软件工程

第12章 体系结构设计

徐本柱 软件学院

2018-10

主要内容

- 软件体系结构
- 体系结构类型
- 体系结构风格
- 体系结构考虑要素
- 体系结构决策
- 体系结构设计
- 评估候选的体系结构设计
- 经验学习
- 。 。 。

概念和重要性

- 体系结构设计表示了建立计算机系统所需的
  - 数据结构
  - 和程序构件。
- 需要考虑
  - 系统采取的体系结构风格，
  - 系统组成构件的结构、性质，
  - 以及系统中所有体系结构构件之间的相互关系。
- 体系结构设计
  - 提供“宏观”视图
  - 确保可以正确理解该视图

人员

- 软件工程师能够设计数据和体系结构，
  - 但在建造大型复杂系统时，
  - 往往由专家来完成。
- 数据库或者数据仓库设计者
  - 为系统创建数据体系结构。
- 系统体系结构设计师
  - 根据软件需求选择合适的体系结构风格。

步骤和产品

- 体系结构设计始于数据设计，
- 然后导出系统体系结构的一个或者多个表示
  - 对可选的体系结构风格或模式进行分析，
  - 得出最适合于客户需求和质量属性的**结构**。
  - 选定后使用体系结构设计方法对体系结构进行精化。
- 在体系结构设计过程中，将创建
  - 一个包括数据体系结构和程序结构的体系结构模型。
  - 还需描述构件的性质以及交互关系。

体系结构设计目标

- 设计描述为多步过程，从需求信息中综合出
  - 数据和程序的结构表示、
  - 接口特征和
  - 过程细节。
- 设计是由信息驱动的。
  - 软件设计方法通过考虑分析模型的三个域而得到
  - 信息、功能和行为三个域是创建软件设计的指南。
- 目标是
  - 提供导出体系结构设计的系统化方法
  - 体系结构设计是构建软件的初始蓝图。

12.1 软件体系结构

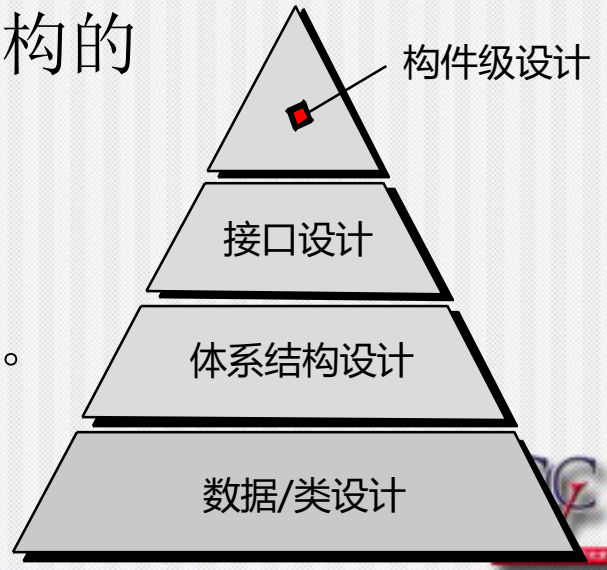
- 第一个程序→分成模块，软件系统→体系结构
- 程序员**负责模块间交互**和**模块装配**的全局属性。
- 好的软件开发人员
  - 经常采用一个或多个体系结构模式作为系统组织策略，
  - 但是只是非正式使用这些模式，
  - 并且在最终系统中没有将这些模式清楚地体现出来。

12.1.1 什么是体系结构

- 软件体系结构指系统的一个或者多个结构，包括
  - 软件的构件，
  - 构件的外部可见属性以及
  - 它们之间的相互关系。
- 体系结构并非可运行软件，是一种表达，能够：
  - (1)在满足既定的需求方面下，分析设计有效性；
  - (2)在设计变更相对容易的阶段，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。

体系结构的层次

- 在体系结构设计的环境中，
  - 软件构件简单到程序模块或者面向对象的类，
  - 包含数据库和客户与服务器网络配置的“中间件”
- 软件体系结构设计考虑设计金字塔中两层次
  - 数据设计和体系结构设计
  - 数据设计表示传统系统中体系结构的
    - 数据构件和
    - 面向对象系统中类的定义
  - 体系结构设计主要关注
    - 软件构件的结构、属性和交互作用。



12.1.2 为什么体系结构如此重要

- 三个关键原因：
  - 软件体系结构的表示有助于利益相关者开展交流。
  - 体系结构突出了早期设计决策，
    - 对随后的所有软件工程工作有深远的影响，
    - 对系统作为一个可运行实体的最后成功有重要作用。
  - “构建了一个相对小的，易于理解的模型，该模型描述了系统如何构成以及其构件如何一起工作”。
- 体系结构设计模型和体系结构模式都是可以传递的，
  - 体系结构的风格和模式可以被应用于其他系统的设计中，
  - 表示一组使软件工程师能以可预见的方式描述体系结构的抽象。

12.1.3 体系结构描述

- 不同参与者从不同角度理解体系结构
  - 角度由不同关注点驱动
  - 是一组体现系统不同视图的工作产品
- 描述目标：
  - 1 建立设计过程使用的概念性框架和词汇表
  - 2 提供体系结构描述的详细准则
  - 3 鼓励良好的体系结构设计实践
- 体系结构描述 (architectural description，AD)展示了多个视图，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”

12.1.4 体系结构决策

- 视图作为体系结构描述一部分
  - 解决特定利益相关者的关注点
  - 体系结构设计师考虑多种可选方案
  - 最终就最能满足关注点的体系结构特征决策
- 体系结构决策本身为体系结构的一种视图

INFO

体系结构决策描述模板

每个主要的体系结构决策可以被记录在案，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]所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、介绍）。

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，并描述为什么要摒弃这些方案。

争论：陈述你为什么选择了这种解决方案而不是其他的候选方案。

意义：指出制定决策对设计的影响。选择方案如何影响其他的体系结构设计问题？解决方案会在某种程度上约束设计吗？

相关决策：其他记录的决策和该决策有什么相关性？

相关关注点：其他需求和该决策有什么相关性？

工作产品：指出在体系结构描述中，决策会在哪里体现出来。

注释：参考可用来制定决策的其他团队的备忘录或文档。

12.2 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、厂房、仓库等
  - 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

