构

- 主程序调用一组程序构件，这组程序构件又去调用其程序构件

2. 远程过程调用体系结构。

- 主程序/子程序的构件分布在多个计算机上。

8.3.1 体系结构风格的分类

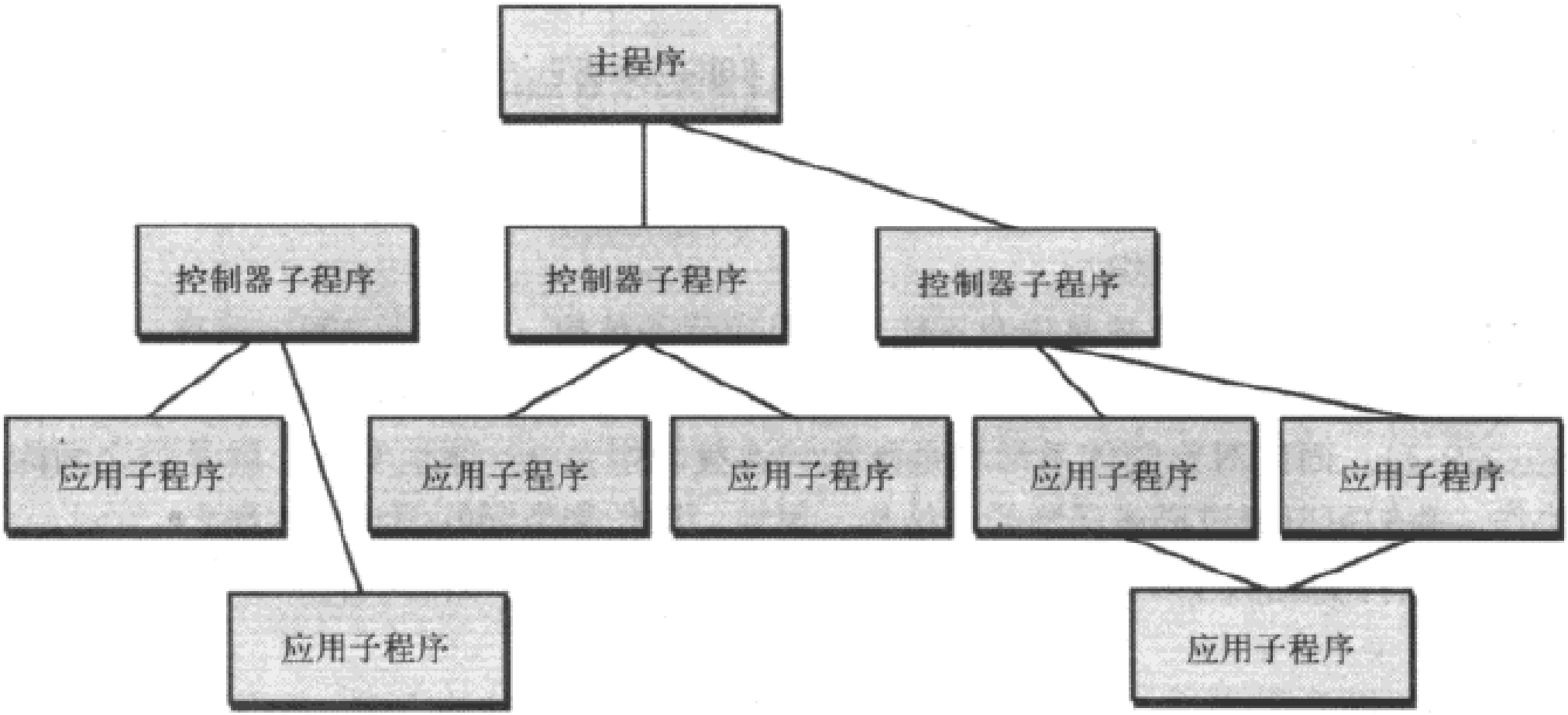
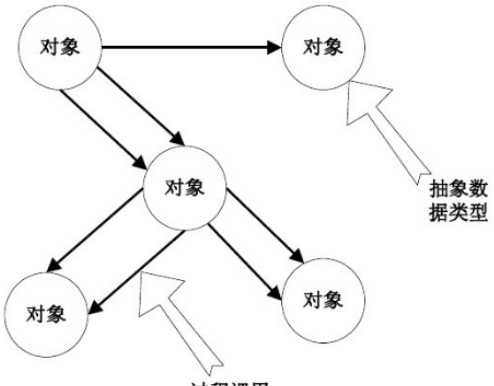


