



中国海洋大学信息学院计算机科学与技术系

软件工程

第五章 总体设计 (part1)

2020 / 11



内容提要

01 什么是总体设计

02 总体设计的基本原理

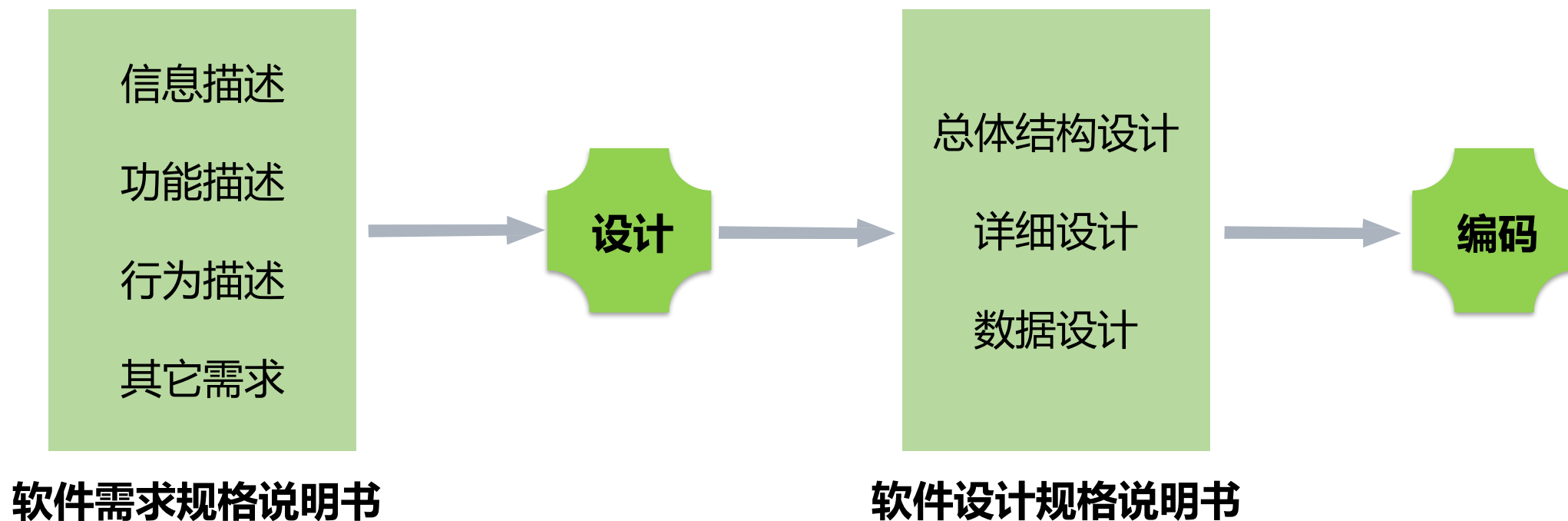
03 软件体系结构的风格

04 面向数据流的设计方法



软件的设计过程

软件设计 = 总体设计 + 详细设计 + 数据设计



软件的设计过程

软件设计 = 总体设计 + 详细设计 + 数据设计

- **总体设计**

- 设计和定义软件的整体实现框架，模块以及模块间关系
- “黑盒子”
- 高层抽象、全局角度

- **详细设计**

- 模块内部细节
- 低抽象、局部角度

- **数据设计**

- 信息描述 → 数据结构
- 数据库，数据文件(配置文件)，网络消息包...

什么是软件的总体设计



软件总体设计主要是确立软件系统的整体实现框架：
有哪些模块？每个模块的功能是什么？它们之间的相互控制关系？
注意：各个模块是一个“黑盒子”

软件的复杂性

问题：当系统的规模和复杂度不断增大的时候，构造整个系统的关键是什么？



一方面，寻找更好的方法，使系统设计更容易理解
另一方面，寻找方法可以实现复杂系统的复用

软件的复杂性

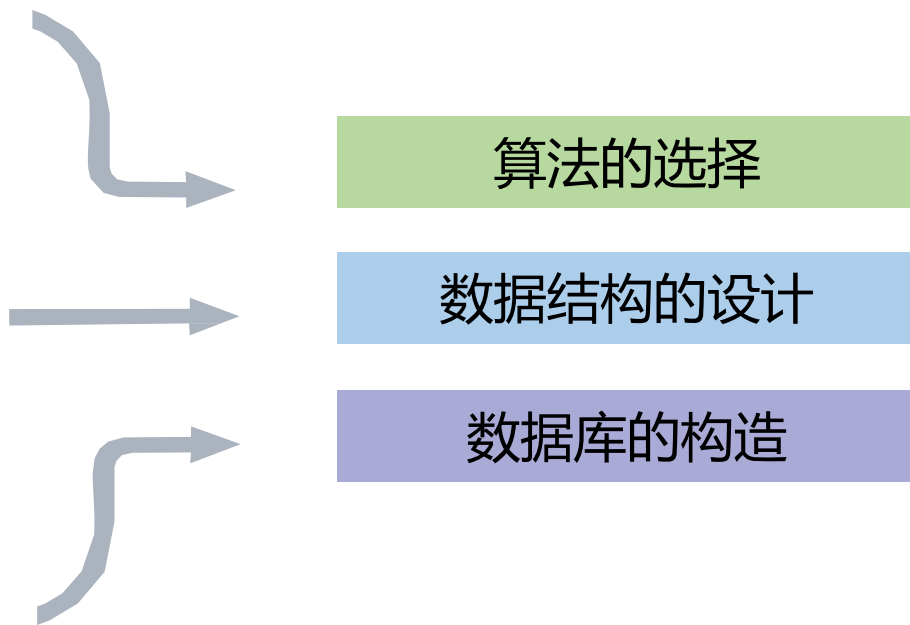
举例：要求编写一个程序，该程序读入一个文本文件，统计在该文本文件中每个英文单词出现的频率，并输出单词频率最高的100个单词