12.3 体系结构风格

- 建筑师使用体系结构风格作为描述机制，
  - 将该房子和其他风格的房子区分开来。
- 更重要的是，体系结构风格也是建筑的样板。
- 进一步规定房子的细节，
  - 具体说明它的最终尺寸，
  - 进一步给出定制的特征，
  - 确定建筑材料等。
- 实际上是建筑风格指导了建筑师的工作。

软件体系结构风格定义

- 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

软件体系结构风格属性

- 一种体系结构风格是一种在整个系统设计上的变换
  - 目的就是为系统的所有构件建立一个结构。
- 在对已有体系结构进行再工程时，强制采用
  - 一种体系结构风格会导致软件结构的根本性改变，
  - 包括对构件功能的再分配。

12.3.1 体系结构风格的简单分类

- 以数据为中心的体系结构（黑板模式）
  - 数据存储位于这种体系结构的中心
  - 其他构件会经常访问数据存储，
  - 对存储中的数据进行更新、增加、删除或修改。
- 典型的以数据为中心的体系结构风格图12-1
- 在某些情况下数据存储库是被动的，
  - 即客户软件独立于数据的任何变化或其他客户软件的动作而访问数据。
- 一个变种是将中心存储库变换成“黑板”，
  - 当用户感兴趣的数据发生变化时，
  - 它将通知客户软件。