图　　主程序/子程序体系结构

8.3.1 体系结构风格的分类

4. 面向对象体系结构

- 封装了数据和必须应用到该数据上的操作，构件间通过信息传递进行通信和合作。



优点：对象隐藏；可以将数据存取操作分解成一些交互的代理程序的集合。
缺点：对象调用必须有对象的标识；对象循环调用等问题。

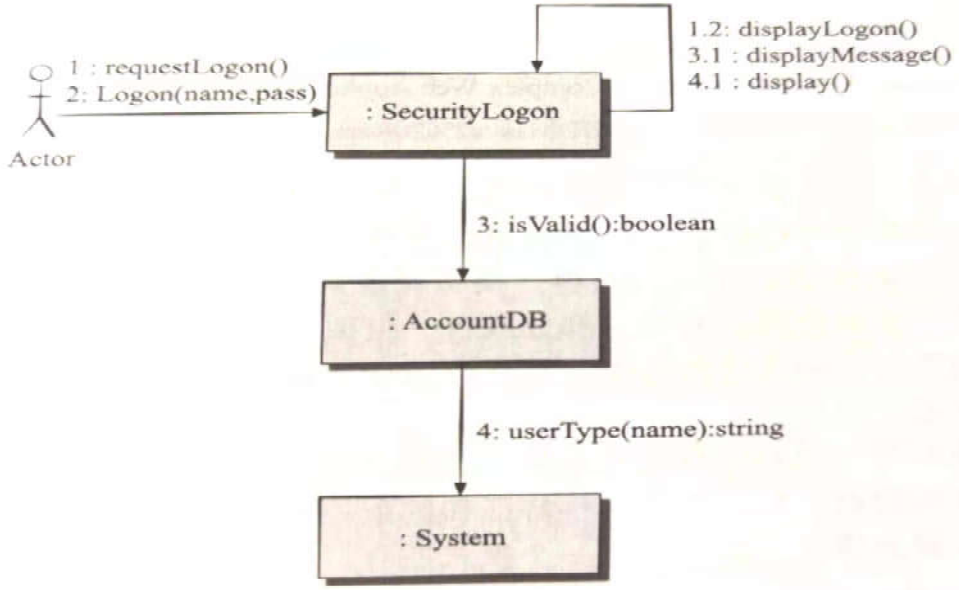


图 10-4 UML 通信图

登陆通讯图

8.3.1 体系结构风格的分类

5. 层次体系结构

- 各模块实现功能的层次不一样

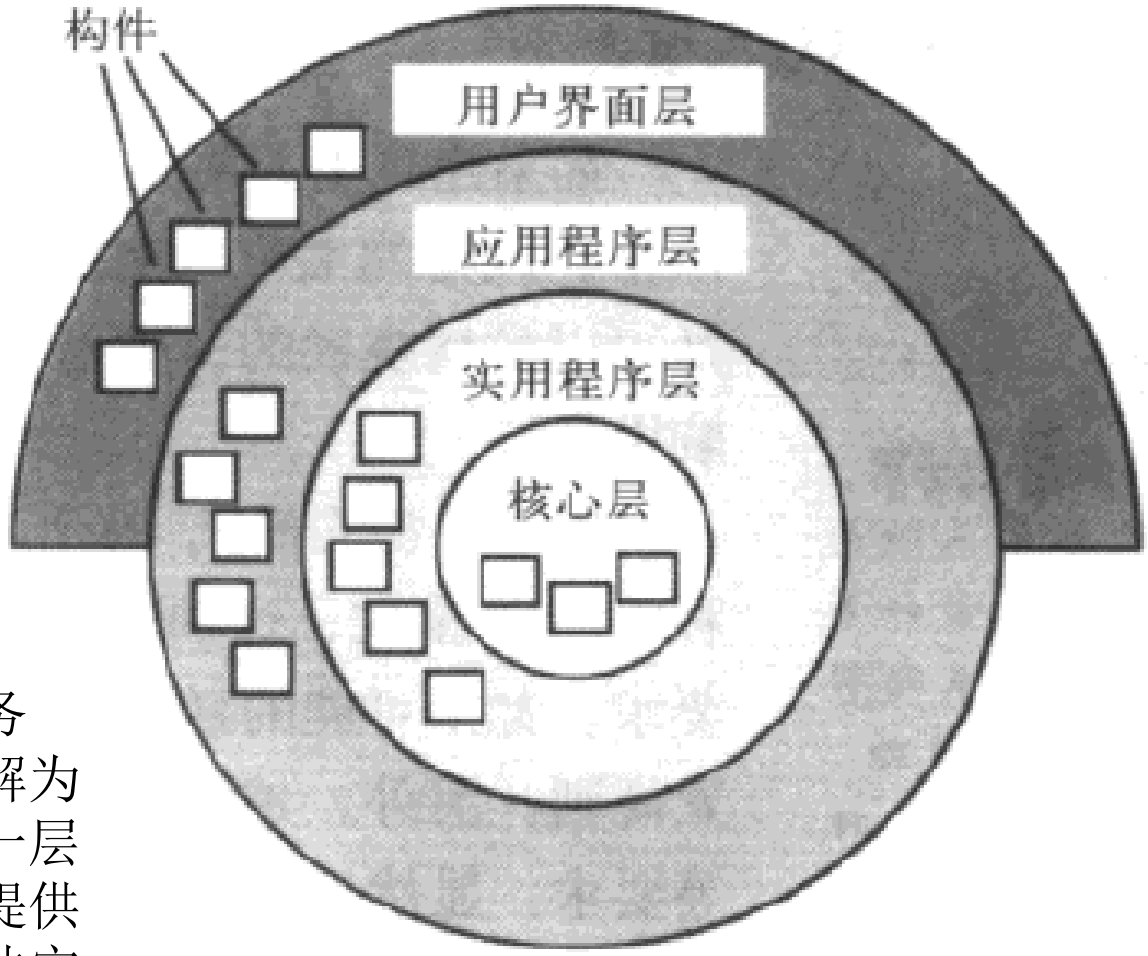


图　　层次体系结构

每一层为上层服务，并作为下层服务
这样的设计允许将一个复杂问题分解为一个增量步骤序列的实现。由于每一层最多只影响两层，同时只给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强调的支持。
缺点：很难找到一个合适的、正确的层次抽象方法。

模型-视图-控制器（Model-View-Controller, MVC)

