第13章　体系结构设计

1.为什么要使用软件体系结构?

- 任何复杂的产品或工程的产出，都需要结构设计，如建筑物。
- 软件体系结构并非可运行的软件，它是软件的一种“宏观”视图，使得软件工程师可以:

① 分析设计满足既定需求方面的有效性;

② 在设计变更相对容易的阶段，去考虑体系结构可能的选择方案;

③ 降低与软件构造相关的风险

软件体系结构的作用

- 软件体系结构的表示有助于对计算机系统开发感兴趣的所有利益相关者开展交流。
- 体系结构突出了早期的设计决策，这些决策对随后所有的软件工程工作有深远的影响。并且对于系统最终的成功，与运行实体同等重要。
- 体系结构“构建了一个相对小的、容易理解的模型，该模型描述了系统如何构成以及其构件如何一起工作”[BAS03]。
- 便于软件复用

软件体系结构的发展

- 经历了两个阶段：
  - 二十世纪八十年代左右，软件体系结构的描述随意性很大
  - 九十年代以来，软件体系结构已经正式作为软件开发中的一个设计活动，“架构设计师”也已成为了一个正式的技术职位，用于描述软件体系结构的语言和工具得到了迅速发展；出现了相应的不同体系结构标准

软件体系结构定义

- D. Garlan和M. Shaw：软件体系结构应包含对组成系统的组件的描述、组件之间的交互关系以及系统组合的模式和应该满足的约束等。
- D. E. Perry和A. L. Wolf：软件体系结构是由元素（element）、形式（form）和准则（rational）构成，其中，元素分成三类：处理元素（processing element）、数据元素（data element）和连接元素（connecting element），形式是由特性（property）和关系（relationship）组成。
- Bass、Clements和Kazman：软件体系结构是指系统的一个或多个结构，它包含软件构件、构件的外部可见属性以及它们之间的相互关系。

说明

- 软件构件可以是简单的程序模块或类，也可以是复杂的“中间件”，子系统。
- 关注构件的外部可见属性，不关心构件的具体实现细节。
- 构件之间的关系可以是简单的过程调用，也可能是复杂的通信协议。
- 体系结构和设计的关系与类和实例的关系类似，一个具体软件的体系结构设计可以看成是某种体系结构的一个实例。

体系结构描述

- IEEE计算机学会提出了IEEE-Std-1471-2000，软件密集型系统体系结构描述的推荐实践做法（《Recommended Practice for Architectural Description of Software-Intensive System》），[IEE00]
  - 建立软件体系结构设计过程中使用的概念性框架和词汇表，
  - 提供表示体系结构描述的详细准则，
  - 鼓励良好的体系结构设计实践。
- IEEE标准将体系结构描述(architectural description, AD) 定义为“一种记录体系结构的产品集合”
  - 描述本身展示了多个视图，且每个视图都是从“一组参与者关注点的角度观察的整个系统的一种表示”

体系结构决策

- 每个体系结构描述的视图都是从“一组参与者关注点的角度观察的整个系统的一种表示”，为了开发每个视图（包括整体的体系结构描述），架构师通常都要从多个可选的方案中选择最能满足关注点的特定的体系结构，该决策本身也可以作为体系结构的一种视图纳入体系结构描述中。

信息栏 体系结构决策描述模板

每个主要的体系结构决策都可以被记录在案，以便以后评审，评审由想要理解已提出的体系结构描述的利益相关者进行。这里给出的是 Tyree 和 Ackerman [Tyr05] 提出模板的修改和缩略版本。

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：指定问题和解决方案陈述的分类（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、可用的模式、项目相关问题）。

候选方案：简要描述所考虑的体系结构设计候选方案，并描述为什么要摈弃这些方案。

2.体系结构类型

- 类型（genre）隐含了在整个软件领域中的一个特定类别。
- 在每种类别中，会有很多的子类别。
  - 例如，在建筑物类型中，会有以下几种通用风格：住宅房、单元楼、公寓、办公楼、工厂厂房、仓库等。软件则可能有：人工智能、通信、金融、游戏、医疗、教育……
  - 在每一种通用风格中，也会运用更多的具体风格。每种风格有一个结构，可以用一组可预测模式进行描述。