黑板结构

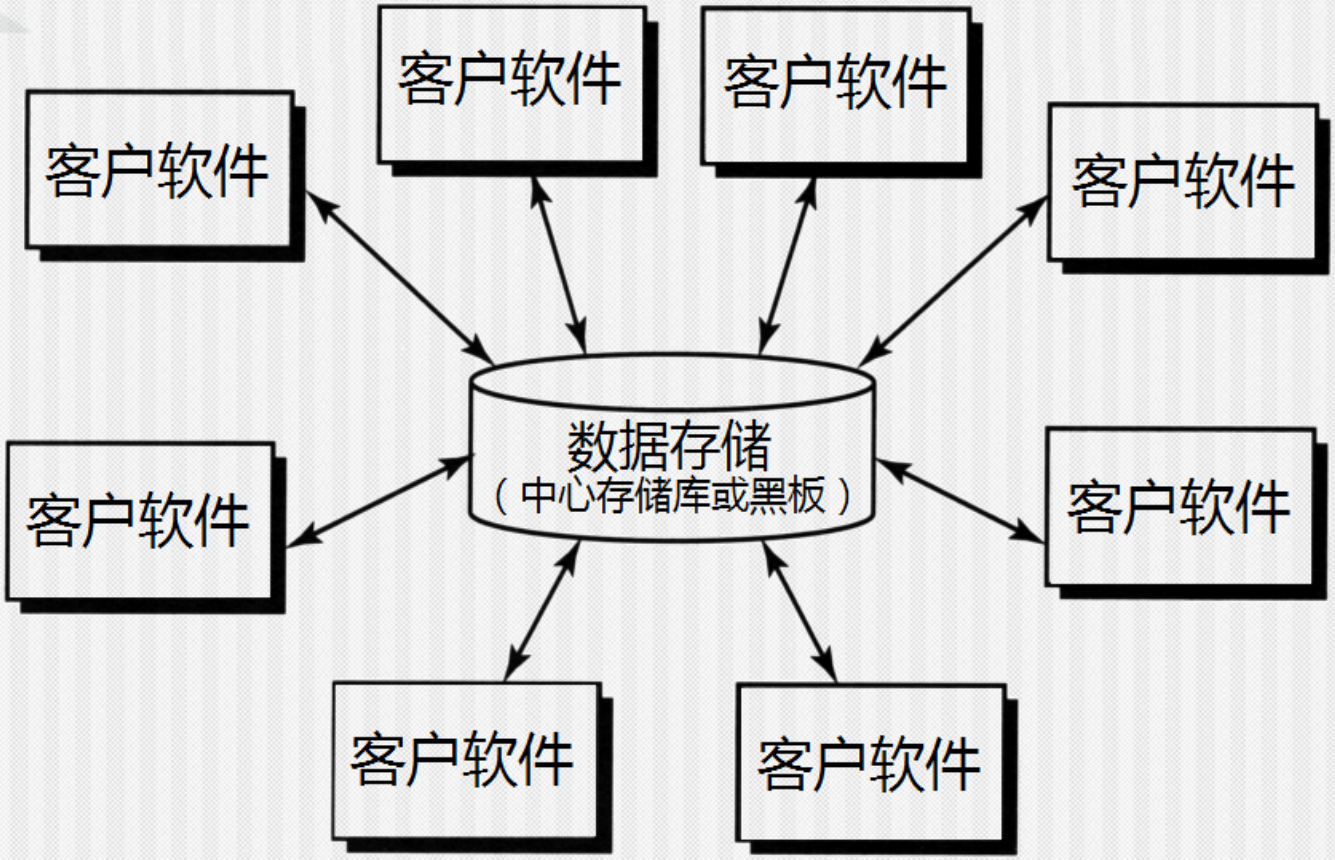


图12-1以数据为中心的体系结构

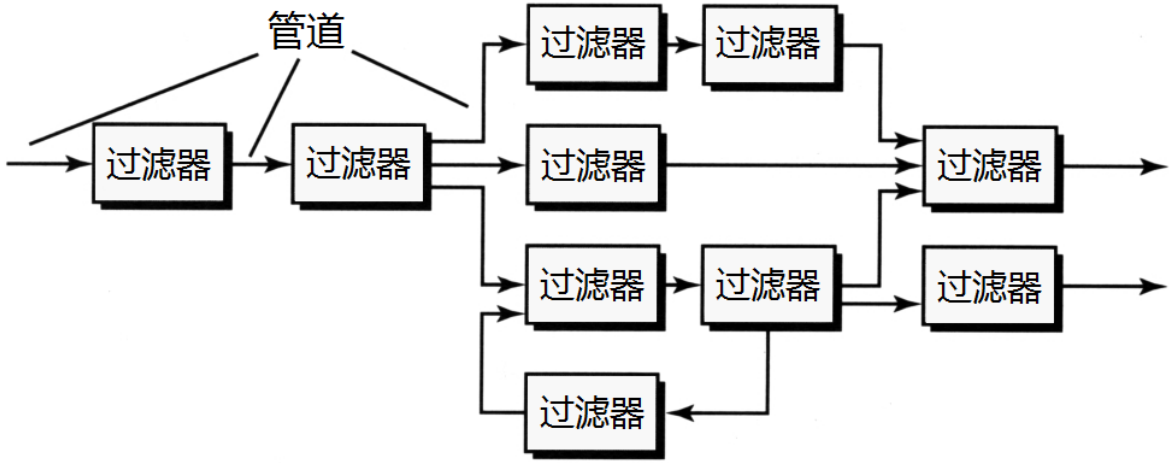
黑板模型的优点

- 以数据为中心的体系结构提升了可集成性，
  - 即现有的构件可以被修改
  - 新的客户构件可以加入到系统结构之中，
  - 而无需考虑其他的客户。
- 数据可以在客户间通过“黑板”机制传送，
  - 客户构件独立地执行过程。

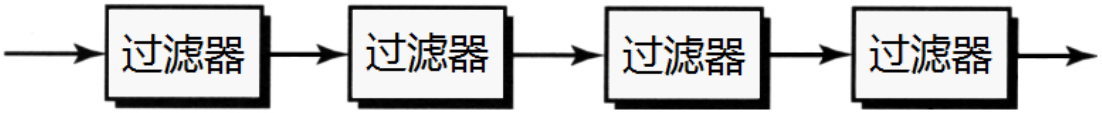
数据流体系结构

- 输入数据经过一系列的计算和操作构件的变换
  - 形成输出数据时，可以应用这种体系结构。
- 管道和过滤器结构(图12-2)
  - 拥有一组被称为过滤器的构件，
  - 通过管道连接，
  - 管道将数据从一个构件传送到下一个构件。
  - 每个过滤器独立于其上游和下游的构件而工作，
  - 过滤器的设计要针对某种形式的数据输入，
  - 产生某种特定形式的数据输出
  - 过滤器没有必要了解与之相邻的过滤器的工作。

体系结构风格的简单分类



(a)管道和过滤器



(b)批序列

数据流退化成单线的变换，则称为批序列。
接收一批数据，然后应用一系列连续的构件(过滤器)变换它。

调用和返回体系结构

- 能够设计出相对易于修改和扩展的程序结构
- 存在两种子风格：
  - 主程序/子程序体系结构
    - 将功能分解为一个控制层次，
    - 其中“主”程序调用一组程序构件，
    - 程序构件又去调用别的程序构件。图8-3描述了该种系统结构。
  - 远程过程调用体系结构
    - 主程序/子程序体系结构的构件分布在网络的多个计算机上。
- 面向对象体系结构
  - 系统的构件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与合作。