构造上述软件的重点是设计哪些部分



软件的复杂性

举例：要求编写一个程序，该程序读入一个文本文件，统计在该文本文件中每个英文单词出现的频率，并输出单词频率最高的100个单词



软件的复杂性

举例：Web信息检索是对发布在Web信息资源进行搜集、整理和组织，形成一个信息资源索引库，并通过检索界面将最符合用户要求的网站或网页提供给用户



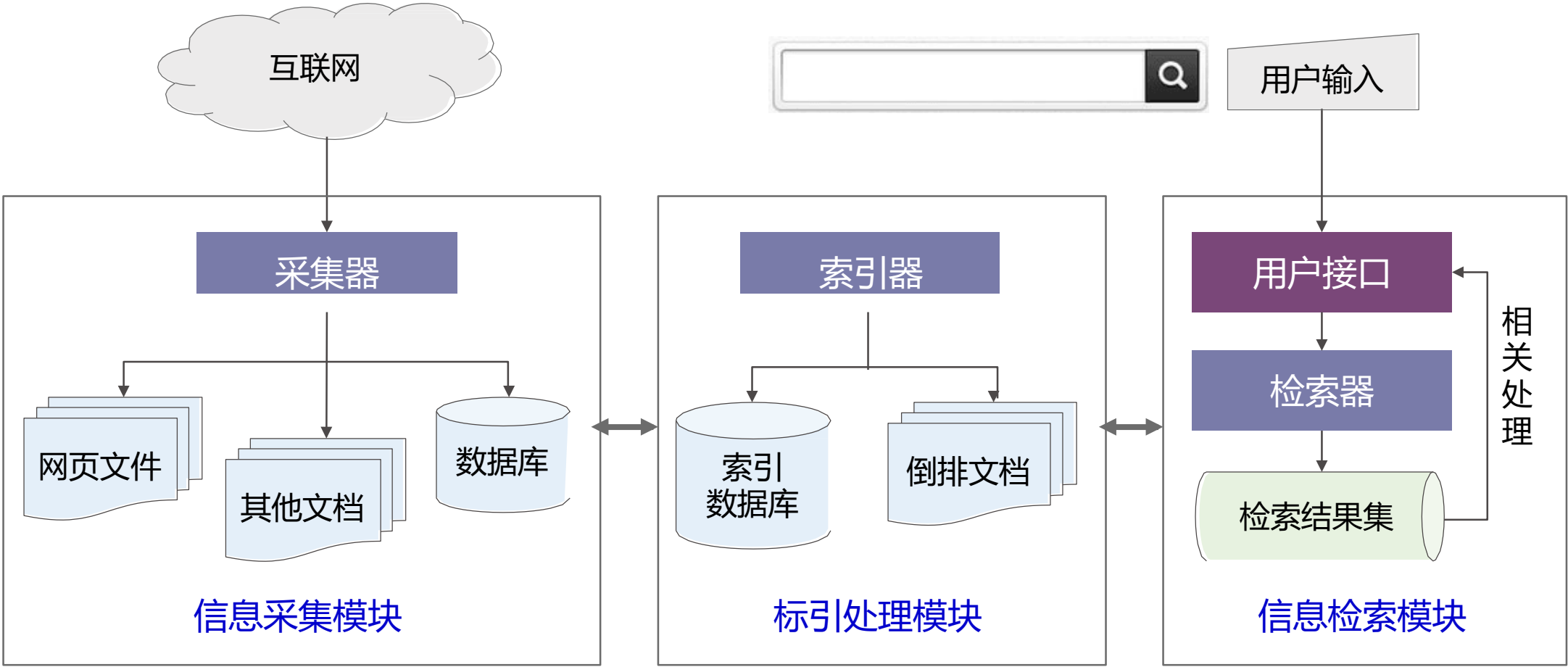
软件的复杂性

举例：Web信息检索是对发布在Web信息资源进行搜集、整理和组织，形成一个信息资源索引库，并通过检索界面将最符合用户要求的网站或网页提供给用户

Web信息检索的处理过程：

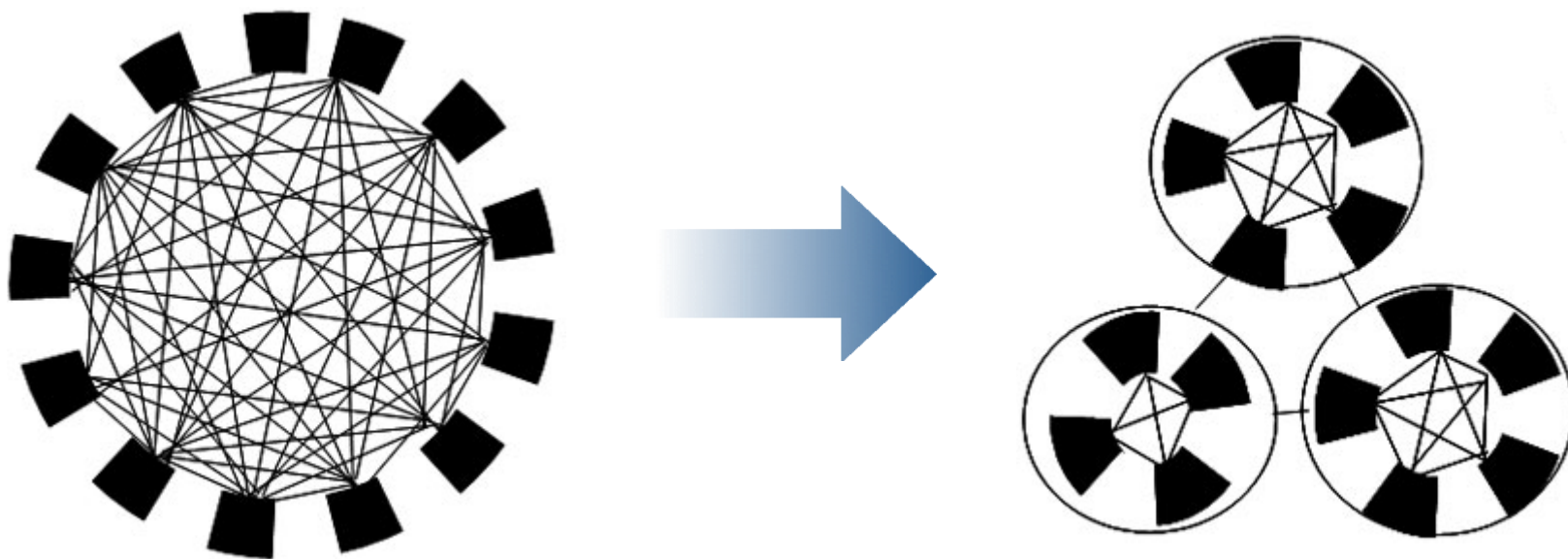


软件的复杂性



处理复杂性

随着软件系统的规模和复杂性不断增加，对系统的**全局结构设计和规划**变得比**算法的选择和数据结构的设计**明显重要得多



BOEING 787-8



- 1 Radome housing "smart" radar
- 2 Forward pressure bulkhead - composite
- 3 Two-crew cockpit featuring five 228 x 305mm Rockwell Collins LCDs, electronic flight instrument system and flight bag with dual head-up display system
- 4 Avionics bay
- 5 Forward retracting, fully steerable twin-wheel nose landing gear - Messier-Dowty
- 6 Lavatory
- 7 Cabin crew jump-seat
- 8 Flight crew access ladder
- 9 Flight crew rest area
- 10 Fuselage - single-piece composite fibre-glass carbon laminate
- 25 Aileron - Fibreglass
- 26 Main-wing box - carbon laminate
- 27 Outboard hinged panels - carbon laminate
- 28 Outboard flap - carbon laminate
