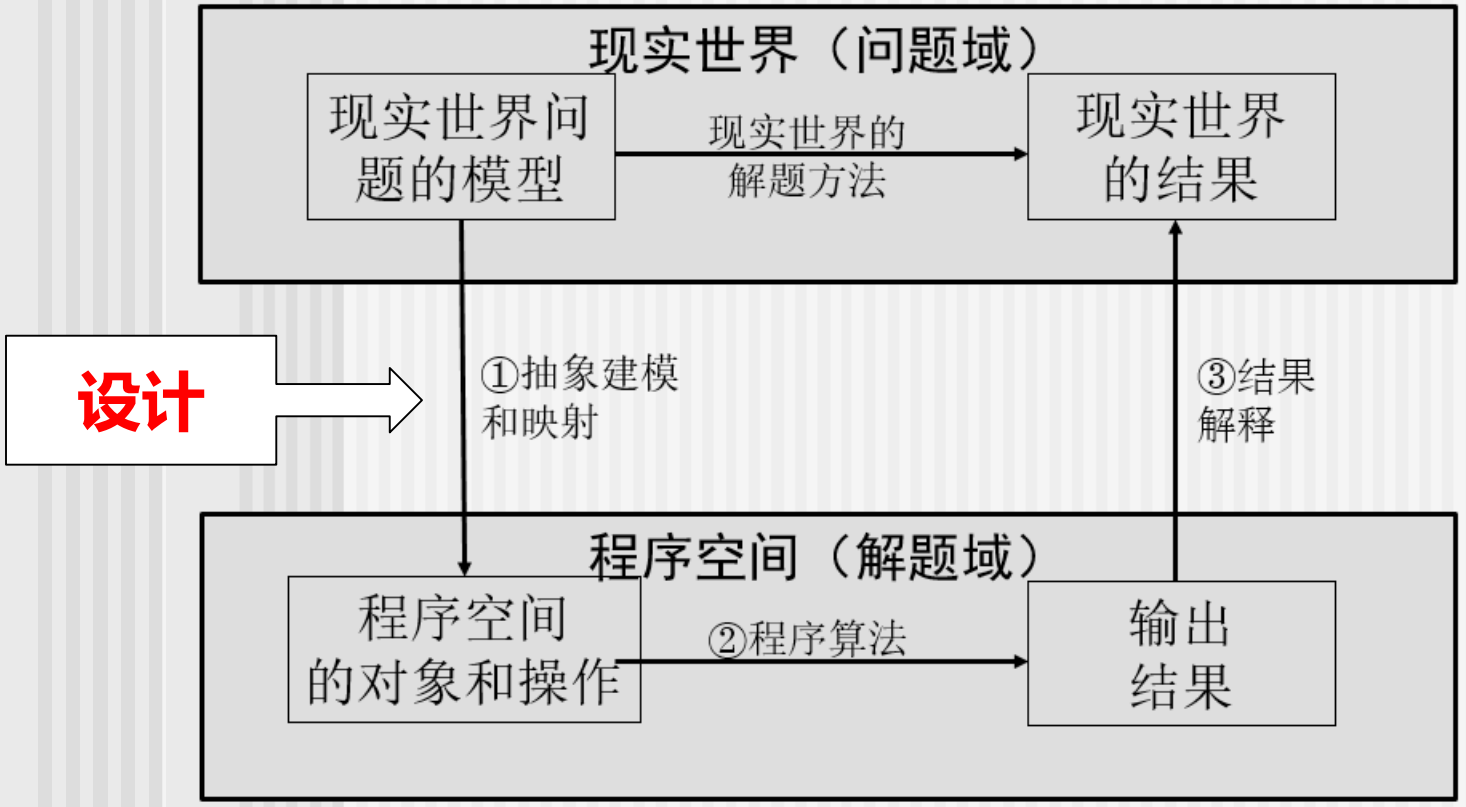
第12章 软件设计的基本概念

1.什么是软件设计?


现实世界（问题域）
现实世界问题的模型
现实世界的解题方法
现实世界的结果
设计
①抽象建模和映射
③结果解释
程序空间（解题域）
程序空间的对象和操作
②程序算法
输出结果

软件设计的目的

- 创建软件的表示或模型，为将要构建的系统或产品提供软件体系结构、数据结构、接口和构件的细节。
- 设计要包含需求分析规范说明（需求模型）中规定的所有需求的实现
- 从软件设计开始，软件工程师就工作在解题空间（程序空间）中，所使用的术语往往都是计算机领域的。

软件设计过程

- 包括一系列原理、概念和实践，可以指导高质量的系统或产品开发
- 设计原理建立了指导设计工作的最重要原则
- 软件设计随着新的方法、更好的分析和更广泛理解的进展而不断发展