体系结构风格的简单分类

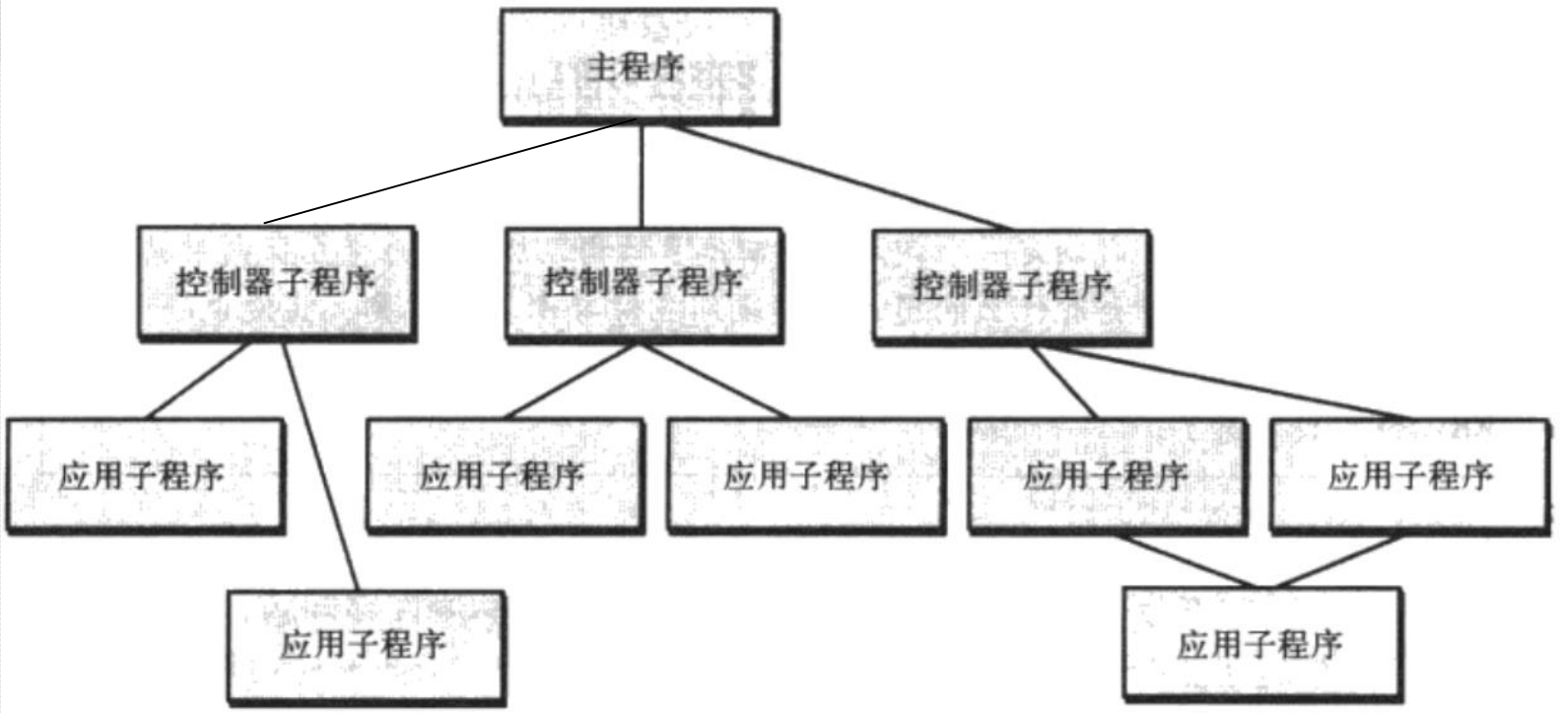
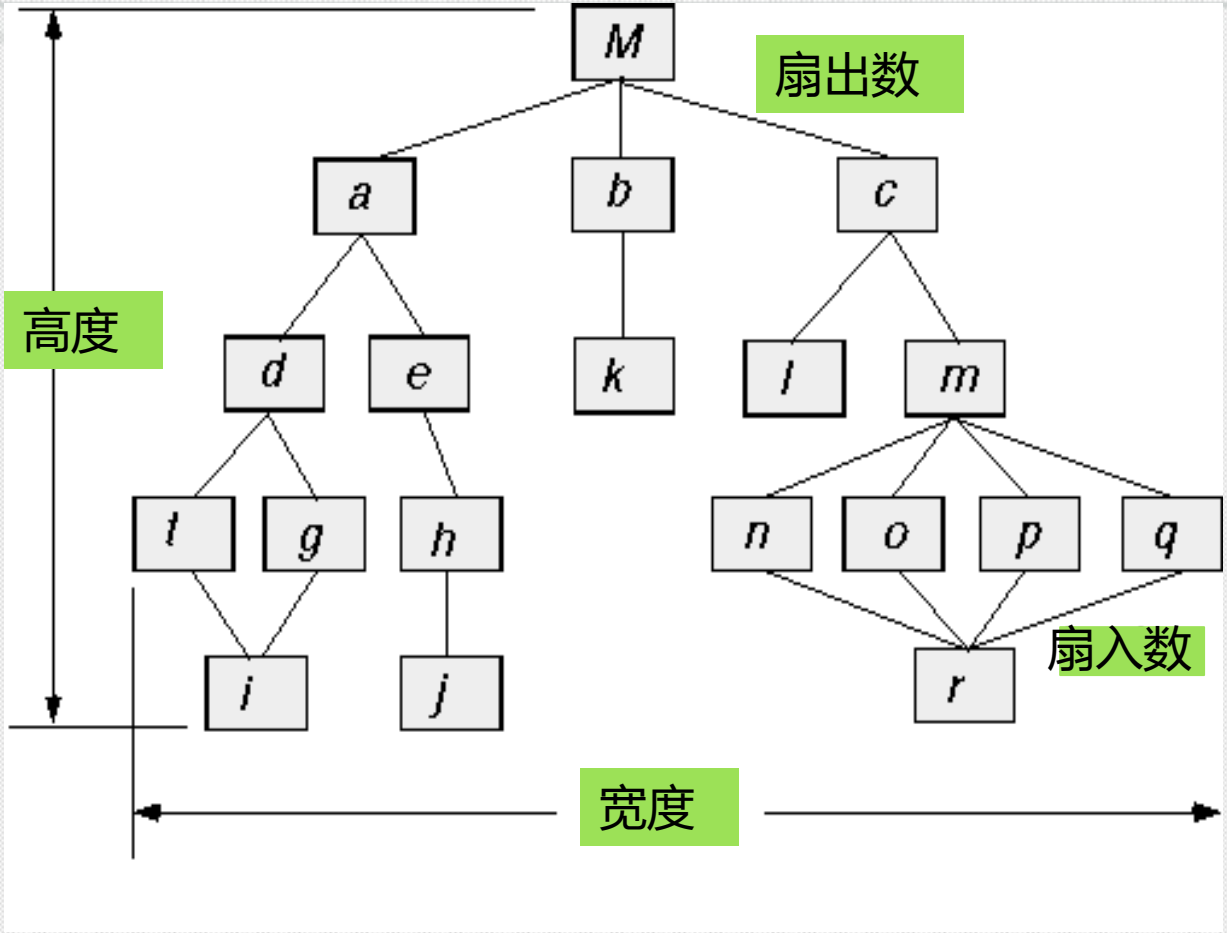


图12-3 主程序/子程序体系结构

调用和返回体系结构



层次体系结构

- 层次体系结构的基本结构如图12-4
  - 定义了一系列不同的层次，
  - 每个层次各自完成操作，
  - 这些操作不断接近机器的指令集。
- 在最外层，构件完成用户界面的操作；
- 在最内层，构件完成与操作系统的连接；
- 中间层提供各种实用程序服务和应用软件功能。