模型：包含所有应用特定内容和处理逻辑；

视图：包含所有接口特定功能并能够显示终端用户所需的内容和操作逻辑；

控制器：管理对模型和视图的访问并协调它们之间的数据流。

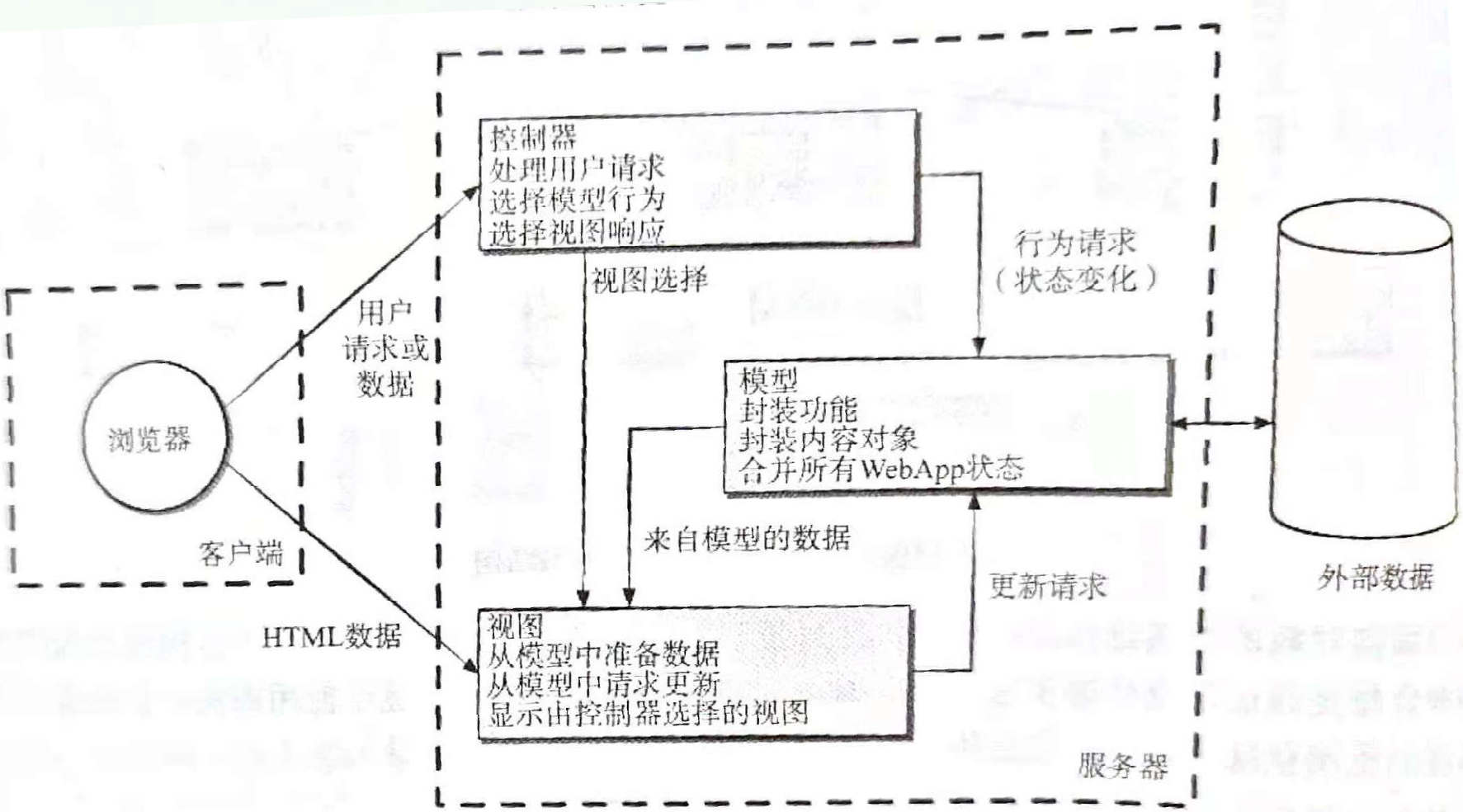
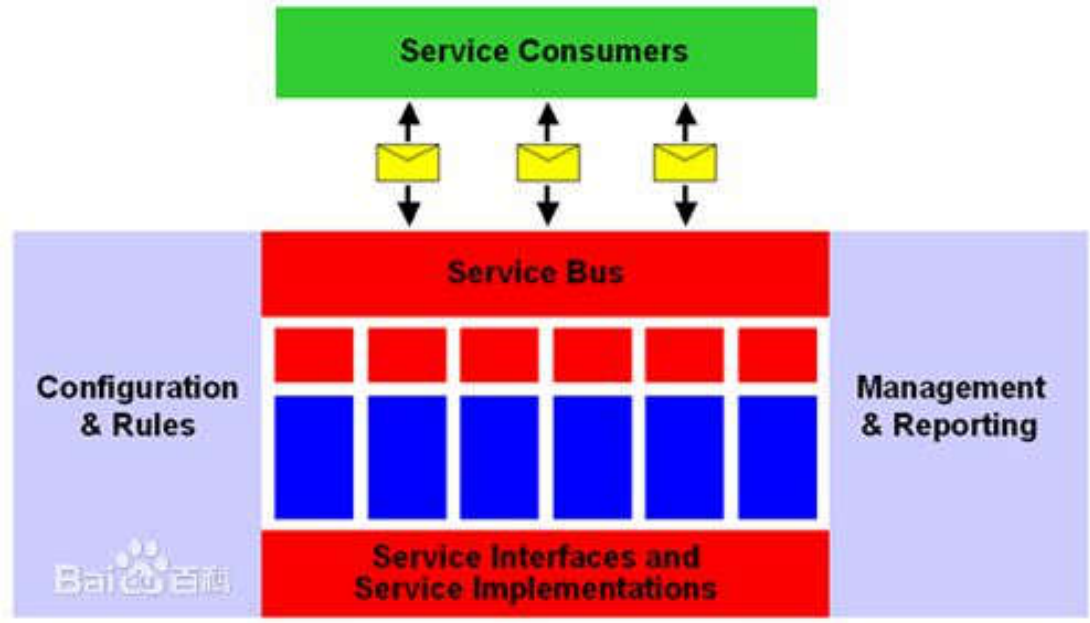