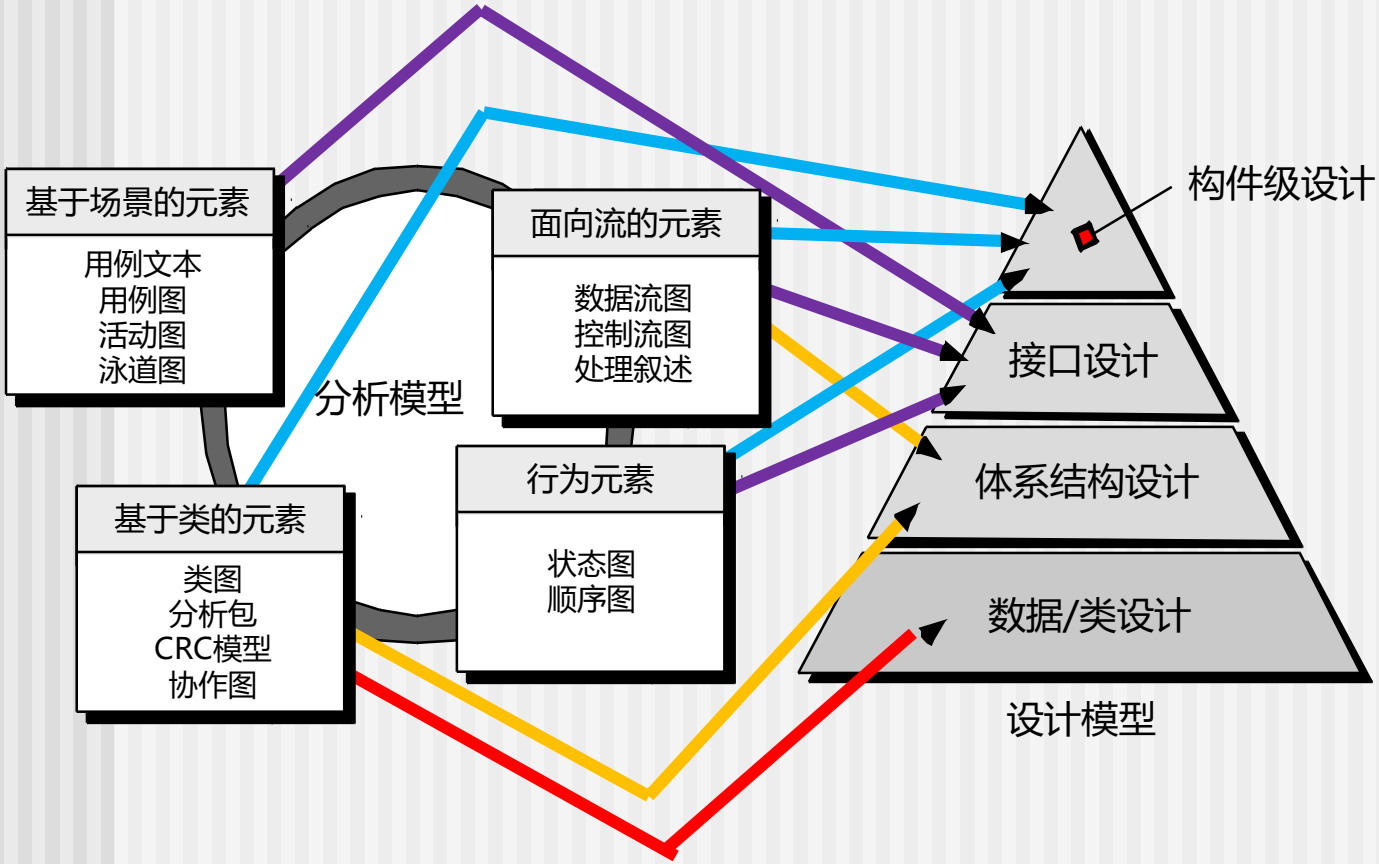
软件设计的内容

- 数据/类设计：将分析类模型变换为软件设计类的实现模型及相应的数据结构。
- 体系结构设计：描述软件主要构造元素之间的关系、可满足系统需求的体系结构风格和模式，以及影响体系结构实现的约束。

软件设计的内容

- 接口设计：描述软件元素、硬件元素和终端用户间如何通信，接口意味着信息流（如数据和控制）和特定的行为类型。
- 构件级设计：将软件体系结构的构造元素变换为对软件构件的过程性描述

分析模型 → 设计模型



需求模型到设计模型的转换（数据是所有其他设计元素的基础）

2.软件设计过程

- 软件设计是一个迭代的过程
- 通过设计过程，需求被变换为用于构建软件的“蓝图”。蓝图描述了软件的整体视图，初始的设计是在高抽象层次上的表达，可以直接跟踪到特定的系统目标以及更详细的数据、功能和行为需求。（概要设计）
- 后续的细化导致更低抽象层次的设计表示。当然这些标识仍然能够跟踪到需求，但是连接更加错综复杂了。（详细设计）
- 通过技术评审评估设计质量

软件设计方法的演化

- 模块化程序开发和自顶向下的“结构化”方式对软件进行求精的方法
- 面向对象的方法
- 设计模式
- 面向方面的方法
- 模型驱动开发
- 测试驱动开发

通用设计任务集

1. 检查信息域模型，并为数据对象及其属性设计合适的数据结构。
2. 使用分析模型选择一种适用于软件的体系结构风格（模式）。
3. 将分析模型分割为若干设计子系统，并在体系结构内分配这些子系统：
   - 确保每个子系统是功能内聚的。
   - 设计子系统接口。
   - 为每个子系统分配分析类或功能。
4. 创建一系列的设计类或构件：
   - 将分析类描述转化为设计类。
   - 根据设计标准检查每个设计类，考虑继承问题。
   - 定义与每个设计类相关的方法和消息。
   - 细化每个构件的接口。
   - 定义构件级的数据结构。
   - 评估设计类或子系统，并为这些类或子系统选择设计模式。
   - 评审设计类，并在需要时进行修改。
5. 设计外部系统或设备所需要的所有接口。
6. 设计用户接口：
   - 评审任务分析的结果。
   - 基于用户场景对活动序列进行详细说明。
   - 创建接口的行为模型。
   - 定义接口对象和控制机制。
   - 评审接口设计，并根据需要进行修改。
7. 进行构件级设计：
   - 在相对较低的抽象层次上详细描述所有算法。
   - 评审每个构件并修正所有已发现的错误。
8. 开发部署模型。

设计的质量目标

- 设计必须实现所有包含在需求模型中的明确需求，而且必须满足利益相关者期望的所有隐含需求。
- 对于那些生成代码的人和那些进行测试以及随后维护软件的人而言，设计必须是可读的、可理解的指南。
- 设计必须提供软件的全貌，从实现的角度说明数据域、功能域和行为域。

设计的质量指导原则

- 设计应展示出这样一种结构：（1）已经使用可识别的体系结构风格或模式创建；（2）由展示出良好设计特征的构件构成（将在本章后面讨论）；（3）能够以演化的方式实现，从而便于实现和测试。
- 设计应该模块化；也就是说，应将软件逻辑地划分为元素或子系统。
- 设计应该包含数据、体系结构、接口和构件的清晰表示。
- 设计应导出数据结构，这些数据结构适用于要实现的类，并从可识别的数据模式提取。
- 设计应导出显示独立功能特征的构件。
- 设计应导出接口，这些接口降低了构件之间以及与外部环境连接的复杂性。
- 设计的导出应根据软件需求分析过程中获取的信息采用可重复的方法进行。
- 设计应使用能够有效的表示法来传达其意义。