体系结构风格的简单分类

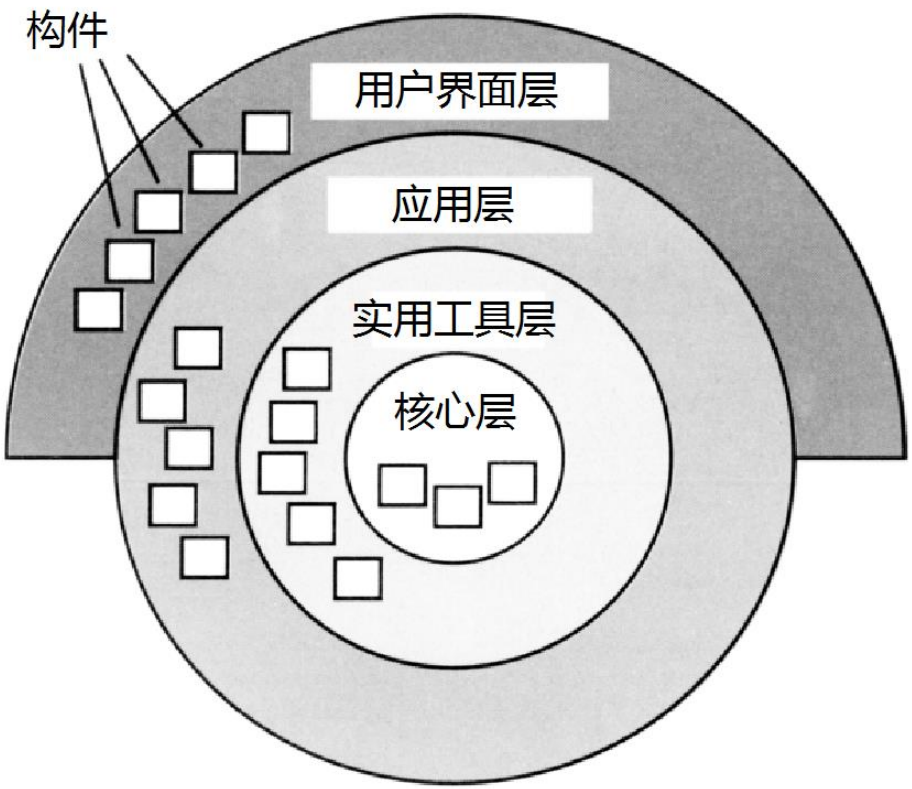


图12-4 层次体系结构

上述体系结构风格仅仅是软件设计师可用风格中的一小部分。

一旦需求工程提示了待构建系统的特征和约束，就可以选择最适合这些特征和约束的体系结构风格或者风格的组合。

在很多情况下，会有多种风格是适合的，需要对可选的体系结构风格进行设计和评估。

12.3.2 体系结构模式

- 开发需求模型时，软件必须解决许多问题
  - 问题很广泛
  - 跨越整个应用领域
- 需求模型定义了必须对问题回答的环境
- 一系列限制和约束会影响需解决问题的处理方式
- 模式提出了
  - 作为体系结构设计基础的
  - 体系结构解决方案
- 一系列常见问题，用具体的体系结构模式来处理

体系结构风格与模式的区别

- 体系结构模式也对体系结构的设计施加一种变换。
- 体系结构模式与风格在许多基本方面存在不同：
  - (1)体系结构模式涉及的范围要小一些，它更多集中在体系结构的某一局部而不是体系结构的整体；
  - (2)模式在体系结构上施加规则，描述了软件是如何在基础设施层次上（如并发）处理某些功能性方面的问题；
  - (3)体系结构模式倾向于在系统结构的环境中处理特定的行为问题。
- 模式与风格结合起来建立整个系统结构的外形。

12.3.3 组织和求精

- 设计过程留下一系列可供选择的体系结构
  - 需要一组用于评估所导出的体系结构设计的设计标准
- 下面问题有助于深入了解体系结构风格：
  - 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（即控制呈现的几何形状）是什么样？控制是否同步或构件是否异步操作？
  - 数据： 构件间如何进行数据通信？数据流是否是连接的，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（即，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，它们的角色是什么？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（即数据构件是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？

12.4 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

12.5 体系结构决策

1. 决定每一个决策所需要的信息条目
2. 定义决策和准确需求间的联系
3. 提供当评估备选决策时改变状态的机制
4. 定义可追溯的决策的先决关系条件
5. 关联重要决策和其产生的体系结构视图
6. 在制定时记录并沟通所有决策

12.6 体系结构设计

- 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
- 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

12.6.1 系统的环境表示

- 体系结构设计层，用体系结构环境图（ACD）
  - 对软件与外围实体的交互方式进行建模
- 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实现对环境的建模。
- 图12-5给出了体系结构环境图的通用结构。