- 29 Flaperon - carbon laminate
- 30 Inboard spoiler (winglet) - carbon laminate
- 31 Four wheel main landing gear - Messier-Dowty incorporating Goodrich or Messier-Bugatti electronic brakes
- 32 Main landing gear support beam
- 33 Inboard flap - fibreglass epoxy
- 34 Centre wing box - CFRP
- 35 Controlable photochromatic passenger windows - 483 x 270mm
- 36 Mid-fuselage lavatories

Main instrument panel, shroud and centre console



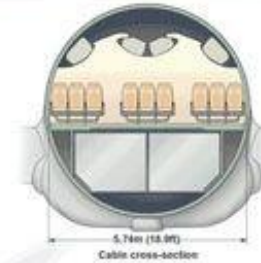
- 11 Broadband-enabled first-class cabin - 14 seats depicted
- 12 VHF 1 antenna
- 13 Forward cargo hold
- 14 Galleys and lavatories
- 15 Broadband-enabled business class cabin - 44 seats depicted 54in pitch
- 16 Cabin entry emergency/exit doors with inflatable escape slides
- 17 Fuselage circumferential frame
- 18 Landing/lift light
- 19 Rolls-Royce Trent 1000 - rated at 70,000lb thrust
- 20 General Electric GEnx turbo-fan - rated at 70,000lb thrust
- 21 Aluminium alloy leading-edge slats
- 22 Navigation light cluster
- 23 Raked wingtip - composite carbon sandwich
- 24 Static discharge wicks