设计原则

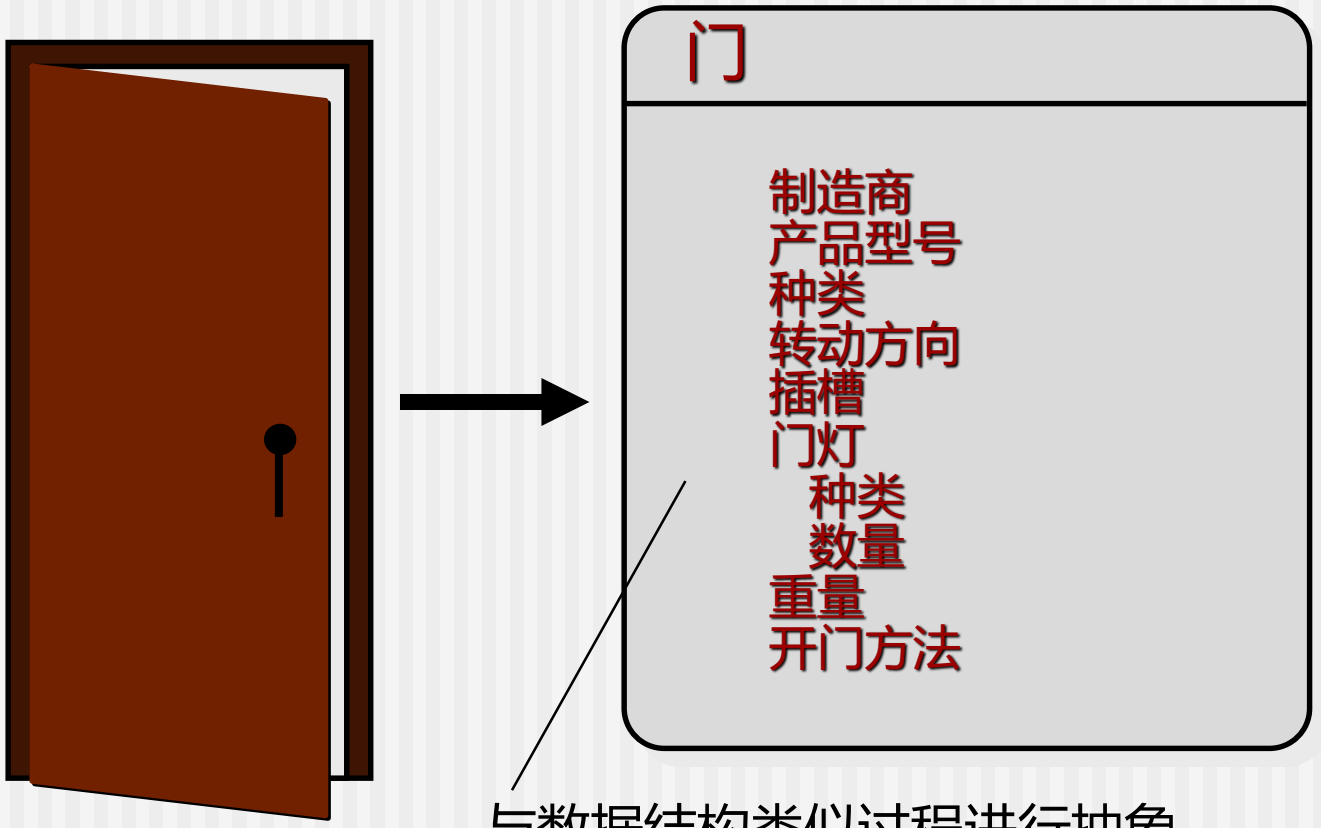
- 设计过程不应做“井底之蛙”；
- 设计应具有可追踪性，能回溯到分析模型；
- 设计不必每次从头做起；
- 设计应缩小软件与现实世界中问题的“智力距离”
- 设计应展现一致性和集成性；
- 设计应适应可能产生的变更；
- 设计应能够平滑处理异常数据，时间或操作条件；
- 设计不是编码，编码也不是设计；
- 设计之初即能够评估质量，而不是在编码之后；
- 设计应坚持评审以减少概念性（语义性）错误；

来自 Davis [DAV95]

3.软件设计的概念

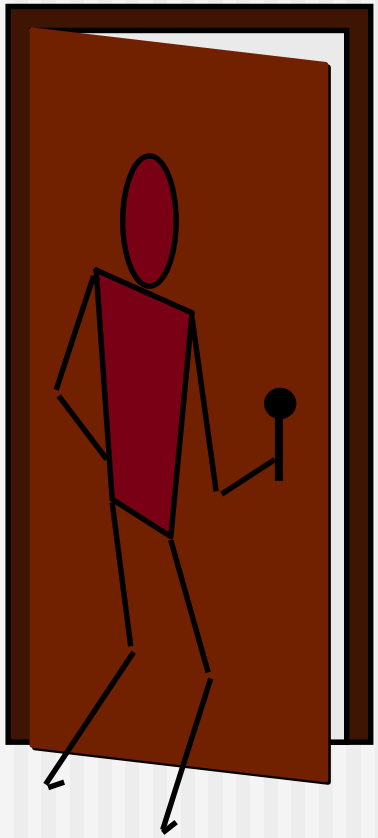
- 抽象——数据，过程
- 体系结构——软件的整体结构
- 模式——传递已验证设计方案的精髓
- 关注点分离——任何复杂问题在被分解为若干块后将更易处理
- 模块化——数据和功能的分割
- （信息）隐蔽——控制接口
- 功能独立——单一功能和低耦合
- 求精——细化所有抽象的细节
- 方面——理解全局需求如何影响设计的机制
- 重构——简化设计的重组技术
- 面向对象的设计概念——附录 II
- 设计类——提供设计细节以实现分析类

数据抽象



与数据结构类似过程进行抽象

过程抽象



开门

进门的细节

方法

运用与进入有关的事物的“知识”完成抽象

体系结构

“软件的整体结构和这种结构为系统提供概念完整性的方式” **[SHA95a]**

结构特性。体系结构设计中，这一特性定义了系统的构件（如模块、对象、过滤器）、构件被封装的方式以及构件之间相互作用的方式。例如，对象封装了数据和过程，过程操纵数据并通过方法调用进行交互。

外部功能特性。体系结构设计描述应当指出设计体系结构如何满足需求，这些需求包括：性能需求、能力需求、可靠性需求、安全性需求、可适应性需求以及其他系统特征需求。

相关系统族。体系结构应当能抽取出在一类相似系统开发中经常遇到的重复性模式。本质上，设计应当能够重用体系结构构件。

体系结构

“软件的整体结构和这种结构为系统提供概念完整性的方式” **[SHA95a]**

结构特性。体系结构设计中，这一特性定义了系统的构件（如模块、对象、过滤器）、构件被封装的方式以及构件之间相互作用的方式。例如，对象封装了数据和过程，过程操纵数据并通过方法调用进行交互。