3.体系结构风格

每种风格描述一种系统类别，包括：

① 完成系统需要的某种功能的**一组构件**（例如，数据库、计算模块）；

② 能使构件间实现“通信、合作和协调”的**一组连接件**；

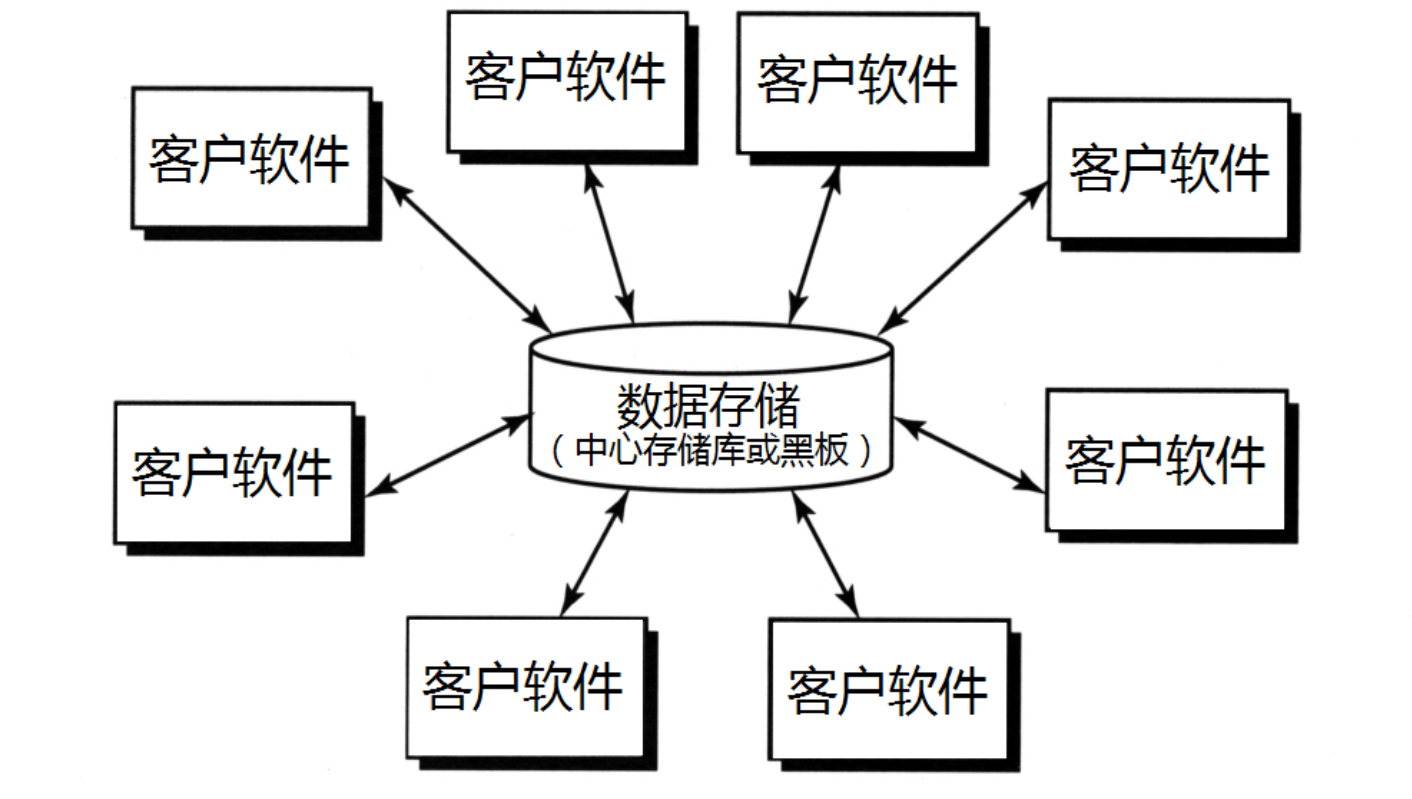
③ 定义构件如何集成为系统的**约束**；

④ **语义模型**，能使设计者通过分析系统组成成分的已知属性来理解系统的整体性质 。

3.1体系结构风格分类

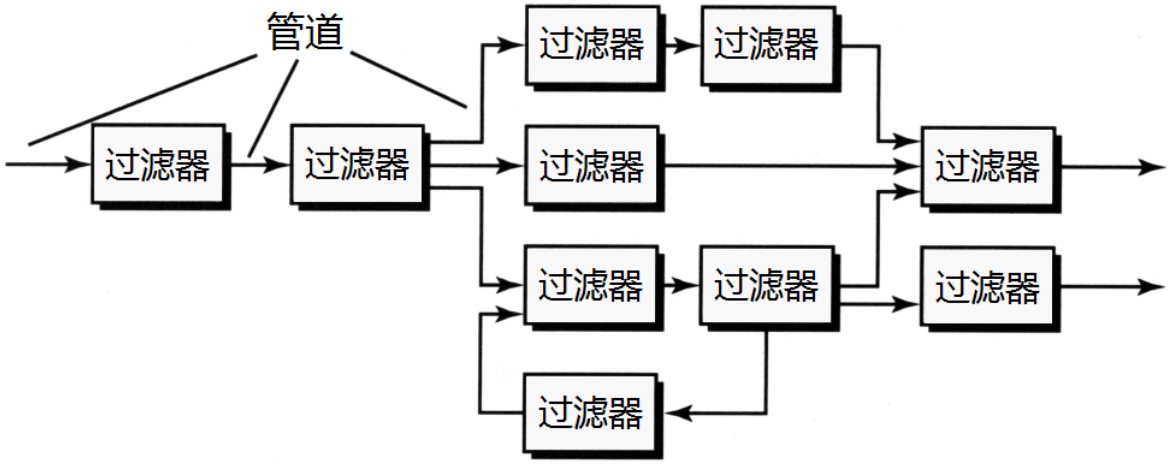
- 以数据为中心的体系结构
- 数据流体系结构
- 调用和返回体系结构
- 面向对象体系结构
- 层次体系结构
- ……