- 37 Rear cargo hold
- 38 Broadband-enabled passenger cabin - 176 seats depicted 33in pitch
- 39 Cabin floor structure - composite
- 40 VHF 2 antenna
- 41 Rear fuselage - single-piece composite fibre-glass carbon laminate
- 42 Access ladder to cabin-crew rest berths
- 43 Rear lavatories
- 44 Overhead cabin crew rest cabin - six berths
- 45 Rear galleys
- 46 Rear pressure bulkhead - composite
- 47 Tailplane - carbon laminate
- 48 APU - Hamilton Sundstrand



Front elevation showing in-flight wing flex

- 49 Fin - carbon laminate
- 50 Rudder - carbon sandwich
- 51 Leading-edge - aluminium alloy
- 52 Tail cone - composite fibre-glass carbon laminate
- 53 Rear cargo door
- 54 Inboard flaps and spoilers - shown deployed
- 55 Flap fairings - composite
- 56 Main fuel tanks
- 57 Centre wing tank
- 58 Concession storable antenna
- 59 Engine pylon - aluminium alloy, steel and titanium construction
- 60 Engine cowling - carbonfibre epoxy construction



5.74m (18.8ft)
Cable cross-section

Boeing 787 family

787-3 - Length: 57m Passengers 223



787-8 - Length: 57m Passengers 223



787-9 - Length: 62m Passengers 259



787-10X - Length: 68m



Tim Bicheno-Brown
2006

www.flightglobal.com

FLIGHT
INTERNATIONAL



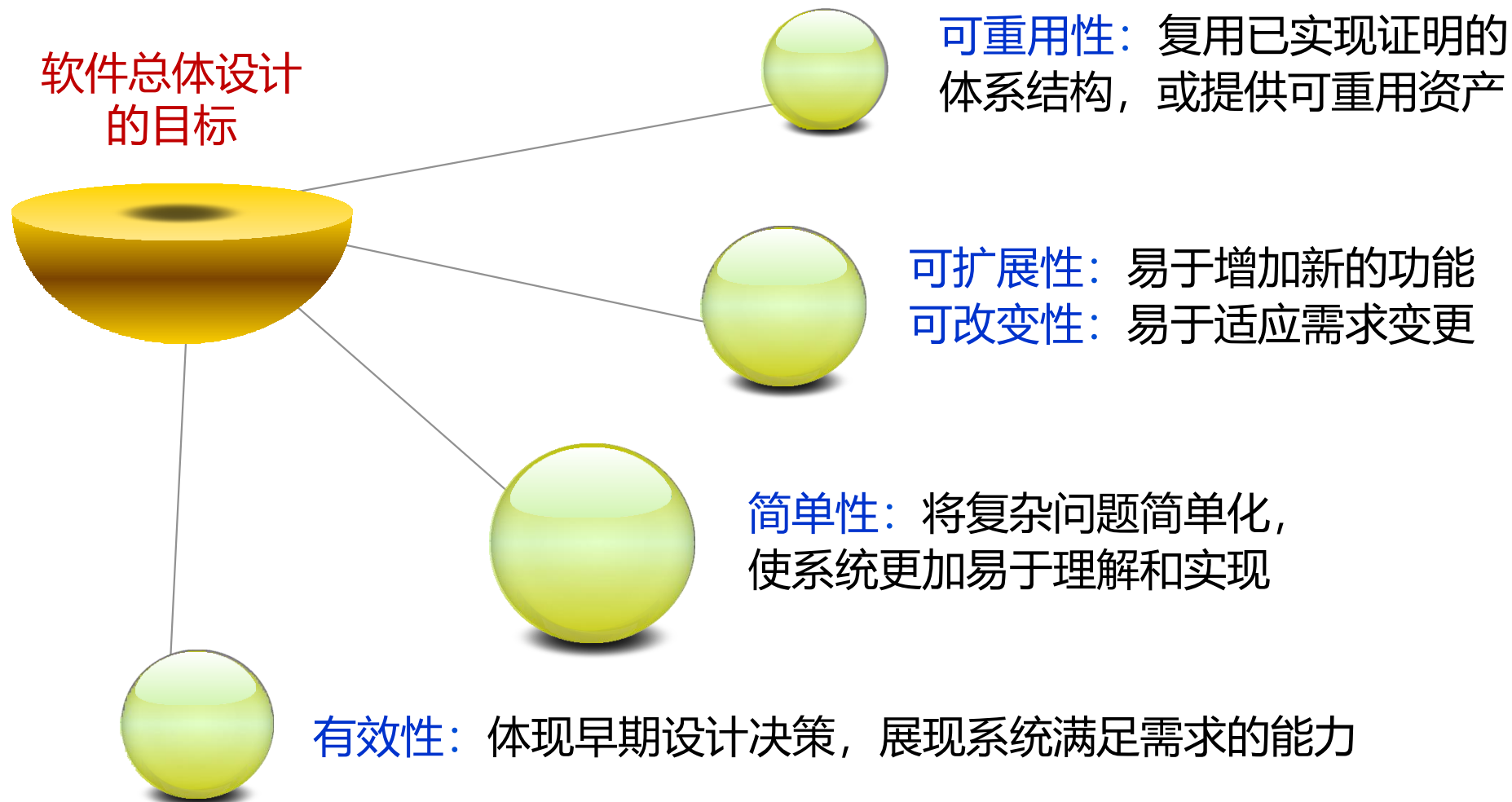
起源于建筑学的“结构”