外部功能特性。体系结构设计描述应当指出设计体系结构如何满足需求，这些需求包括：性能需求、能力需求、可靠性需求、安全性需求、可适应性需求以及其他系统特征需求。

相关系统族。体系结构应当能抽取出在一类相似系统开发中经常遇到的重复性模式。本质上，设计应当能够重用体系结构构件。

模式

设计模式模板

模式名称 —— 以简短但富于表现力的名字描述模式的本质

目的 —— 描述模式及模式所做的工作

别名 —— 列举模式的所有同义名

动机 —— 提供问题的实例

适用性 —— 注明模式所使用的特殊情况

结构 —— 描述完成模式所需的类

关联者 —— 描述完成模式所需的类的职责

协作 —— 描述其他模式对解决方案的贡献

效果 —— 描述实现模式时必须考虑的可能要做的折中以及使用模式的效果

相关模式 —— 相关设计模式的交叉索引

关注点分离

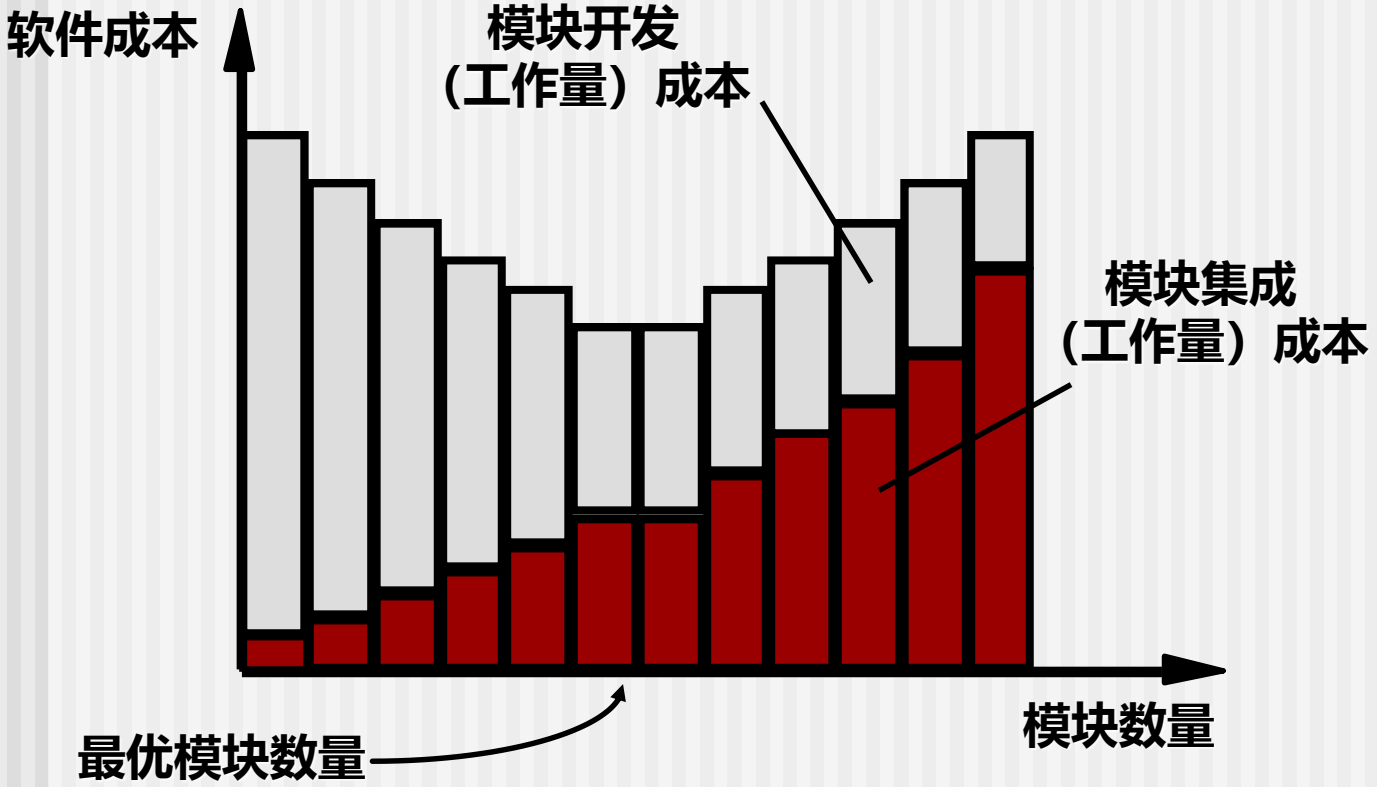
- 如果被分解为可以独立解决和（或）优化的若干块，任何复杂问题都能够更容易地被处理。
- 一个**关注点**是一个特征或行为，被指定为软件需求模型的一部分。
- 通过将关注点分割为更小的关注点（由此产生更多可管理的块），使得解决一个问题需要付出更少的工作量和时间。