图 10-6 MVC 体系结构（来源：摘自 Jacyntho, Mark Douglas, Schwabe, Daniel and Rossi, Gustavo, "An Architecture for Structuring Complex Web Applications," 2002, available at http://www-di.inf.puc-rio.br/schwabe/papers/OOHDMJava2%20Report.pdf）

8.3.1 体系结构风格的分类

6.SOA

- 整个应用程序被设计和实现为一组相互交互的服务



8.3.2 体系结构模式

- 不同模式操作下的一些特征：
  - 并发性
    - “操作系统进程管理”模式
    - “任务调度器”模式

表 4.4 一些调度设计模式

模式	说明	优点	成本
循环执行	调度器会在重复循环中按相同的顺序运行一系列(每一个)任务	简单 公平 可预测性高	低响应 不稳定 性能次优 需要调整
时间触发的循环执行	和循环执行一样,除了每个循环是在特定的时刻开始之外	简单 公平 可预测性高 与参考时钟同步	低响应 不稳定 性能次优 需要调整
速率单调调度(RMS)	假定所有任务在一段时间结束时都是周期性的和有期限的。在设计时可以根据时间来指定优先级——时间越短,优先级越高。总是运行最高优先级的任务	稳定 优化 鲁棒	不公平 不能扩展到非常复杂的系统 比较复杂 难预测
最早截止期限优先(EDF)	在运行时根据与截止期限的逼近程度(即紧迫性)来指定优先级。总是运行最高优先级的等待任务	优化 鲁棒	不公平 缺乏经验的实现会导致抖动 不稳定 比较复杂 难预测

8.3.2 体系结构模式（续）

- 持久性
  - “数据库管理系统”模式
- 分布性
  - 分布性问题有两个元素（1）实体间连接方式；（2）实体间通信的特性。
  - 典型模式：代理模式

表 4.6　一些分布模式

模式	说　明	优　点	成　本
共享内存	使用多端口(通常是双端口)内存来共享全局数据和事件	• 可以有效地共享大数据集 • 运行时的性能开销低	• 需要特殊的硬件来管理同步 • 不能很好地适用于大量的互连
观察者	使用订阅/退订功能来装备服务器,以从客户机分离	• 保持了适当的客户机-服务器知识 • 良好的运行时性能 • 易于实施不同的通知策略	服务器有点复杂
代理	在不同的地址空间中实现了的观察者模式	• 可以将通信方式的详细信息与应用程序的语义分离 • 可以最大程度地减少网络流量	代理本身可能很复杂
端口代理	将所有通信媒体的详细信息封装到互联的对象(端口)中来管理消息的编组、传输和解组	• 可以将应用程序的语义与通信语义分离 • 在不改变应用程序软件的情况下可以更改网络协议 • 有助于可移植性和可重用性	• 需要编写多组代理 • 端口代理可能很复杂
数据总线	可以将数据虚拟化为一个公共存储库“总线”,以加入分布以及分离客户机和服务器。有“推”和“拉”两个变种	• 可以将数据定位到一个单一的位置 • 可以很好地管理大数据集 • 可以很好地扩展到多个客户机 • 可以简化连接的拓扑结构	• 对单点故障脆弱 • 数据总线本身可能很复杂
代理程序	可以提供一个客户机和服务器的存储库,使连接变得灵活和鲁棒	• 易于实现容错系统 • 可以很好地支持对称多重处理	• 代理程序可能很复杂(虽然有可用的商业解决方案) • 代理程序通常需要很大的内存空间和显著的性能开销

8.3.3 组织和求精

- 对导出的体系结构设计进行评估的标准：
  - 控制：在体系结构中如何管理控制？是否存在一个不同的控制层次？如果是，构件在控制层次中担当什么角色？构件如何在系统中传递控制?构件间如何共享控制？控制拓扑结构是什么样？控件是否同步或构件是否异步操作?

8.3.3 组织和求精（续）

- 数据：构件间如何进行数据通信？数据流是否是连续的，或数据对象是否是零散地传递给系统？数据传递的模式是什么（也就是说，数据是从一个构件传递到另一个构件，还是系统中构件可以全局共享数据？）是否存在数据构件（如黑板或中心存储库）？如果存在，他们的角色是什么？功能构件如何和数据构件交互？数据构件是被动的还是主动的（即数据构件使否主动地和系统中其他构件交互）？数据和控制如何在系统中交互？