系统的环境表示

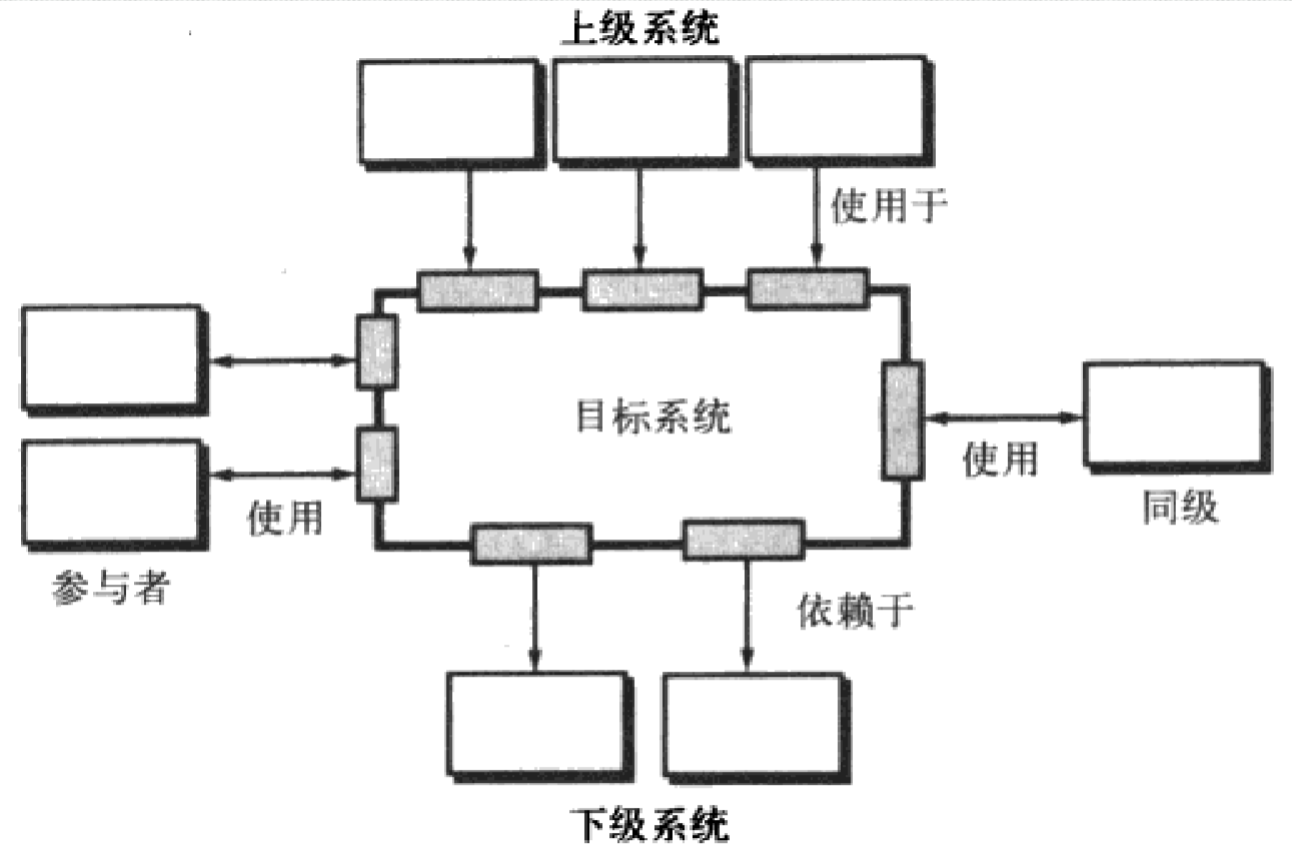


图12-5 体系结构环境图

系统的环境表示

与目标系统交互的系统可以表示为：

- 上级系统：作为目标系统高层处理方案的一部分。
- 下级系统：
  - 为目标系统所使用，
  - 并为了完成目标系统的功能提供必要的数据和处理。
- 同级系统：在对等的基础上相互作用。
- 参与者：
  - 指那些通过产生和消耗必不可少的处理所需的信息
  - 实现与目标系统交互的实体（人、设备）。

外部实体都通过某一接口与目标系统进行通信。

SAFEHOME实例

为了说明ACD的使用，我们再次考虑SafeHome产品的住宅安全功能。整个SafeHome产品的控制器和基于因特网的系统对于安全功能来说都处于上一级，在图9-6中它们在上方。监视功能是一个同级系统，并且在以后的产品版本中还要使用住宅安全功能（或被住宅安全功能使用）。户主和控制面板都是参与者，它们既是安全软件所用信息的生产者，又是安全软件所供信息的使用者。最后，传感器为安全软件所使用，并且在图中显示为下一级。

SAFEHOME实例

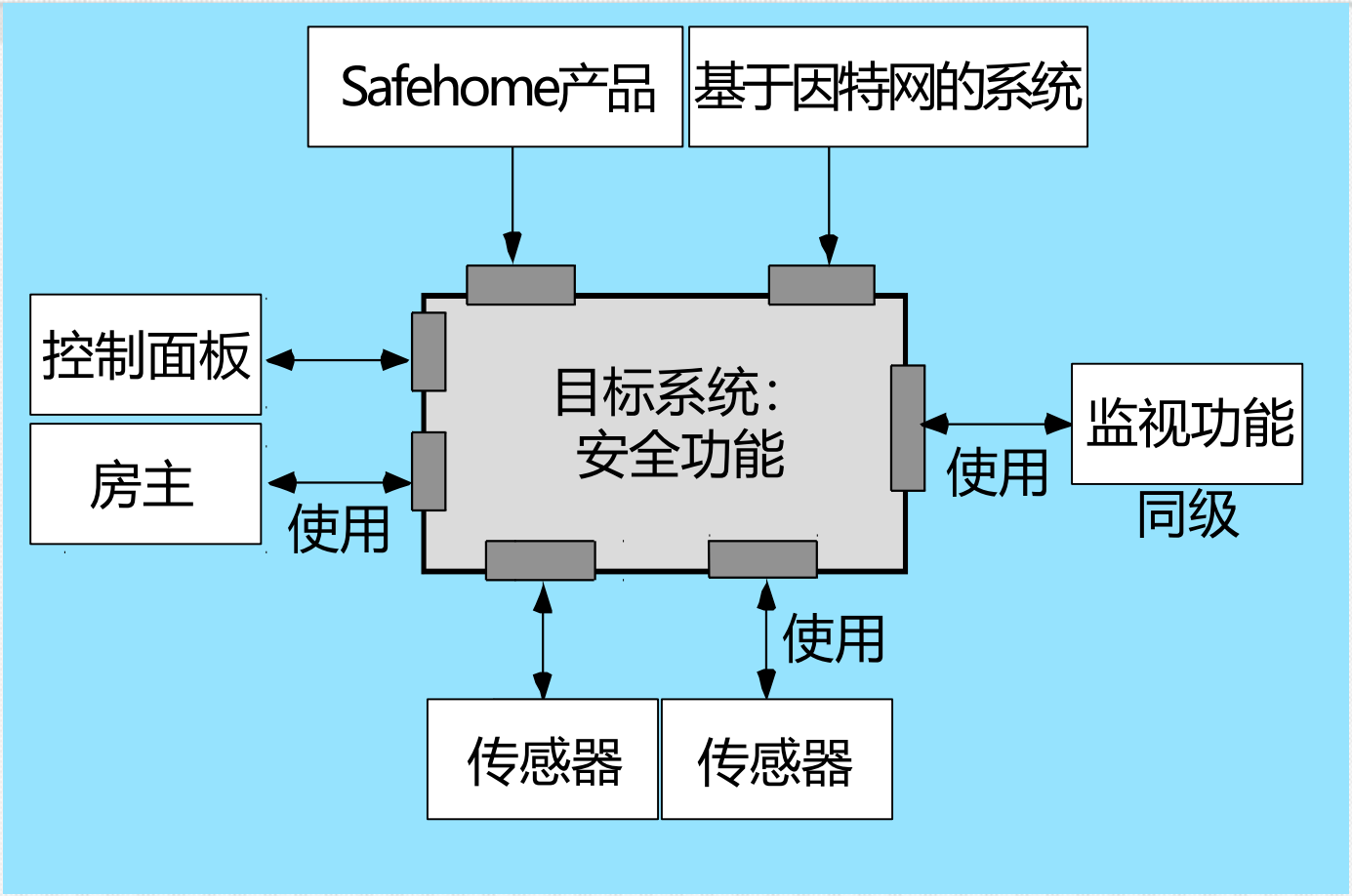


图12-6 SafeHome安全功能的体系结构环境图

12.6.2 定义原型

- 原型是表示核心抽象的类或模式
  - 抽象对于目标系统体系结构的设计非常关键
- 设计相对复杂的系统，只需相对较小的原型集合
- 目标系统的体系结构由这些原型组成，
  - 原型表示体系结构中稳定的元素
  - 可以基于系统行为可用多种方式进行对元素实例化
- 可通过检验需求模型的分析类获得