模块化

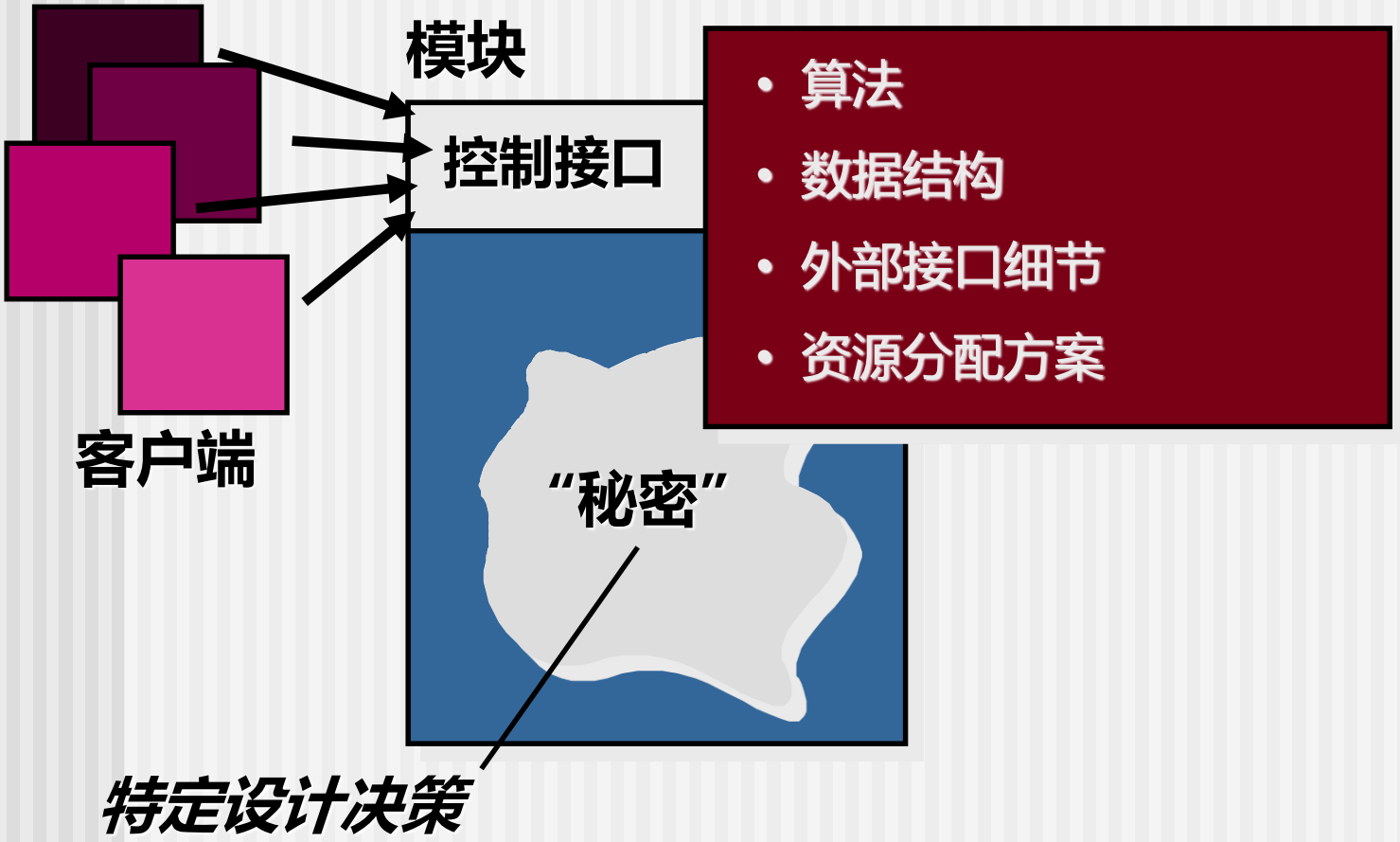
- “模块化是软件的单一属性，它使程序能被智能化地管理”[Mye78]
- 软件工程师难以掌握单块软件（即由一个单独模块构成的大程序）。
  - 其控制路径的数量、引用的跨度、变量的数量和整体的复杂度使得理解这样的软件几乎是不可能的。
- 几乎所有的情况下，为了理解更容易，都应当将设计划分成许多模块，这样做的结果，构建软件所需的成本将会随之降低。