8.4 体系结构设计

- 设计应该定义与软件交互的外部实体（其它系统、设备、人）和交互的特性。
- 需要信息在需求工程阶段获取。

8.4.1 系统的环境表示

- 系统环境图通过描述系统的出入信息流、用户界面和相关的支持处理等来对环境建模。
- 在体系结构层，软件架构师用体系结构环境图对软件与外部实体交互方式进行建模。

8.4.1 系统的环境表示（续）

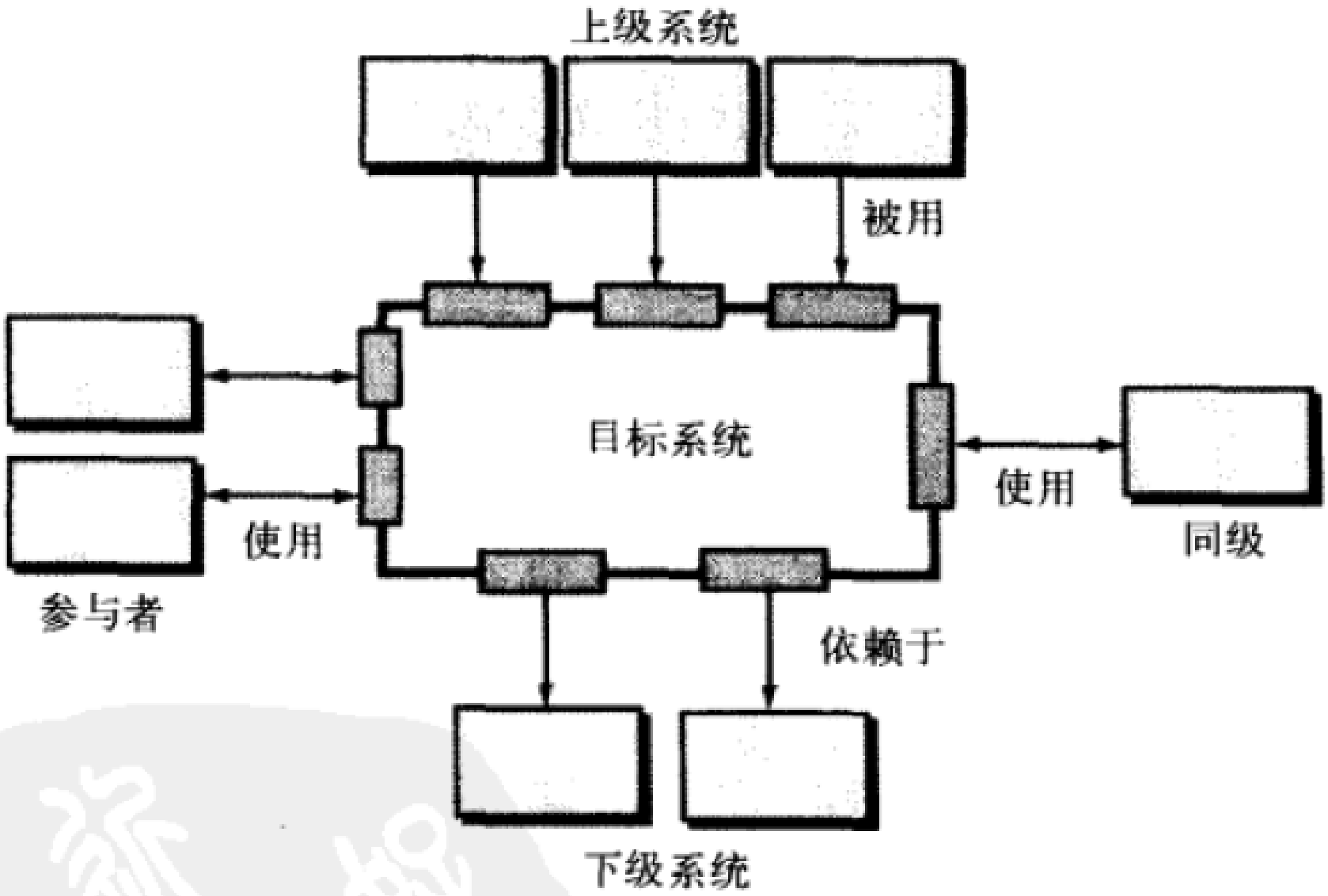


图　　体系结构环境图（摘自[BOS00]）

8.4.1 系统的环境表示（续）

与目标系统交互的系统可以表示为上级系统、下级系统、同级系统和参与者。

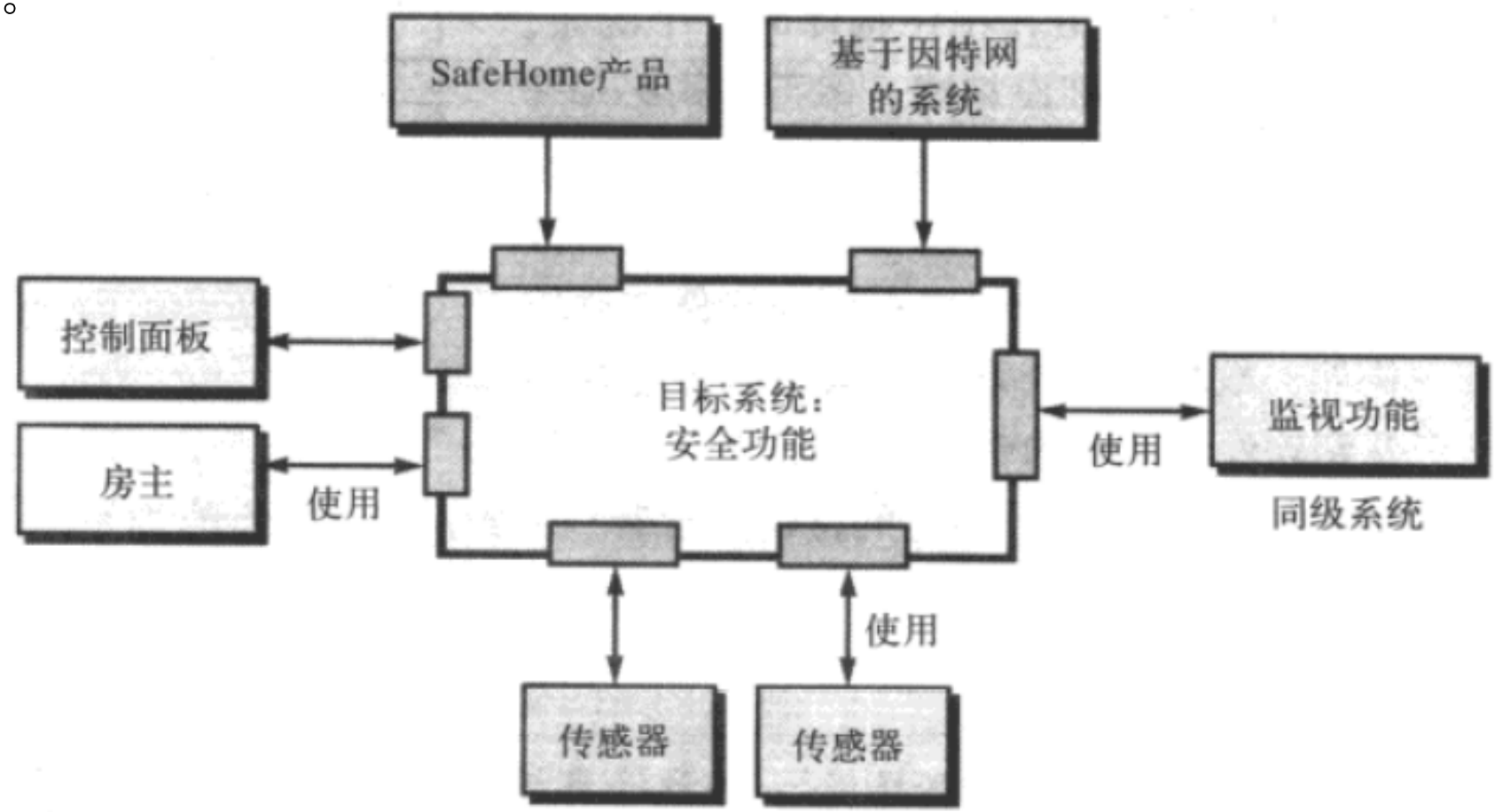


图　　SafeHome安全功能的体系结构环境图