SAFEHOME实例

SafeHome住宅安全功能的讨论，可能会定义下面的原型：

- **结点**。表示住宅安全功能的输入和输出元素的内聚集合，例如，结点可能由如下元素构成：(1) 各种传感器；(2) 多种警报（输出）指示器。
- **探测器**。对所有为目标系统提供信息的传感设备的抽象。
- **指示器**。表示所有指示警报条件发生的报警机械装置（例如，警报汽笛、闪灯、响铃）的抽象。
- **控制器**。对允许结点发出警报或者撤销警报的机械装置的抽象。如果控制器安装在网络上，那么它们应该具有相互通信的能力。

SAFEHOME实例

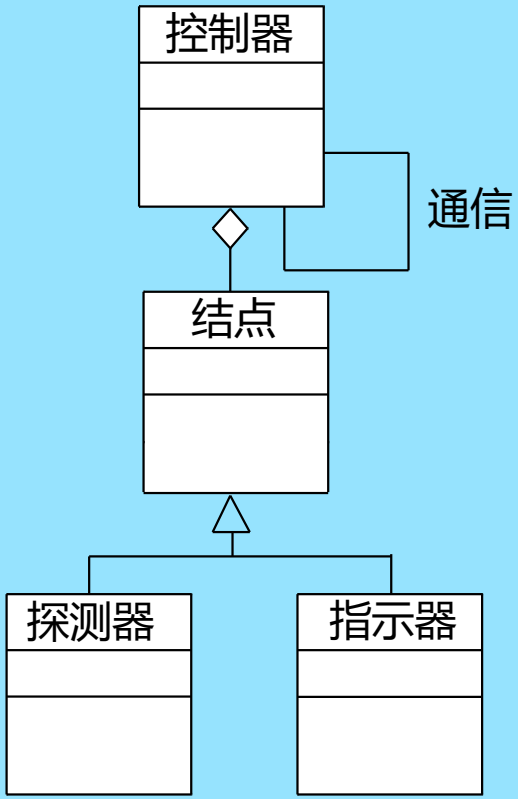


图12-7 SafeHome安全功能原始模型的UML关系图

12.6.3 将体系结构精化为构件

- 软件体系结构精化为构件时系统的结构开始显现
- 如何选择这些构件呢？
  - 从需求模型所描述的类开始。
    - 分析类表示软件体系结构中必须处理的业务领域内的实体。
    - 业务领域是构件导出和精化的源泉。
    - 另一源泉是基础设施域。
      - 体系结构必须提供很多基础设施构件使应用构件能够运作，
      - 但是这些基础设施构件与应用领域没有业务联系
      - 例如，内存管理构件、通信构件、数据库构件和任务管理构件经常集成到软件体系结构中。
    - 体系结构环境图描述的接口
      - 隐含着一个或多个特定的构件
      - 构件处理经过接口的数据

SAFEHOME实例

继续SafeHome住宅安全功能的例子，我们可以定义完成下列功能的顶层构件集合：

- 外部通信管理——协调安全功能与外部实体（如基于Internet的系统、外部报警通知）的通信。
- 控制面板处理——管理所有控制面板功能。
- 探测器管理——协调对系统所有探测器的访问。
- 警报处理——审核所有报警条件并执行相应动作。

每一个顶层构件都必须经过反复的精化迭代，然后被安置在SafeHome体系结构中。为每个构件都定义相应的设计类（包含必要的属性和操作）。然而，需要注意的是，在进行构件级设计之前，不要说明所有属性和操作的设计细节（第11章）。

图10-8描述了整个的体系结构（由UML构件图表示）。处理SafeHome图形用户界面和Internet接口的构件的事务一进入，就被外部通信管理获得。这个信息由用于选择合适产品功能（安全功能）的SafeHome执行者构件来管理。控制面板处理构件与户主交互来实现安全功能的安装/解除（arm/disarm）。探测器管理构件选取探测器检测报警条件，而当检测到警报时，警报处理构件将产生相应的输出。