以数据为中心的体系结构

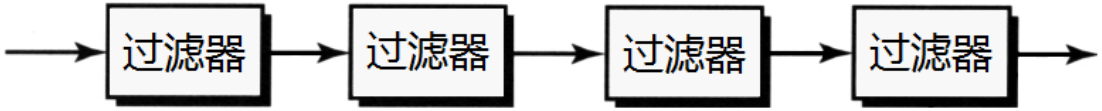


数据存储位于中心，其他构件访问该数据存储，对其中数据进行增删改查等操作。（黑板机制，客户软件可以相互独立）

数据流体系结构



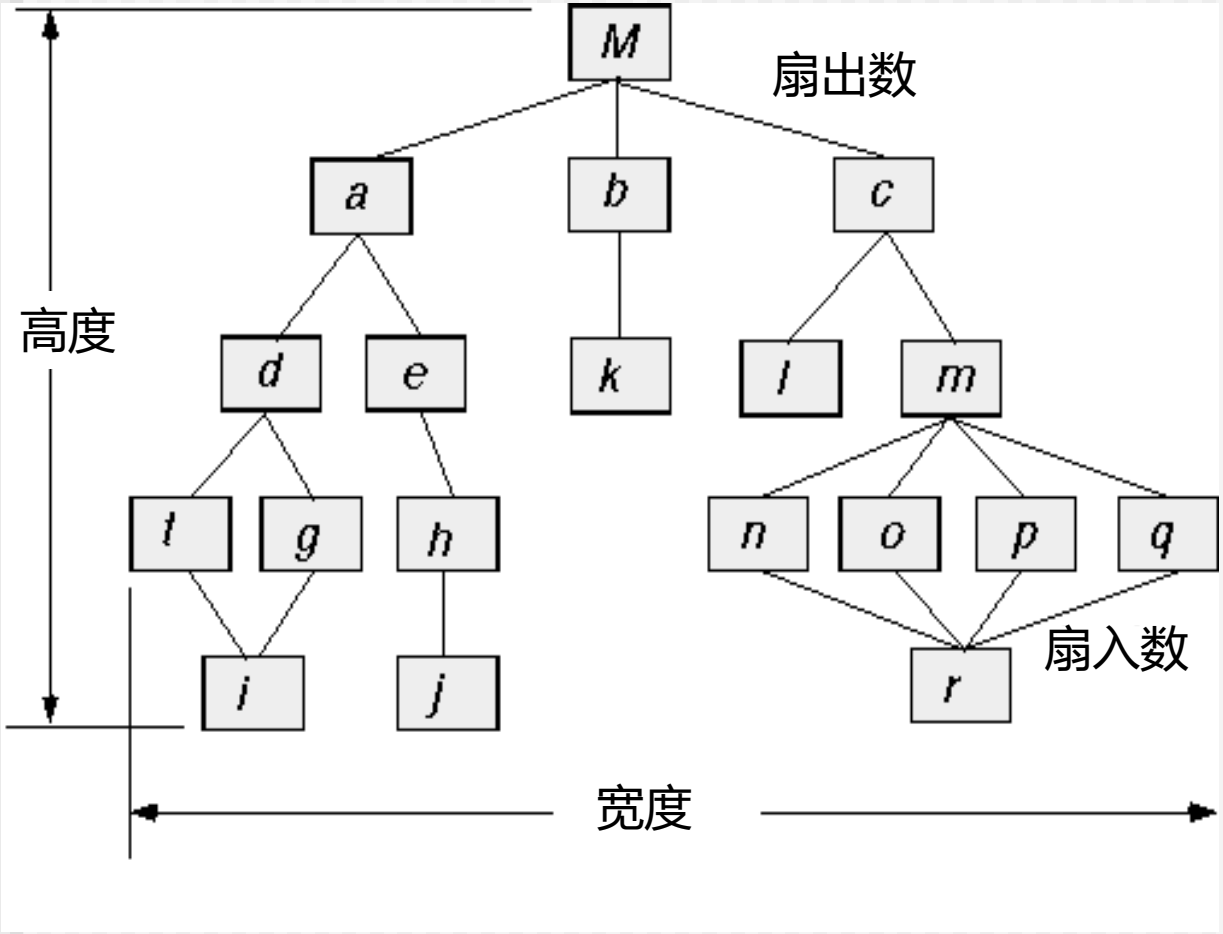
(a)管道和过滤器



(b)批序列

管道-过滤器模式，数据流过管道，在过滤器进行相应处理并流出

调用和返回体系结构



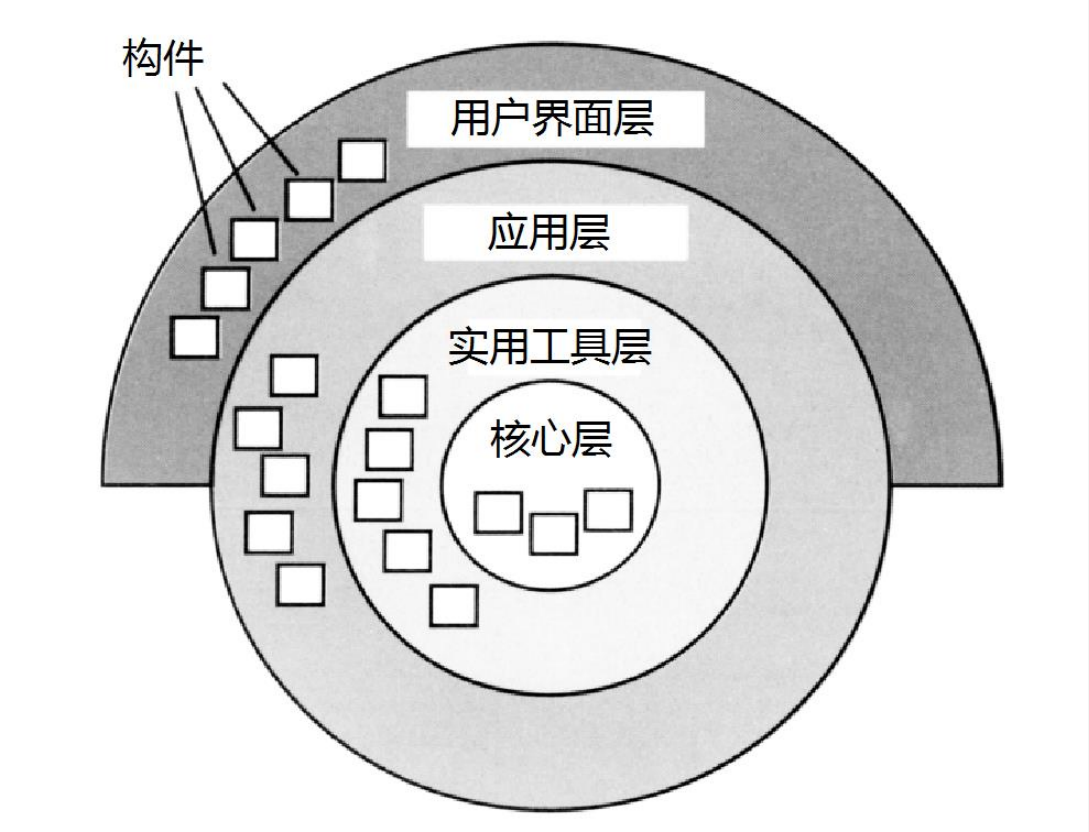
主程序/子程序体系结构、远程过程调用体系结构

面向对象体系结构

系统的构建封装了数据和必须用于控制该数据的操作，构件间通过消息传递进行通信和协作。

分布式对象（SOA）

层次体系结构



3.2体系结构模式

- 并发性——应用系统必须以一种模拟并行的方式来操作多个任务
  - 操作系统进程管理 模式
  - 任务调度器 模式
- 持久性——如果数据从创建它的进程执行以来一直存在，则该数据会持久存在。两种常见模式如下：
  - 数据库管理系统 模式将DBMS的存储和存取能力用于应用系统的体系结构中
  - 应用级持久性 模式在应用体系结构中建立了持久性特征
- 分布性——系统或系统中构件在一个分布的环境中相互通信的方式
  - 代理 在客户端构件和服务器构件之间充当“中间人”

4.体系结构考虑要素

- 经济性 —— 最好的软件应该是整洁的并依赖抽象化以减少无用的细节。
- 易见性 —— 对于那些随后将验证这些模型的软件工程师而言，体系结构的决策及其依据应该是显而易见的。
- 隔离性 —— 不产生隐藏依赖的关注点分离
- 对称性 —— 体系结构的对称性意味着它的属性是均衡一致的
- 应急性 —— 紧急的、自组织的行为和控制