8.4.2 定义原始模型

- 将目标系统用初略的功能框图表示出来
- 是表示核心抽象的类或者模式，该抽象对于目标系统体系结构的设计非常关键。
- 目标系统的体系结构由这些原始模型组成

8.4.2 定义原始模型（续）

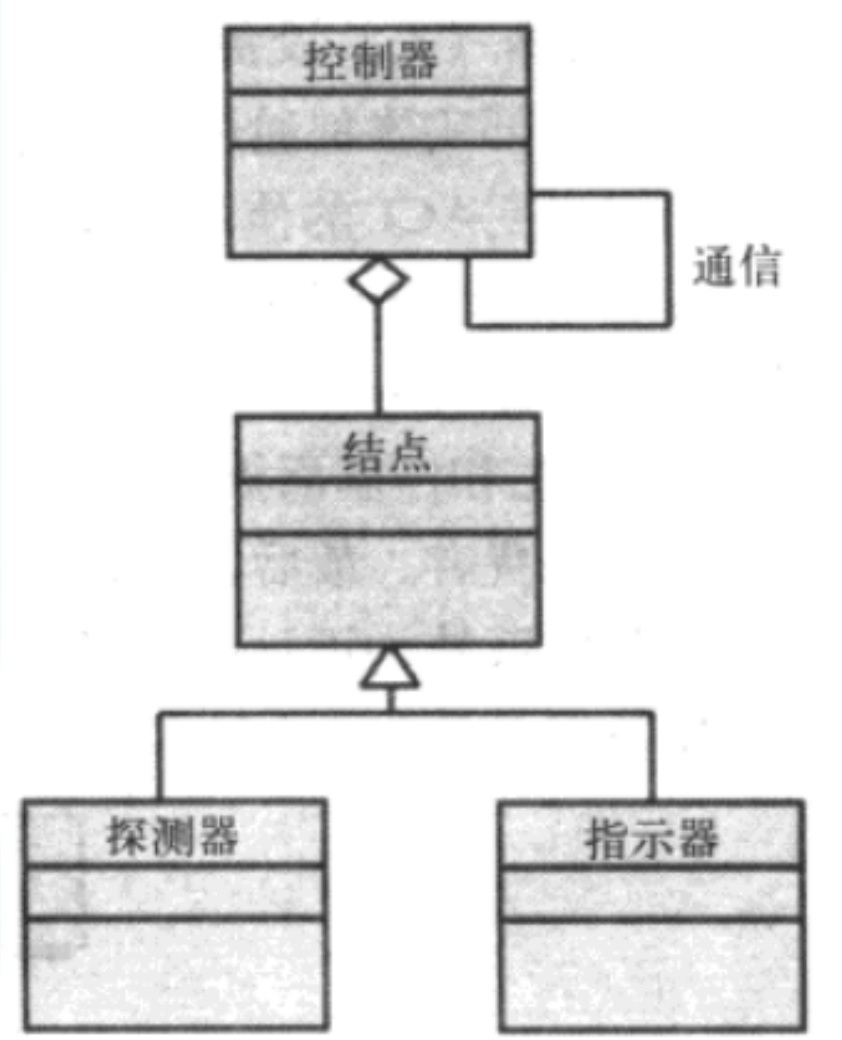


图　　SafeHome安全功能原始模型的UML关系图（摘自[BOS00]）

8.4.3 将体系结构精化为构件

- 导出和精化构件：
  - 应用领域是导出和精化构件的一个源泉。
    - 体系结构从分析模型中所描述的类开始
  - 另一源泉：基础设施域
    - 与应用领域没有业务联系，但体系结构必须容纳很多基础设施构件使应用构件能够运作，如内存管理构件、通信构件、数据库构件……