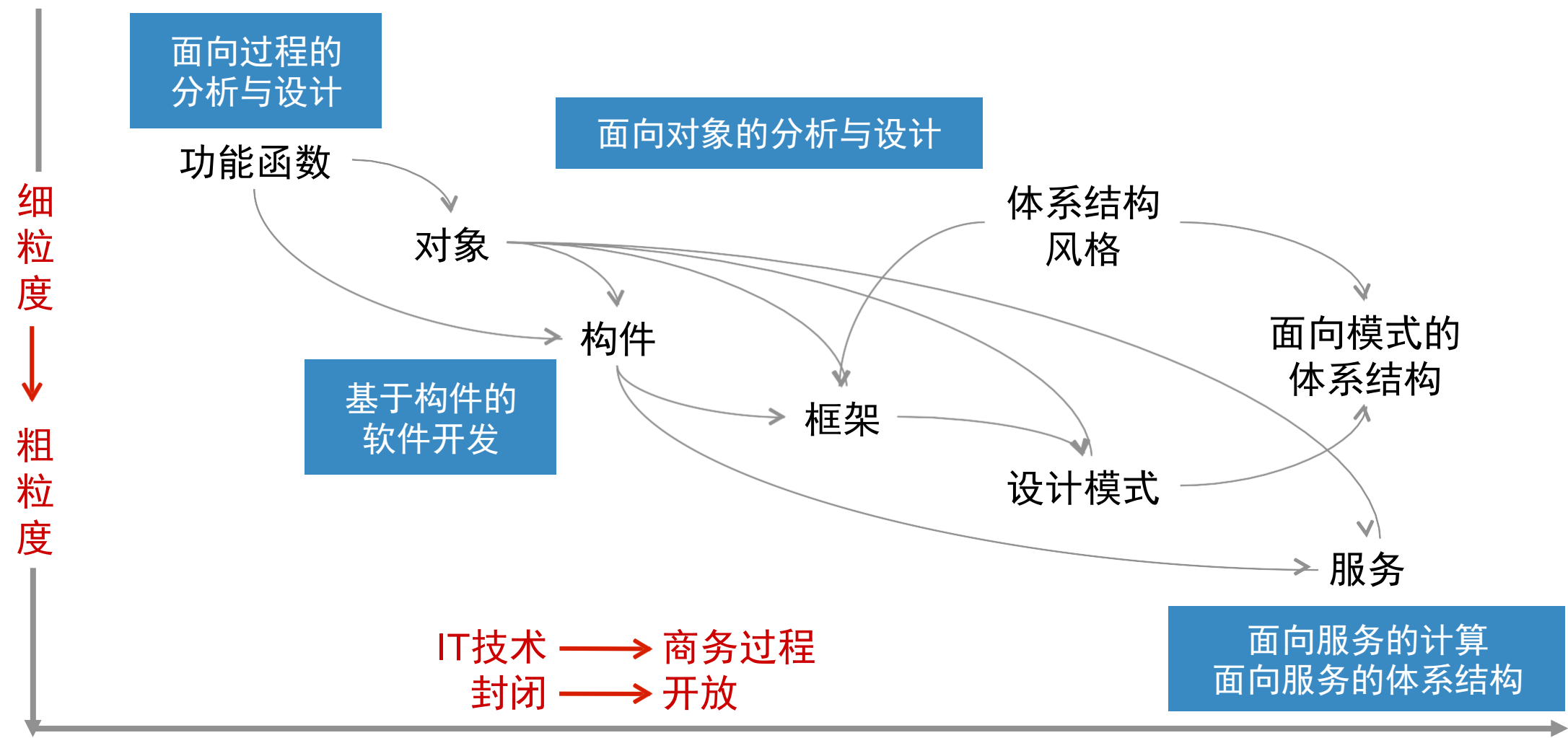
建筑设计原则：坚固、适用、赏心悦目

- 有哪些基本的建筑单元？
- 如何将这些基本单元进行组合形成整体建筑？
- 建筑单元怎样搭配才合理？
- 不同类型的建筑有什么典型的结构？
- 如何快速节省地进行建造施工？
- 怎样对建造完成的建筑进行适当的修改？
- 如何保证单元的更改不会影响整栋建筑的质量？