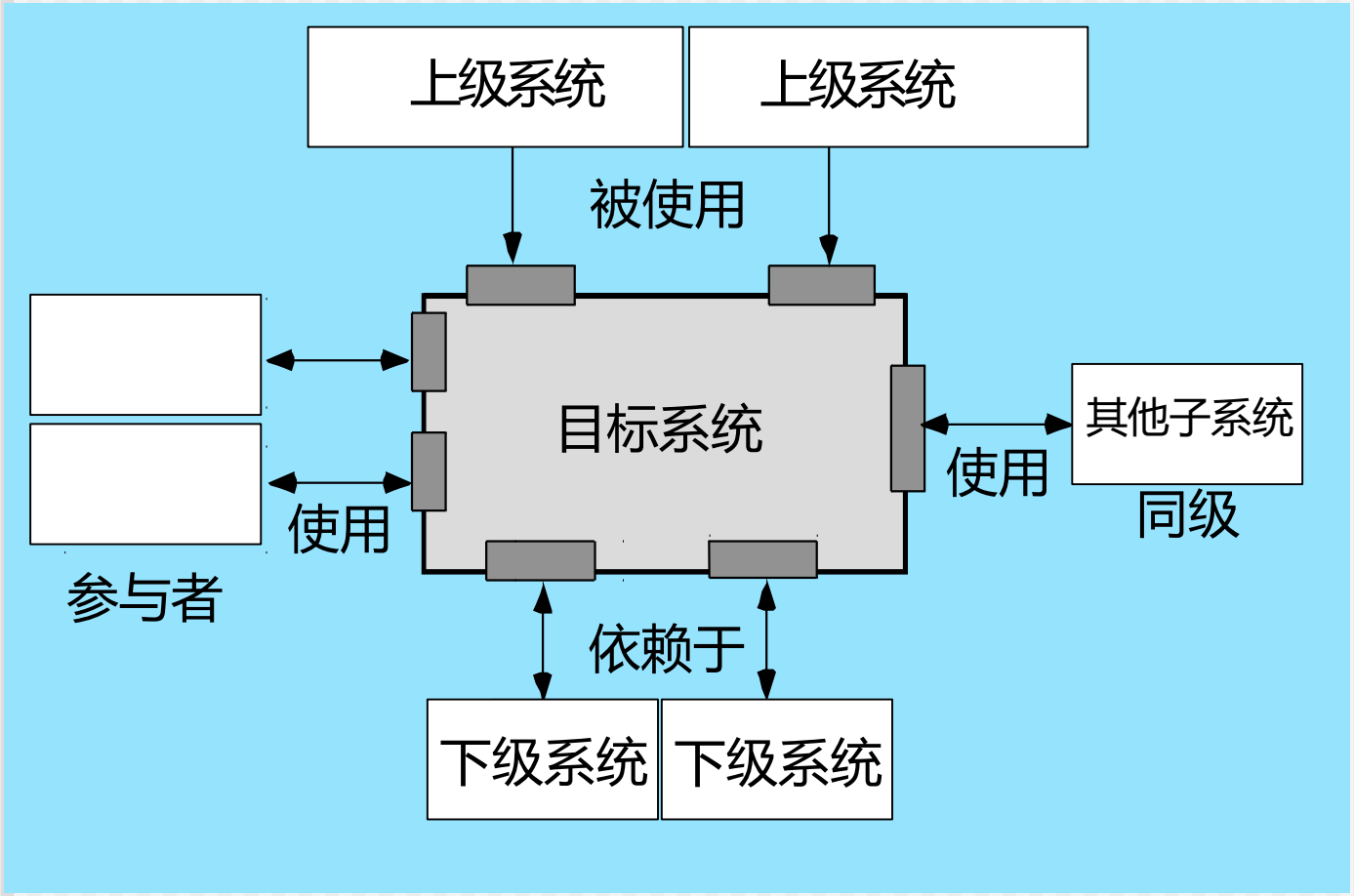
5.体系结构决策

- 记录了关键的设计问题以及所选用的体系结构方案背后的原理。
- 决策通常包括：
  - 软件系统组织
  - 结构元素的选取
  - 结构元素之间协作意向所定义的接口
  - 组合而成的子系统
  - 体系结构模式、应用技术、中间件、编程语言的选择等
- 决策模型，作为体系结构描述的一部分

6.体系结构设计

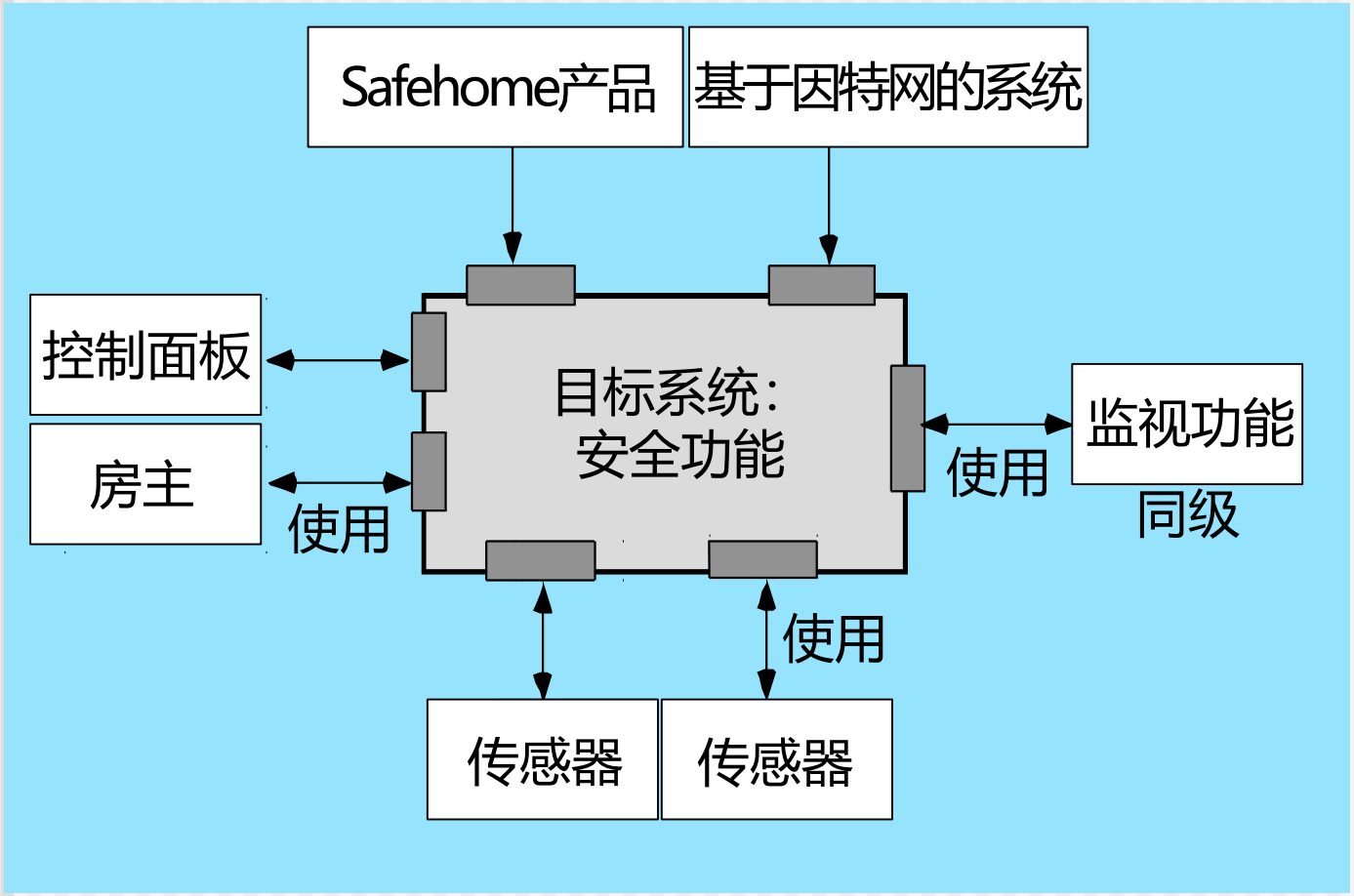
- 首先建立软件的环境模型，软件必须放在所处环境进行开发
  - 设计应该定义与软件交互的外部实体（其他系统、设备、人）和交互的特性（即接口）
- 确定一系列体系结构原型集
  - 原型 是表示系统行为元素的一种抽象（类似于类）
- 设计师通过定义和细化实施每个原型的软件构件来指定系统的结构