8.4.3 将体系结构精化为构件（续）

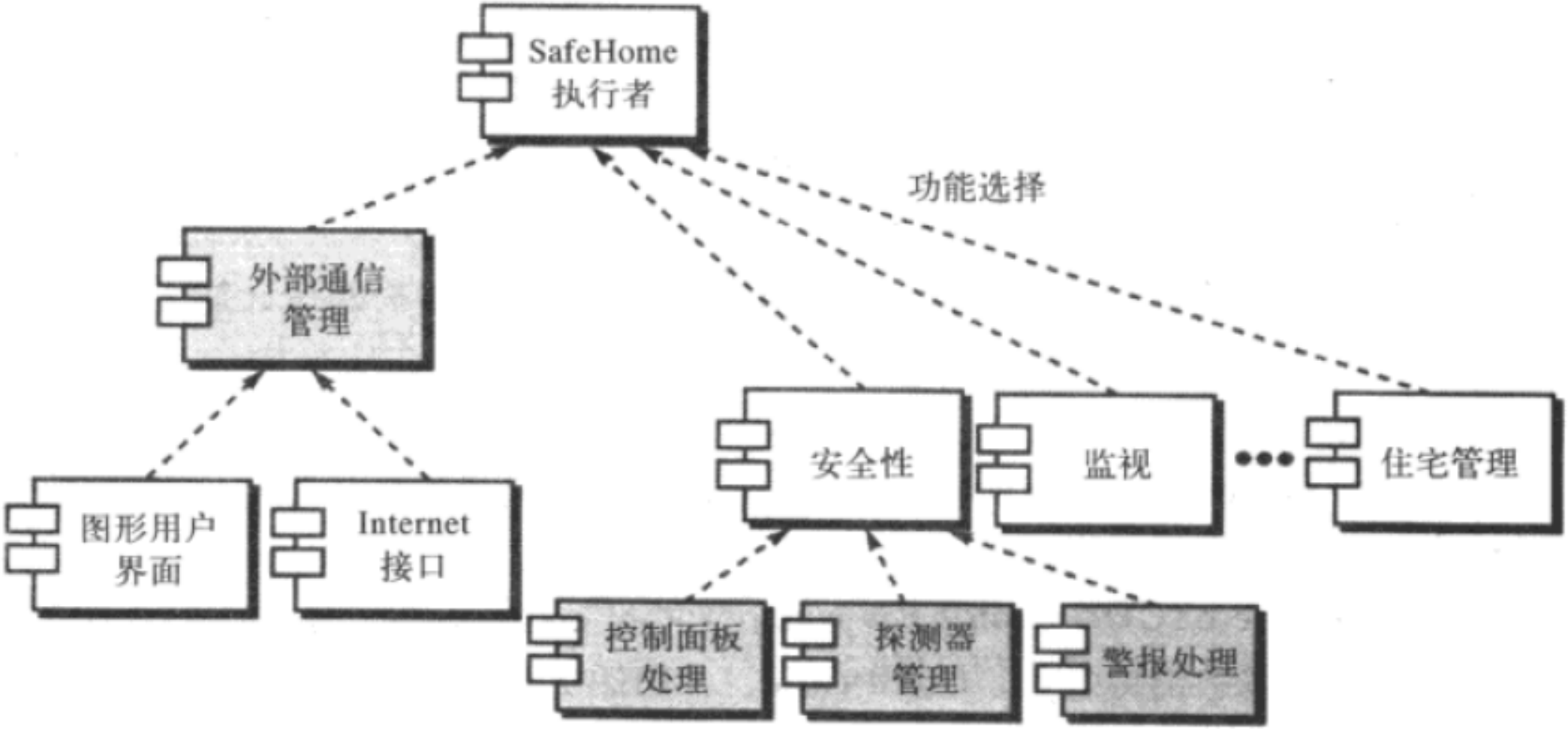


图　　带有高层构件的SafeHome整体体系结构

8.4.4 描述系统实例

- 结合一个具体的实例进一步精化模型，目的是证明结构和构件都是合理的。

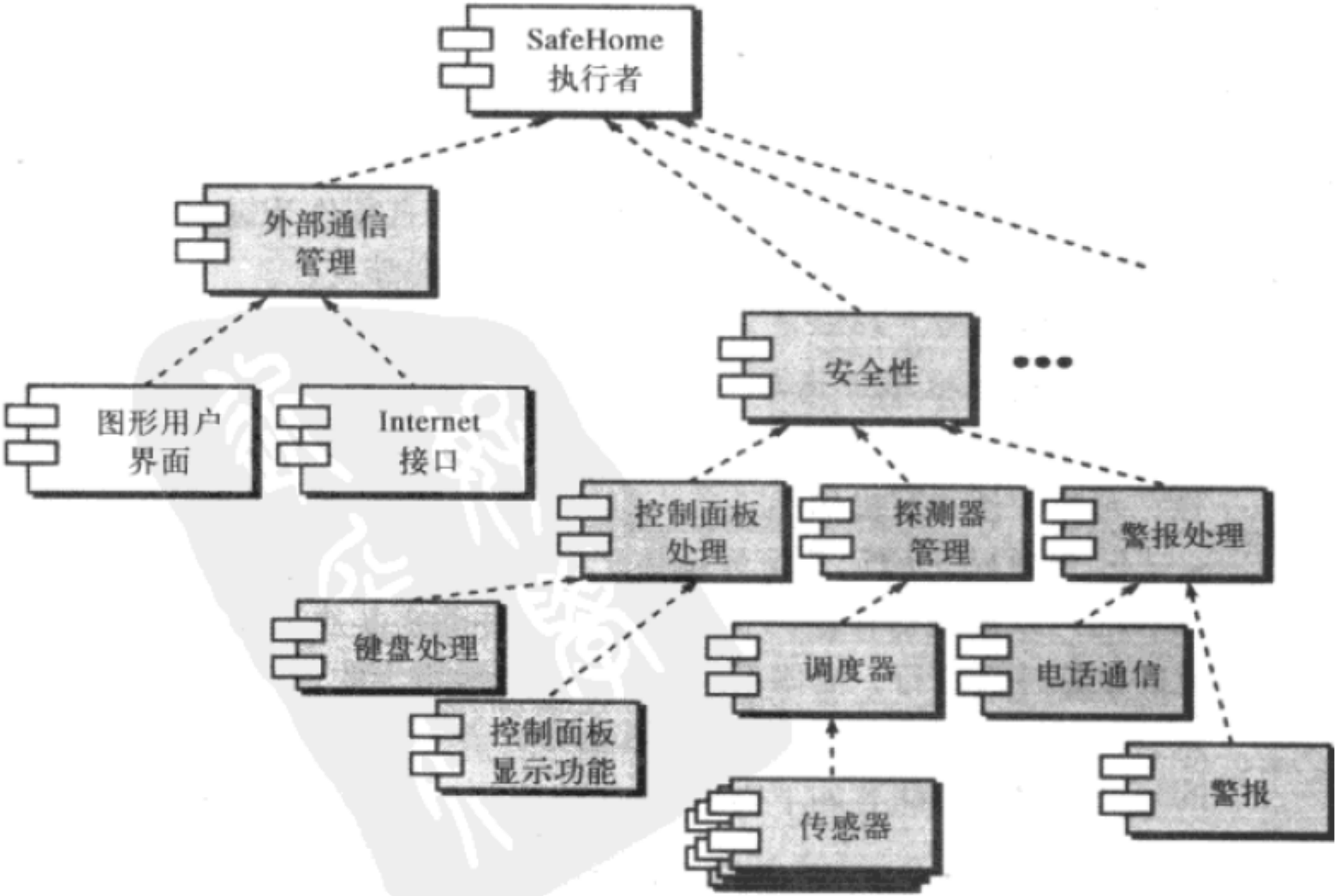


图　　带有构件详尽细节的安全功能实例

8.5 评估可选的体系结构设计

- 设计可能导致多种可选的体系结构
- 每种体系结构均需要评估，以确定哪种体系结构最适合要解决的问题

8.5.1 体系结构权衡分析方法

- 下面的设计分析活动是迭代进行的：

1. 收集场景。开发一系列用例
2. 诱导需求、约束和环境描述。确保所有共利益者的关注点
3. 描述那些已经被选用于解决场景和需求的体系结构风格/模式

8.5.1 体系结构权衡分析方法（续）

4. 通过孤立地考虑每个属性来评估质量属性。包括：可靠性、性能、安全性、可维护性、灵活性、可测试性、可移植性、可复用性和可操作性。
5. 针对特定的体系结构风格，弄清质量属性对每种体系结构属性的敏感性。
6. 使用在第5补中进行的敏感性分析鉴定候选体系结构。

8.5.1 体系结构权衡分析方法（续）

- SafeHome体系结构选择？
  - 调用和返回风格？　　面向对象？
  - 已有完整的用例集合，将每个用例用于这两个体系结构中，查看系统的反应，即构件和连接器是如何工作的。
  - 考虑变化场景、质量场景，并将其运行到体系结构中