软件总体设计的目标



软件架构的发展



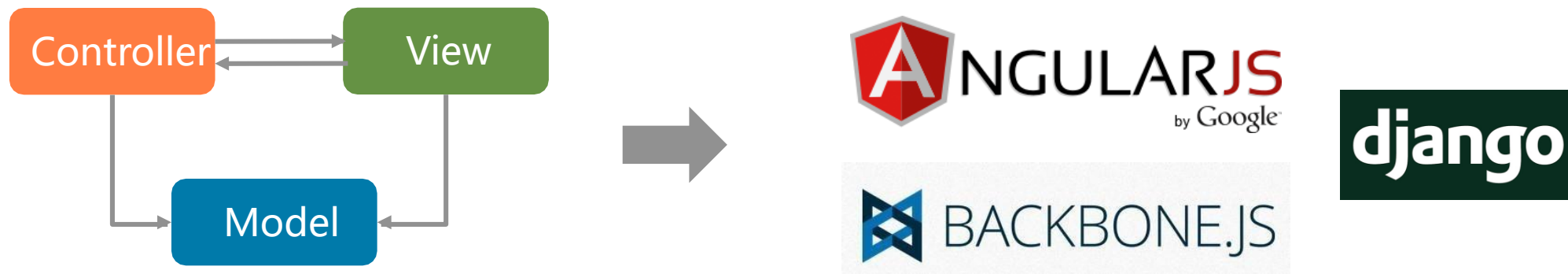
风格、模式和框架

- **设计风格：**用于描述某一特定应用领域中系统组织的惯用模式，反映了领域中众多系统所共有的结构和语义特性。
- **设计模式：**描述了软件系统设计过程中常见问题的一些解决方案，通常是从大量的成功实践中总结出来的且被广泛公认的实践和知识。
- **软件框架：**软件框架是由开发人员定制的应用系统的骨架，是整个或部分系统的可重用设计，由一组抽象构件和构件实例间的交互方式组成。

风格、模式和框架

框架和总体设计的关系：

- 总体设计的呈现形式是一个设计规约，框架则是“半成品”的软件；
- 总体设计的目的是指导软件系统的开发，框架的目的是设计复用。



框架和设计模式的关系

- **框架**给出的是整个应用的体系结构；而**设计模式**则给出了单一设计问题的解决方案，且可以在不同的应用程序或者框架中进行应用。
- 举例：一个网络游戏可以基于网易的Pomelo框架开发，这是一个基于Node.js的高性能、分布式游戏服务器框架；在实现某个动画功能时，可能会使用观察者模式实现自动化的通知更新。
- **设计模式**的目标是改善代码结构，提高程序的结构质量；**框架**强调的是设计的重用性和系统的可扩展性，以缩短开发周期，提高开发质量。