体系结构环境



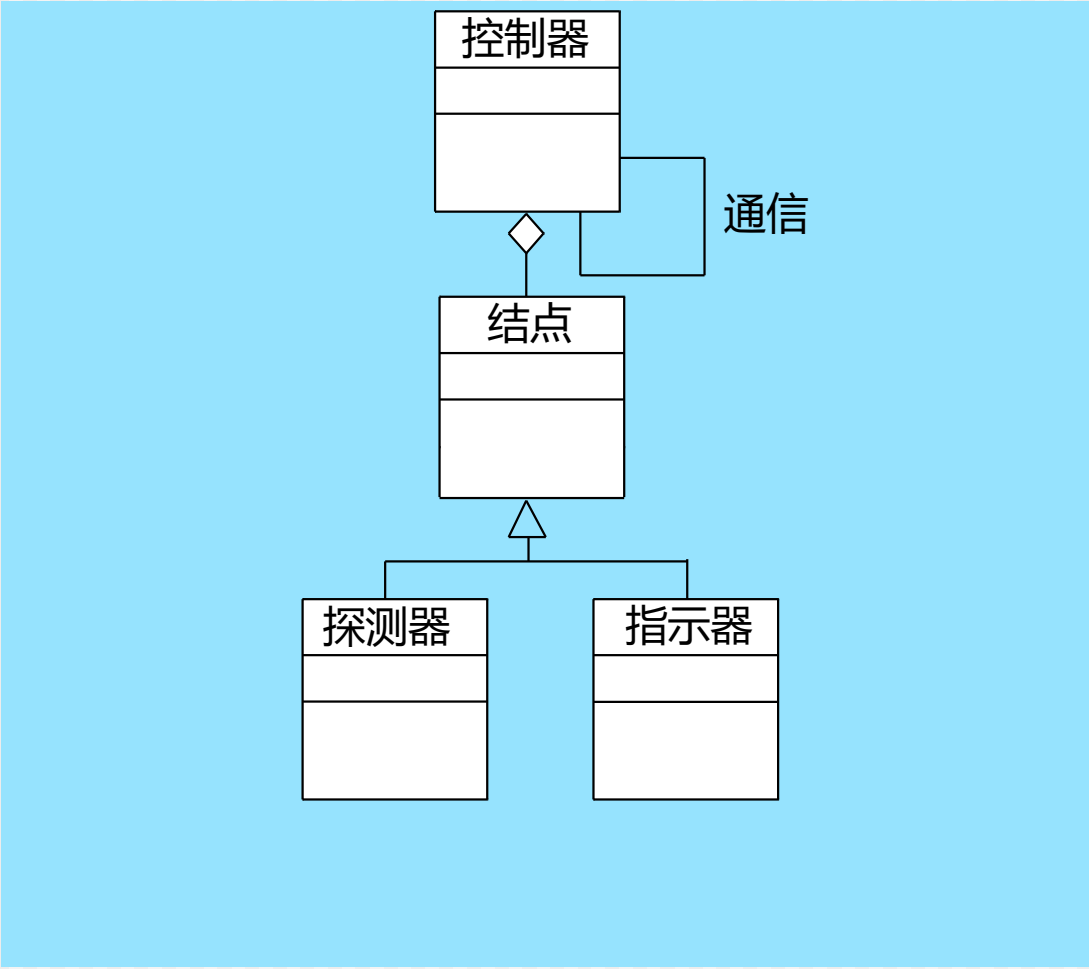
体系结构环境图

体系结构环境示例



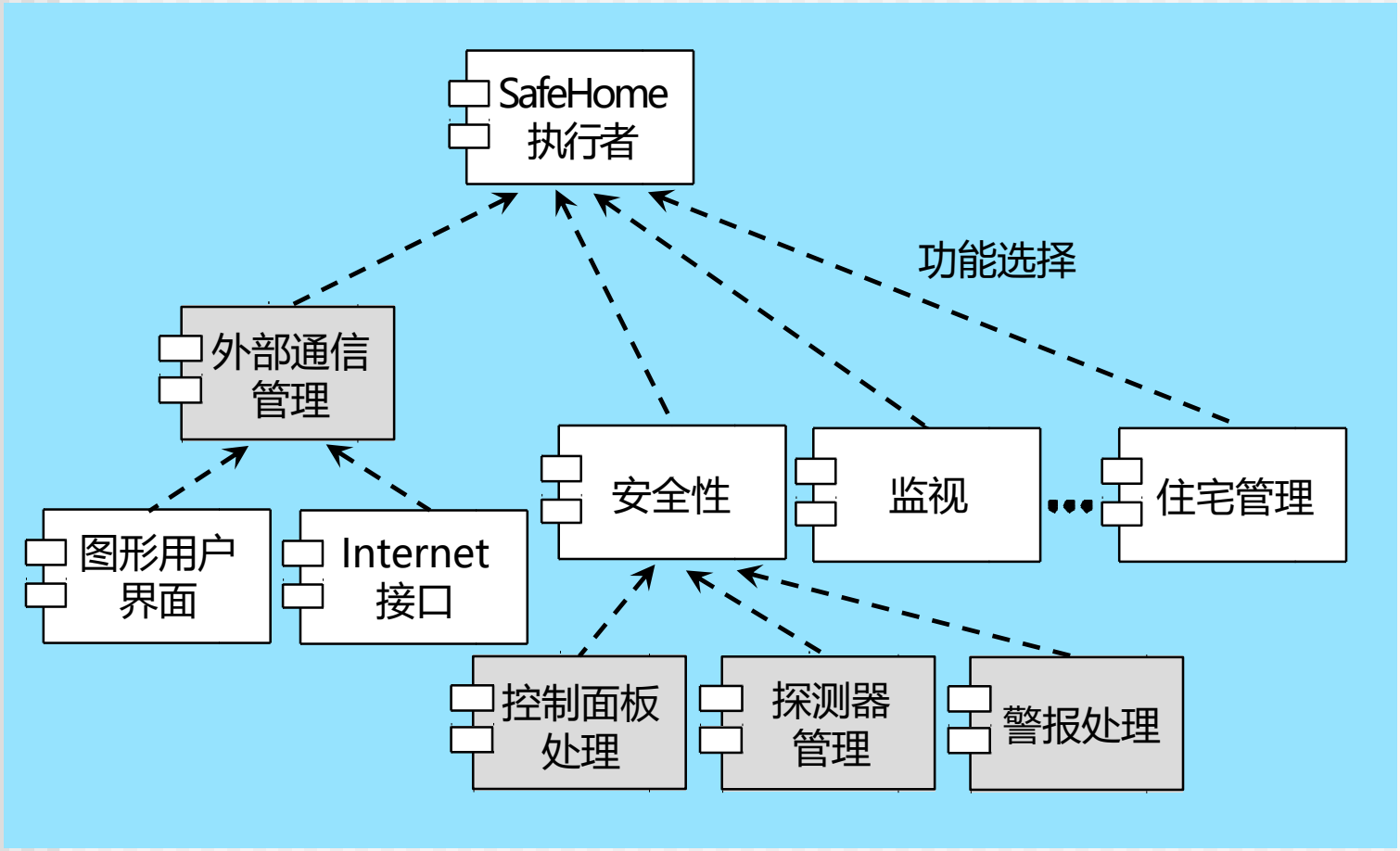
SafeHome安全功能的体系结构环境图

原型定义示例



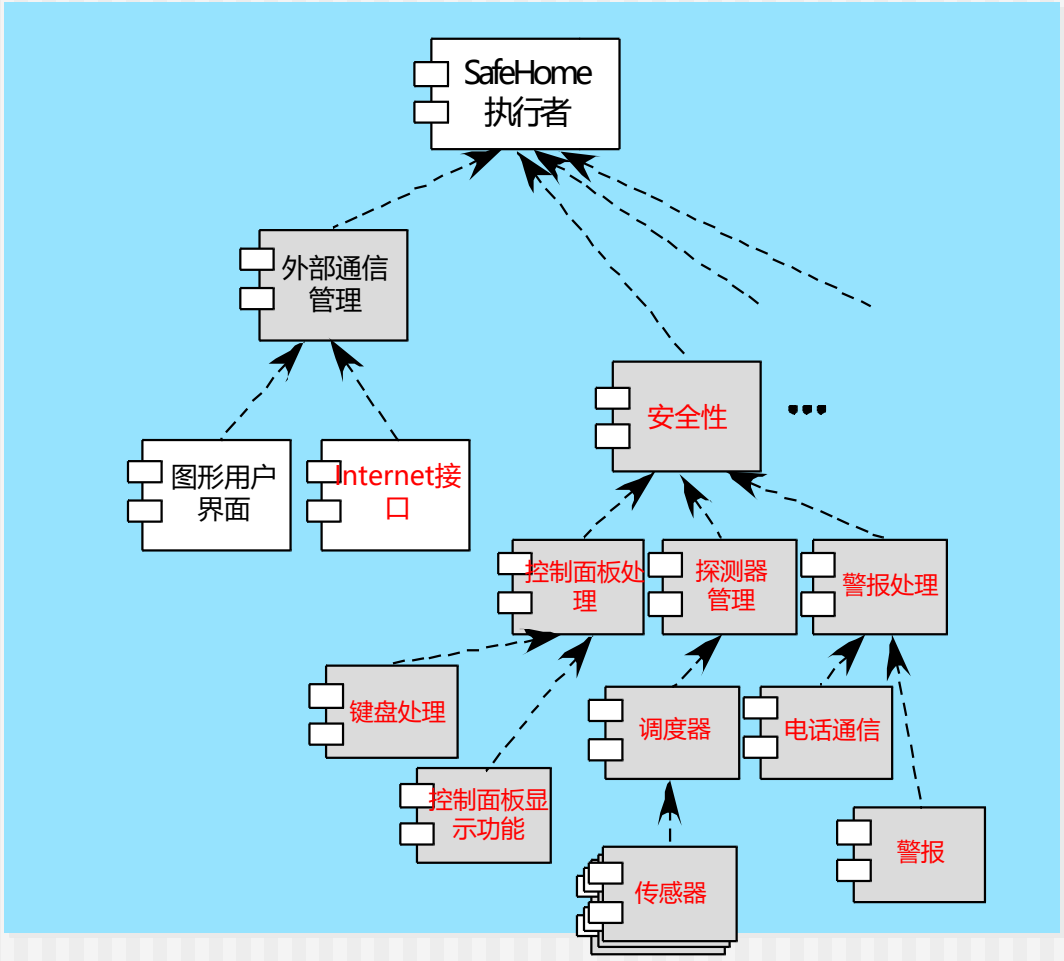
SafeHome安全功能原型的UML关系

细化体系结构为构件



带有顶层构件的SafeHome整体体系结构

精化构件结构（描述系统实例）



7.评估候选的体系结构设计

SEI提出的体系结构权衡分析法ATAM，评估过程：

1. 收集场景。
2. 引出需求、约束和环境描述。
3. 描述那些已经被选择用于解决场景和需求的体系结构风格/模式。
    - 模块视图
    - 过程视图
    - 数据流视图
4. 通过单独地考虑每个属性来评估质量属性。
5. 针对特定的体系结构风格，确定质量属性对各种体系结构属性的敏感性。
6. 使用在第5步进行的敏感性分析鉴定候选体系结构（在第3步开发的）。

体系结构复杂性

- 通过考虑体系结构中构件间的依赖，对体系结构的整体复杂性进行评估 [Zha98]
  - 共享依赖　表示在使用相同资源的消费者间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间的依赖关系。
  - 约束依赖　表示在一组活动间相关控制流上的约束。

体系结构描述语言（ADL）