模块化：权衡

对于特定的软件设计来说，
多少数量的模块是“合适的”？



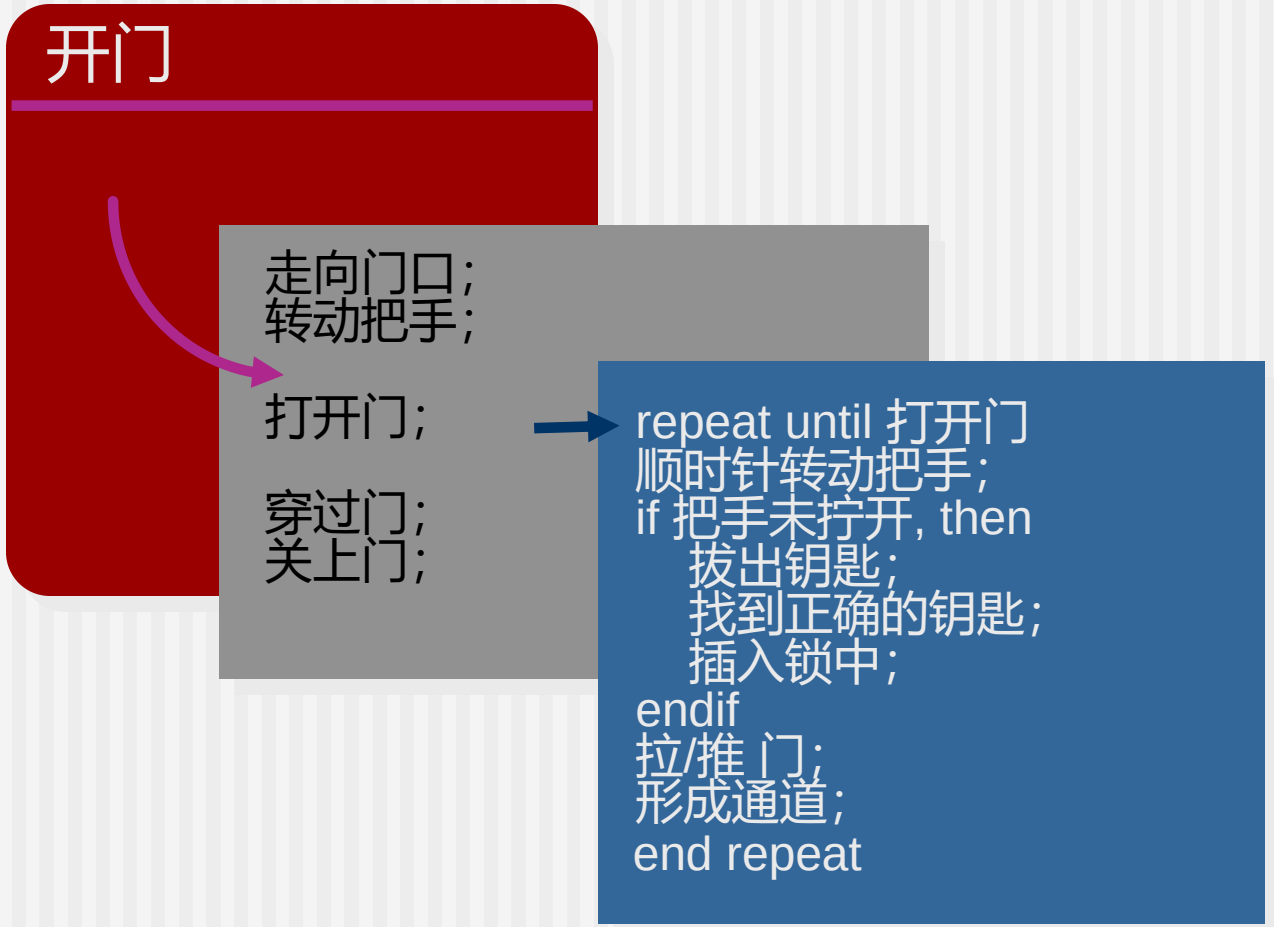
信息隐蔽



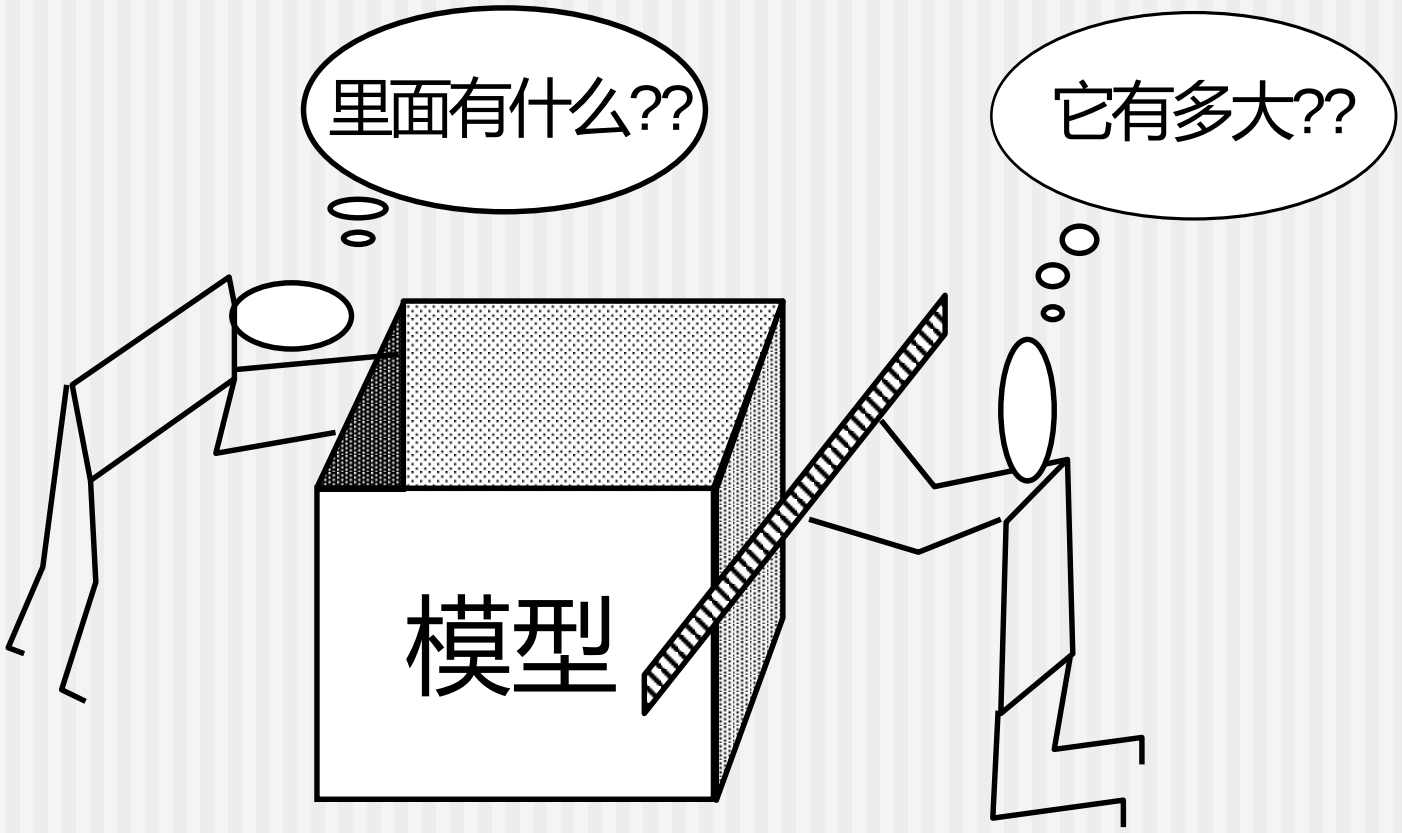
为什么隐蔽信息?

- 降低“副作用”的可能性;
- 减少局部设计决定对全局的影响;
- 突出控制接口处通信;
- 阻止全局数据使用;
- 促进封装——高质量设计的属性之一;
- 形成高质量软件;

逐步求精


开门
走向门口;
转动把手;
打开门;
穿过门;
关上门;
repeat until 打开门
顺时针转动把手;
if 把手未拧开, then
拔出钥匙;
找到正确的钥匙;
插入锁中;
endif
拉/推 门;
形成通道;
end repeat

测量模块：两种观点



功能独立

- 通过开发具有“专一”功能和“避免”与其他模块过多交互的模块，可以实现功能独立。
- *内聚性* 显示了某个模块相关功能的强度；
  - 一个内聚的模块执行一个独立的任务，与程序的其他部分构件只需要很少的交互。简单地说，一个内聚的模块应该（理想情况下）只完成一件事情。
- *耦合性* 显示了模块间的相互依赖性；
  - 耦合性依赖于模块之间的接口复杂性、引用或进入模块所在的点以及什么数据通过接口进行传递。

方面

- 考虑两个需求，A和B。“如果已经选择了一种软件分解[精化]，在这种分解中，如果不考虑需求A的话，需求B就不能得到满足”[Ros04]，那么需求A横切需求B。
- 一个*方面*是一个横切关注点的表示。

方面——举例

- 考虑**SafeHomeAssured.com**网站应用中的两个需求。依照用例ACS-DCV()描述需求A，设计求精将集中于那些能够使注册用户通过放置在空间中的相机访问视频的模块。需求B是一个通用的安全需求，要求注册用户在使用**SafeHomeAssured.com**之前必须先进行验证，该需求用于*SafeHome*注册用户可使用的所有功能中。当设计求精开始的时候，A*是需求A的一个设计表示，B*是需求B的一个设计表示。因此，A*和B*是关注点的表示，且B*横切A*。
- 方面是一个横切关注点的表示， 因此， 需求“ 注册用户在使用**SafeHomeAssured.com**之前必须先进行验证”的设计表示B*是SafeHome 网站应用的一个方面。

重构

- Fowler[FOW99]这样定义重构：
  - “重构是使用这样一种方式改变软件系统的过程：不改变代码[设计]的外部行为而是改进其内部结构。”
- 当重构软件时，检查现有设计的：
  - 冗余性
  - 没有使用的设计元素
  - 低效的或不必要的算法
  - 拙劣的或不恰当的数据结构
  - 以及其他不足，并通过修改获得更好的设计

面向对象的设计概念

- 设计类
  - 实体类
  - 边界类
  - 控制类
- 继承——超类的属性的任何改变都可以立即被所有子类继承
- 消息——刺激接收对象产生某种行为
- 多态——这种特性可显著减少扩展已存在的设计所需工作量