随着软件系统的规模和复杂性越来越大，（ ）变得最为重要

- ☐ A 算法的选择
- ☐ B 数据结构的设计
- ☐ C 数据库的构造
- ☒ D 系统的全局结构设计

提交

内容提要

- 01 什么是总体设计
- 02 总体设计的基本原理
- 03 软件体系结构的风格
- 04 面向数据流的设计方法



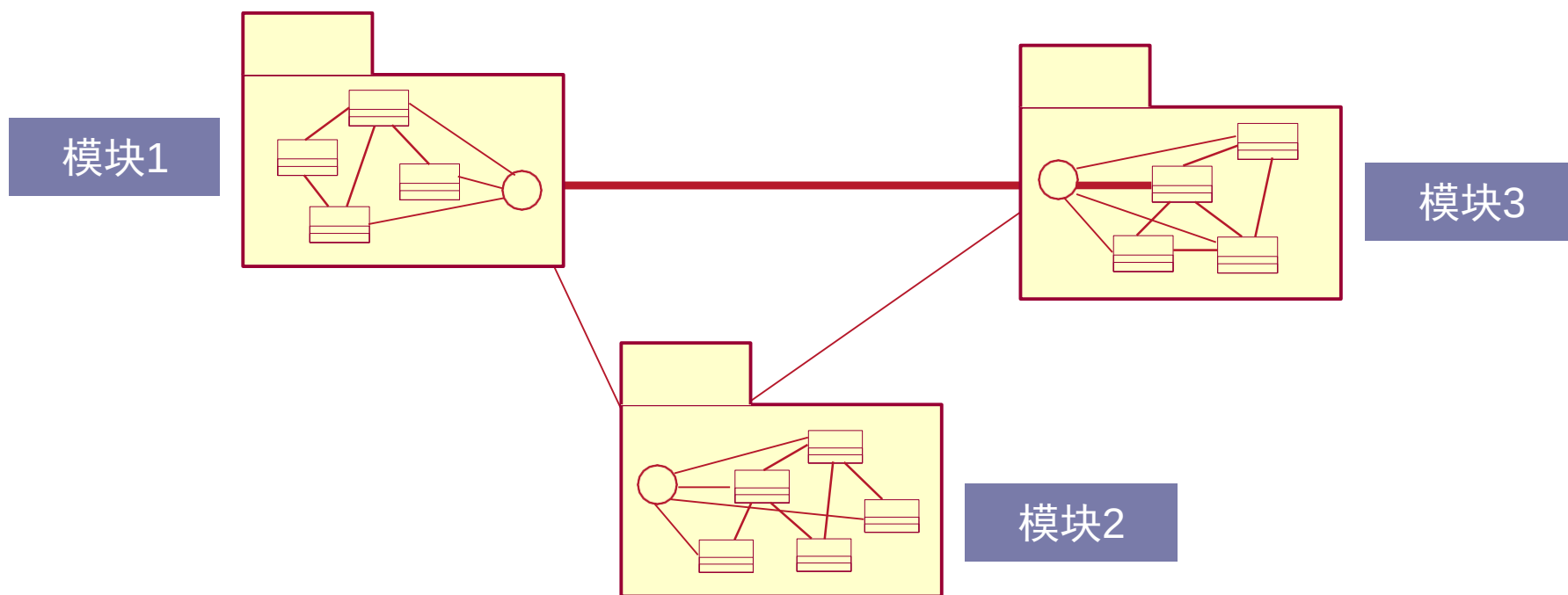
总体设计的基本原理

设计原则是系统分解和模块设计的基本标准，应用这些原则可以使代码更加灵活、易于维护和扩展



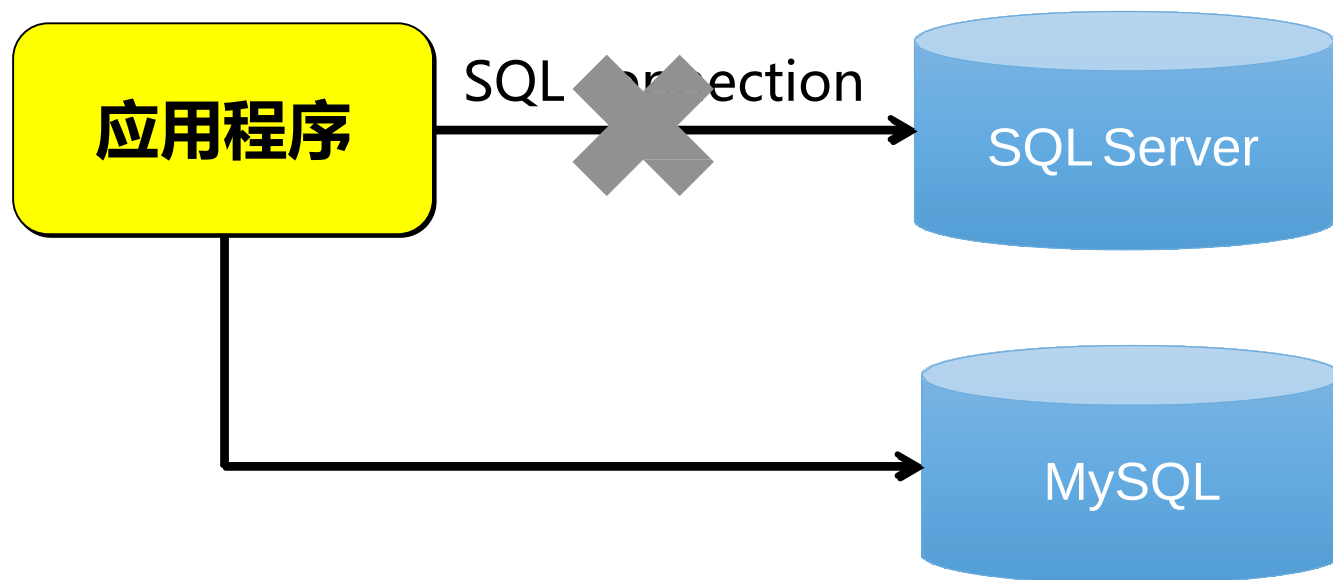