- 体系结构描述语言（architectural description language, ADL）提供了一种描述软件体系结构的语义和语法。
- 为设计者提供以下能力：
  - 分解体系结构构件
  - 将单独构件组合成大的体系结构块
  - 表现构件之间接口（连接机制）
- UML包括很多体系结构描述所需要的部分，但其表达能力还不足以充分描述体系结构。

体系结构评审

- 评估软件体系结构满足系统质量需求（如可扩展性或性能）的能力以及识别任何潜在风险的能力
- 尽早检测到设计问题以降低项目成本的潜能
- 常用技术包括基于经验的推理 、原型评估、情境评审、以及检查单

基于模式的体系结构评审

1. 遍历相关的用例，以确定并讨论系统最重要的质量属性。
2. 结合需求讨论系统体系结构图。
3. 协助评审人员识别所使用的体系结构模式，并将系统结构与模式结构相匹配。
4. 使用现有文档和过往用例，检查体系结构和质量属性。
5. 识别并讨论由设计中使用的体系结构模式所引起的质量问题。
6. 针对会议上出现的问题作一个简短的汇总，并对可运行的系统骨架进行相应的修正。

敏捷性与体系结构

- 为避免返工，在编程前即根据用户故事集建立并形成体系结构模型（系统运行骨架）
- 混合模型允许软件架构师根据用户故事集进行故事情节串联
- 运作良好的敏捷项目要求每次“冲刺”迭代都应交付工作产品
- 复审每一次“冲刺”浮现出的代码是一种有用的体系结构评审方式

架构模式的分类

类别	目的	名称	特点	例子