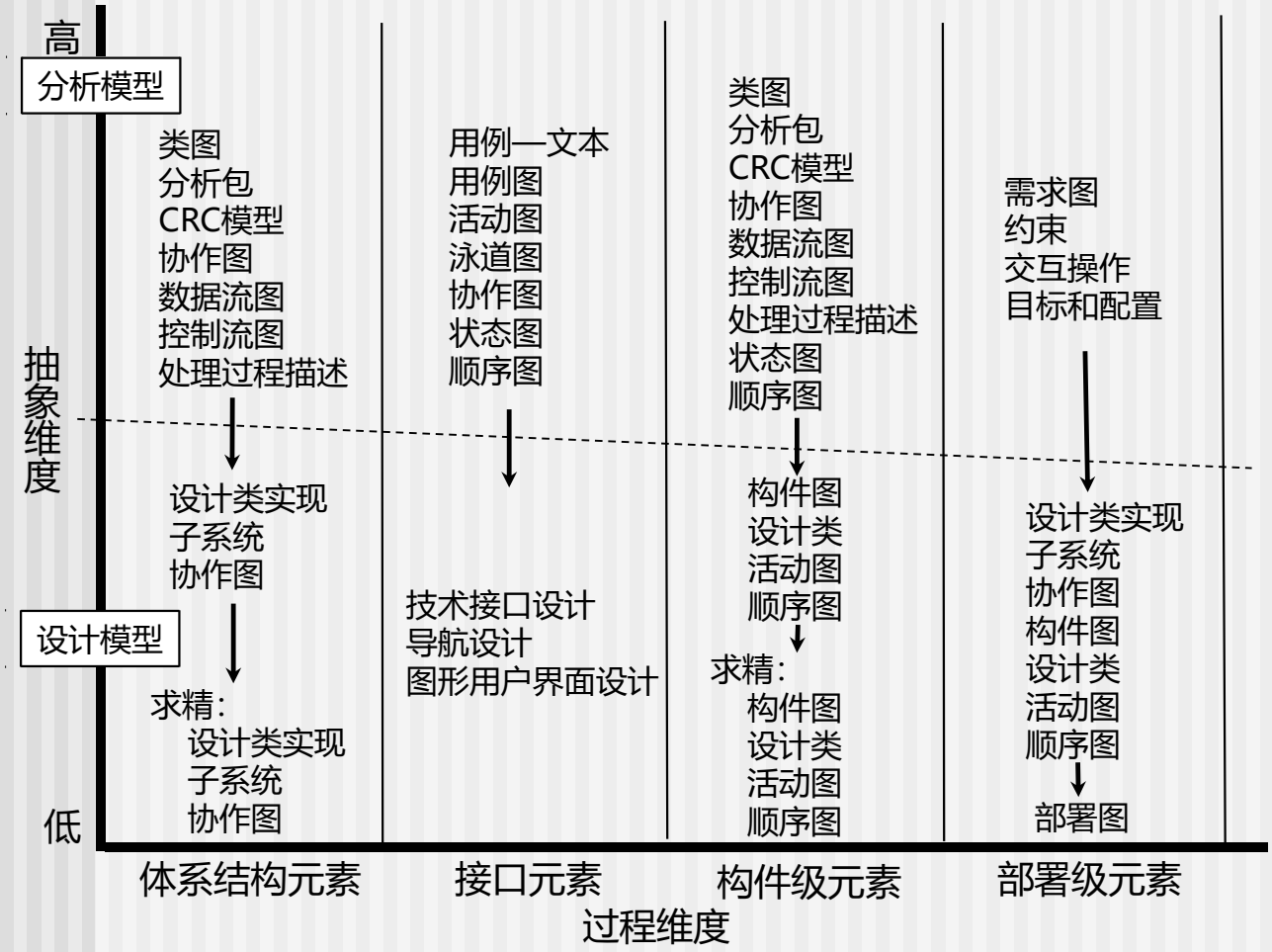
设计类

- 分析类在设计中会被精化为实体类
- 边界类 在设计中被用于创建用户在实用软件时所观看并交互的接口（例如交互窗口或打印机接口等）
  - 边界类在设计中用于管理实体对象代表用户的方式
- 控制类 用于管理
  - 实体对象的创建或更新;
  - 从实体对象获得信息后的边界对象实例化;
  - 各对象间的复杂交流;
  - 对象间或用户与应用间数据交流的确定性

设计类特征

- 完整性：封装所有必要的属性和方法
- 充分性：只包含那些“对实现该类的目的足够”的方法
- 原始性： 一个方法只集中实现该类的某一个服务
- 高内聚性：小的、集中的，专一类
- 低耦合性：类聚合保持最小范围

4.设计模型


高
分析模型
类图
分析包
CRC模型
协作图
数据流图
控制流图
处理过程描述
用例—文本
用例图
活动图
泳道图
协作图
状态图
顺序图
类图
分析包
CRC模型
协作图
数据流图
控制流图
处理过程描述
状态图
顺序图
需求图
约束
交互操作
目标和配置
抽象维度
设计类实现
子系统
协作图
设计模型
求精:
设计类实现
子系统
协作图
技术接口设计
导航设计
图形用户界面设计
构件图
设计类
活动图
顺序图
求精:
构件图
设计类
活动图
顺序图
设计类实现
子系统
协作图
构件图
设计类
活动图
顺序图
部署图
低
体系结构元素
接口元素
构件级元素
部署级元素
过程维度

设计模型元素

- 数据设计元素
  - 数据模型 --> 数据结构
  - 数据模型 --> 数据库架构
- 体系结构设计元素
  - 应用域
  - 分析类与它们之间的关系，协作和表现，并转化为设计实现
  - 模式和“风格” (第9章和第12章)
- 接口设计元素
  - 用户界面（the user interface, UI）
  - 和其他系统、设备、网络或其他信息生成者或使用者的外部接口
  - 各种设计构件之间的内部接口
- 构件级设计元素
- 部署级设计元素

数据建模

- 独立检查过程中的数据对象
- 关注数据域
- 从客户层面的抽象中建立模型
- 说明数据对象间如何相互关联

什么是数据对象?

- 表示软件需要理解的几乎任何综合信息
  - 综合信息——具有许多不同特征属性
- 可以是外部实体（例如任何产生或接受信息的事物），一个对象（例如一份报告或一次展示），一次发生（例如电话铃声） 或事件（例如警报响起）一种角色（例如销售人员），一个组织单元（例如会计部门），一个地点（例如仓库）或 一种结构（例如文件）。
- 数据对象的描述包含数据对象本身及其全部属性
- 数据对象仅囊括数据本身——不包括作用于数据上的参照

数据对象和属性

数据对象中包含若干属性，表现为方面，质量，特征，或者对于对象的描述。

对象：汽车

属性：

- **制造商**
- **型号**
- **车身样式**
- **价格**
- **可选型号**

什么是关系?