SAFEHOME实例

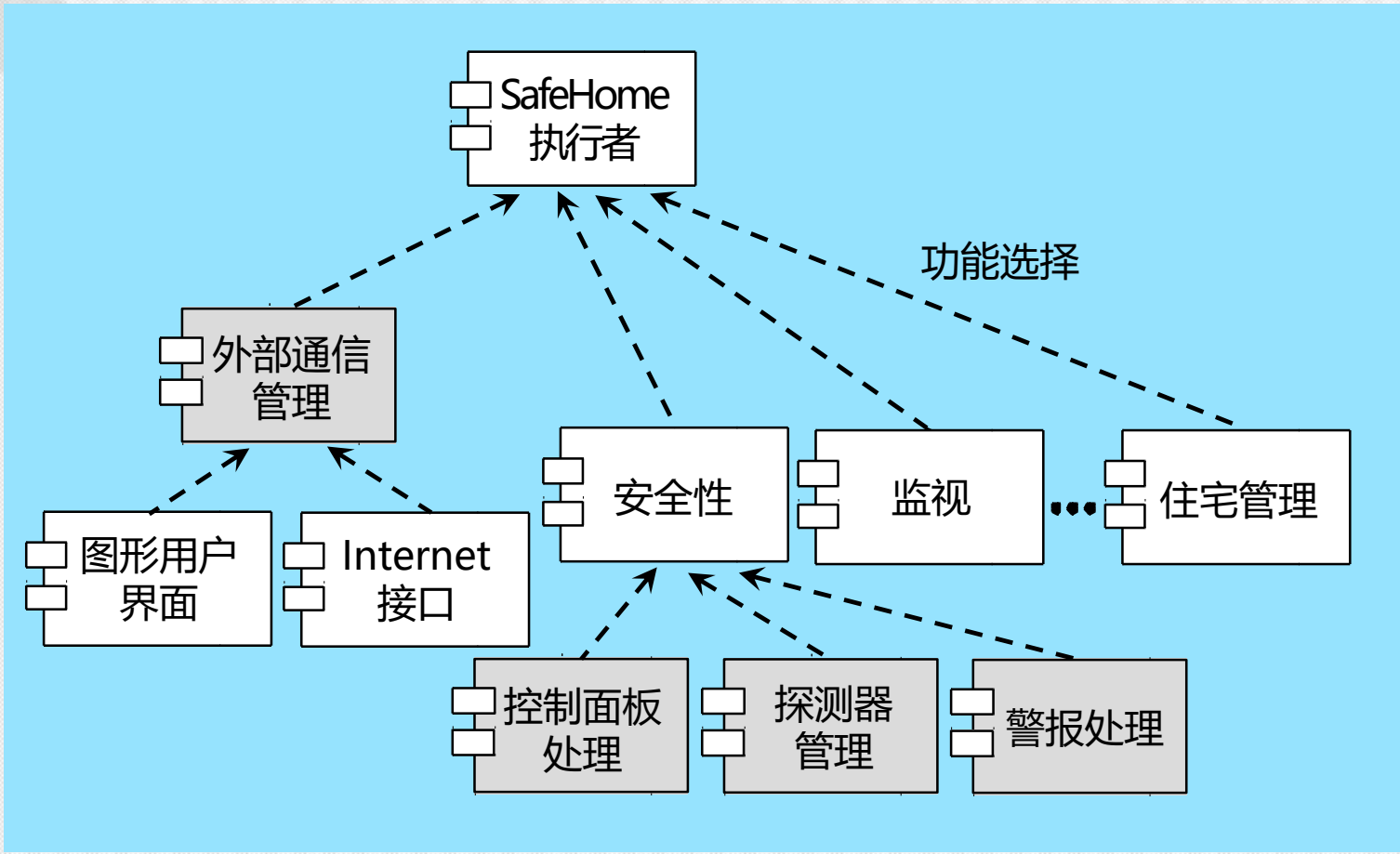


图12-8 带有顶层构件的SafeHome整体体系结构

12.6.4 描述系统实例

- 所建立的体系结构设计依然处于较高的层次
  - 系统环境已表明，
  - 问题域中抽象原型已定义，
  - 系统的整个结构已显现，
  - 并且主要的软件构件也都确定
  - 进一步的精化仍然是必要的。
- 为进行求精，需要开发体系结构的实例，
  - 将体系结构应用到一个特定的问题上，
  - 目的是证明结构和构件都是合理的。

SAFEHOME实例

图9-9描述的是安全系统SafeHome体系结构的一个实例。图9-8中显示的构件被进一步细化以显示更多的细节。例如，探测器管理构件与调度器基础设施构件相互作用，此基础设施构件实现安全系统中使用的每个传感器对象的定时查询。图9-8中显示的每个构件都作了类似的细化。

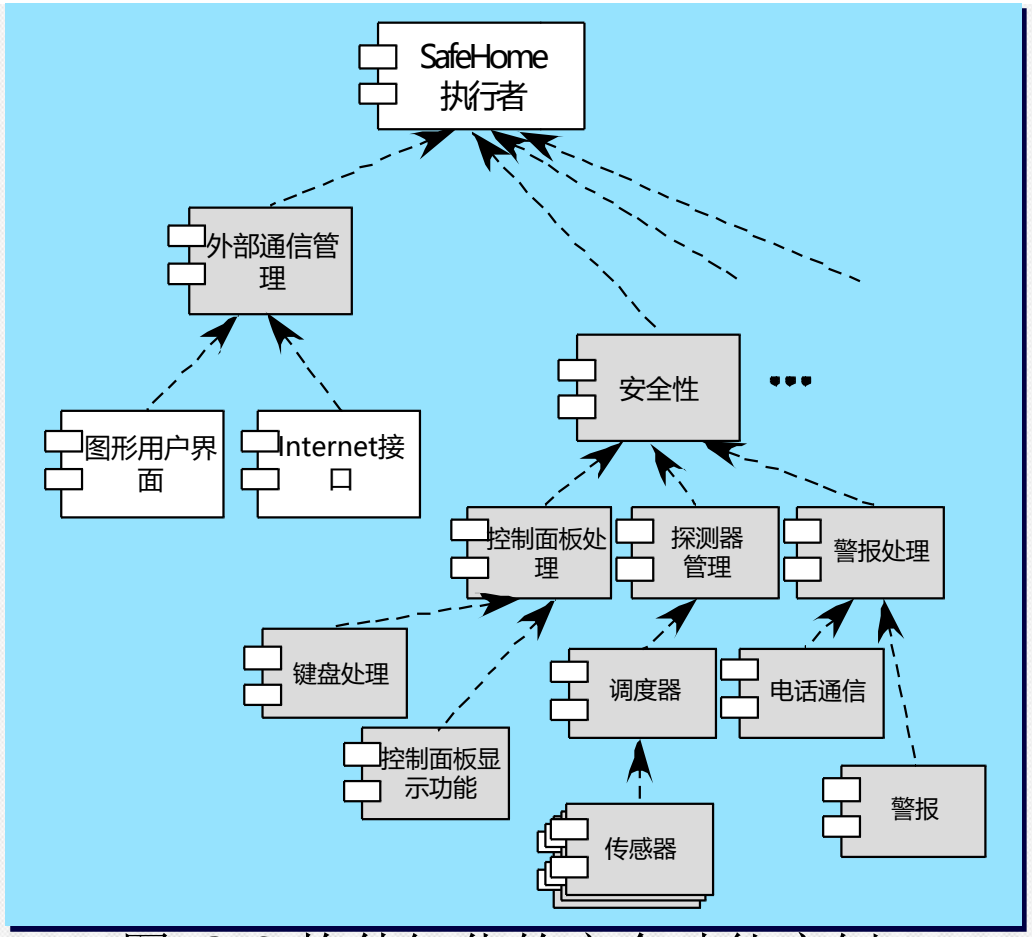


图12-9 构件细化的安全功能实例

12.7 评估可选的体系结构设计

- 设计会导致多种可供选择的候选体系结构，
  - 每一种候选体系结构都需要评估，
  - 以确定哪种体系结构最适合要解决的问题。

体系结构权衡分析方法

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）。

12.7.1 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）

12.7.2 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

12.9 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。

12.11 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

谢谢！