从混沌到结构 (From Mud to Structure)	支持把整个系统任务以可控方式分解成相互协作的子任务	分层(Layer)	.直观的分而治之方式 .层的重用 .依赖性局部化 .可替代性	通信协议栈 Java虚拟机 很多应用系统
		管道-过滤器 (Pipes and Filters)	.重用性好 .便于重组新策略 .交互性差 .适用于复杂处理	编译器 Unix Shell DirectShow 通信协议解析
		黑板(Blackboard)	.算法和数据分离 .耦合度大 .适用于某些AI系统	人工智能系统
分布式系统 (Distributed System)	为分布式应用提供完备的基础架构（涉及到微核、管道和过滤器模式)	代理者 (Broker)	.屏蔽复杂性 .支持互操作 .适用于分布式系统	CORBA RMI 交叉编译系统
交互式系统 (Interactive System)	支持具有人机交互特征的软件系统的构建	MVC (Model-View-Controller)	.灵活性 .广泛流行	很多
		PAC (Presentation-Abstraction-Control)	.灵活性 .复杂	人工智能系统
适应性系统 (Adaptive System)	支持应用的扩展以适应技术的进步和变需求的变更	微内核 (Microkernel)	.适应变化 .支持长生命周期 .复杂	OS JBoss Eclipse
		基于元数据 (Meta-Level-Architecture)	.适应变化 .支持长生命周期 .不易实现	反射框架 Spring MDA

《面向模式的软件体系结构 – 卷1 模式系统》