  - 能更好适合用例和场景的体系结构就是最佳的选择。

8.5.2 体系结构复杂性

- 评估体系结构复杂性的一种有用技术：考虑体系结构中构件间的由系统中的信息/控制流驱动的依赖关系。建议三种类型的依赖：
  - 共享依赖：使用相同资源的消费者或为相同消费者生产的生产者之间的依赖关系。
  - 流依赖：表示资源的生产者和消费者的依赖关系。
  - 约束依赖：在一组活动间相关控制流上的约束，如构件互斥。

8.5.3 体系结构描述语言

- 体系结构描述语言（architectural description language, ADL）为描述软件体系结构提供一套语义和语法。
  - Rapide　UniCon　Aesop　Wright　Acme　UML

8.5.4 精化体系结构设计

- 结构上的简单往往反映出程序的优雅和高效。
- 设计求精应在满足模块化要求的前提下尽量减少构件的数量，在满足信息需求的前提下尽量减少复杂的数据结构。

8.6 敏捷性与体系结构

- 敏捷团队可以通过建立一个体系结构原型（如一个可以运行的系统骨架）并开发清晰的体系结构工作产品，与必要的利益相关者进行沟通，以满足体系结构设计的需要。
- 冲刺中，体系结构故事连通业务用户故事排序优先次序。
- 体系结构设计师要确保的是体系结构的重要部分已经经过斟酌，并且开发者也征求了利益相关者的意见。

概要设计说明书举例

- 概要设计说明书模版
- 概要设计说明书举例

作业

- 在软件体系结构讨论中，经常会遇到体系结构风格、体系结构模式及架构等术语。研究并描述这些术语之间的关系。
- 开发的作业项目的软件体系结构。