- 数据对象之间以不同方式相互关联。
  - 在**人**和**车辆**之间可以建立联系是因为两者存在关联
    - 人 拥有 车
    - 人 参保后驾驶 车辆
- 拥有和参保后驾驶这两种关系定义了人与车辆之间的相关联性
- 一种关系存在多种情况
- 对象会有多种不同的方式产生关系

体系结构设计元素

- 体系结构模型[Sha96]从以下3个来源获得：
  - 关于将要构建的软件的应用域信息；
  - 特定的需求模型元素，如数据流图或分析类、现有问题中它们的关系和协作；
  - 体系结构风格（第13章）和模式（第16章）的可获得性

接口设计元素

- 接口是一组描述类的部分行为的操作，并提供了那些操作的访问方法
- 重要元素
  - 用户界面（User interface，UI）
  - 其他系统的外部接口
  - 各种设计构件之间的内部接口
- 运用UML协作图建模（在UML 1.x 称之为协作图）

接口设计元素

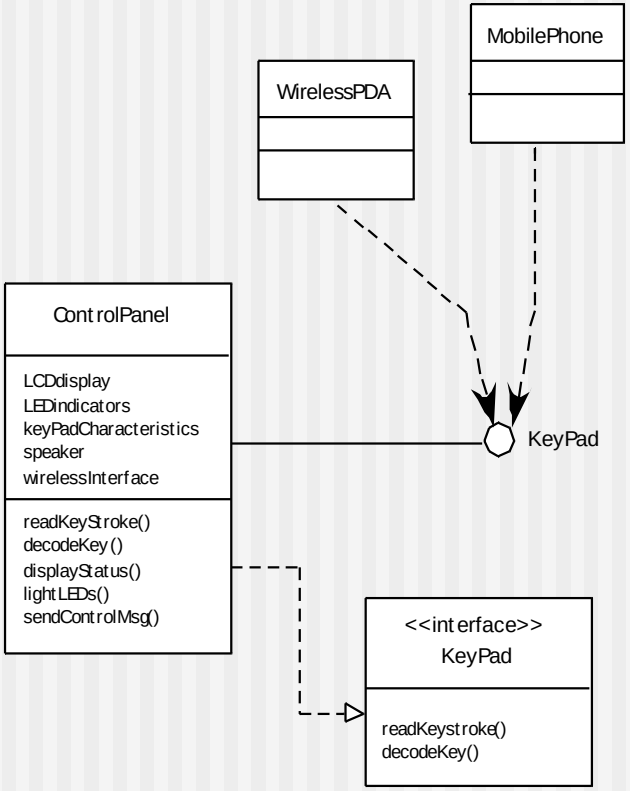
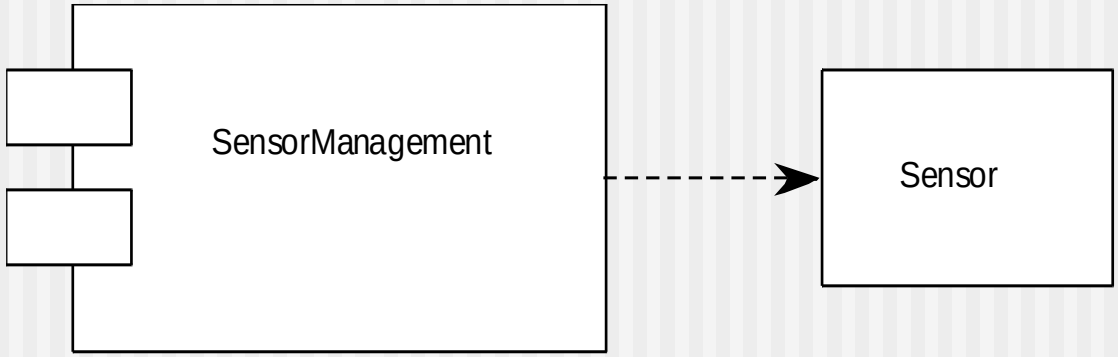


图 ControlPanel 的接口表示

构件级设计元素

- 描述了每个软件构件的内部细节
- 定义了：
  - 所有局部数据对象的数据结构
  - 所有构件内发生的处理的算法细节
  - 允许访问所有构件操作（行为）的接口
- 运用UML构件图， UML活动图，伪代码(PDL语言)及流程图建模

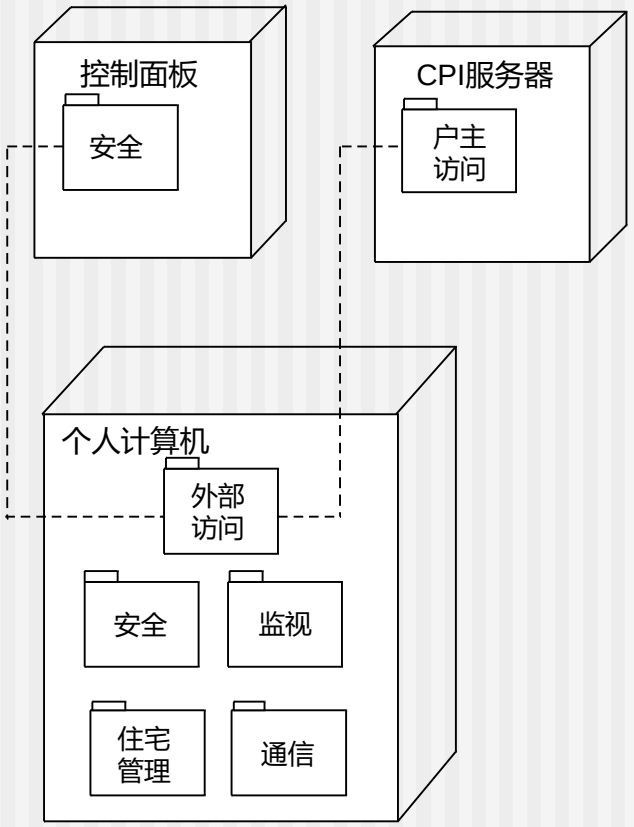
构件级设计元素



部署级设计元素

- 指明软件功能和子系统将如何在支持软件的物理计算环境内分布
- 运用UML部署图建模
- *描述符形式* 的部署图显示了计算环境，但并没有明确地说明配置细节
- *实例形式* 的部署图在后面阶段的设计中明确硬件配置细节

部署级设计元素



UML 部署图