(1) 模块化

模块化是在逻辑和物理上将整个系统分解成多个更小的部分，其实质是“分而治之”，即将一个复杂问题分解成若干个简单问题，然后逐个解决



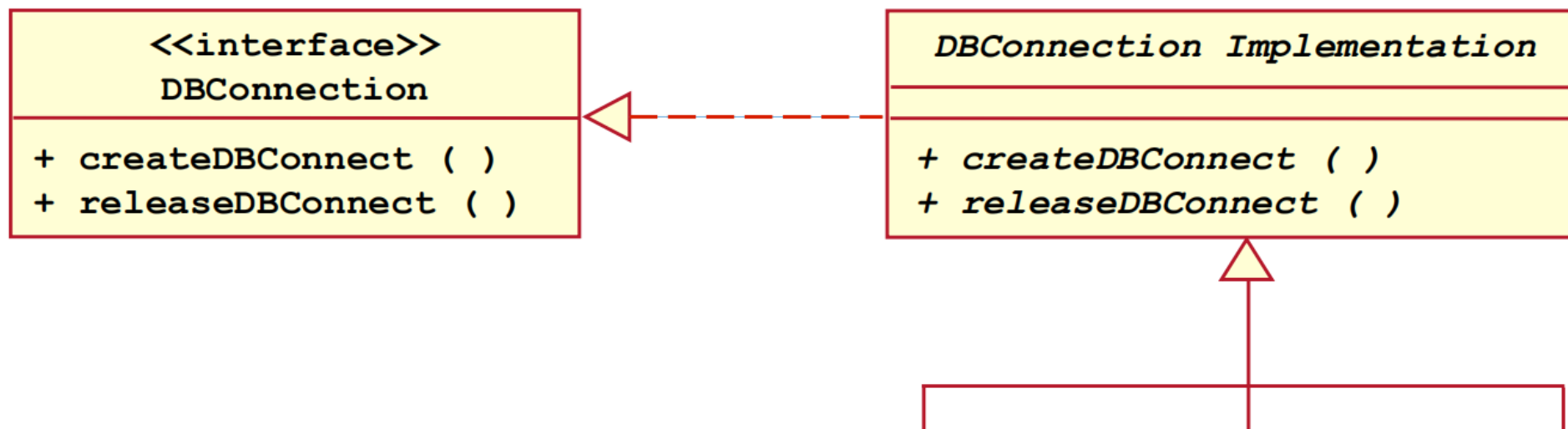
(2) 抽象

抽象是关注事物中与问题相关部分而忽略其他无关部分的一种思考方法。



(2) 抽象

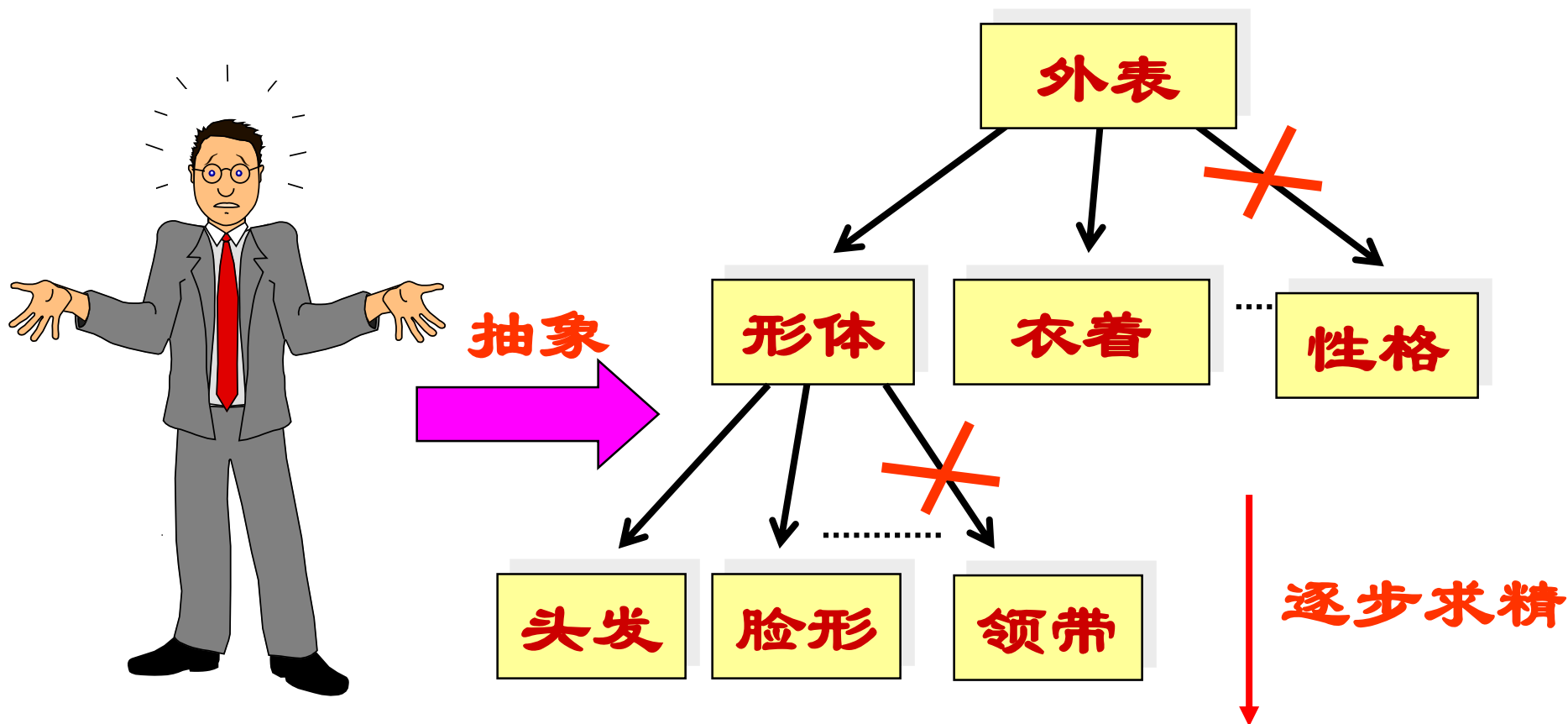
抽象是关注事物中与问题相关部分而忽略其他无关部分的一种思考方法。



抽象是认识复杂事物过程中使用的一种思维工具
可以起到降低复杂性，和增强扩展能力的作用

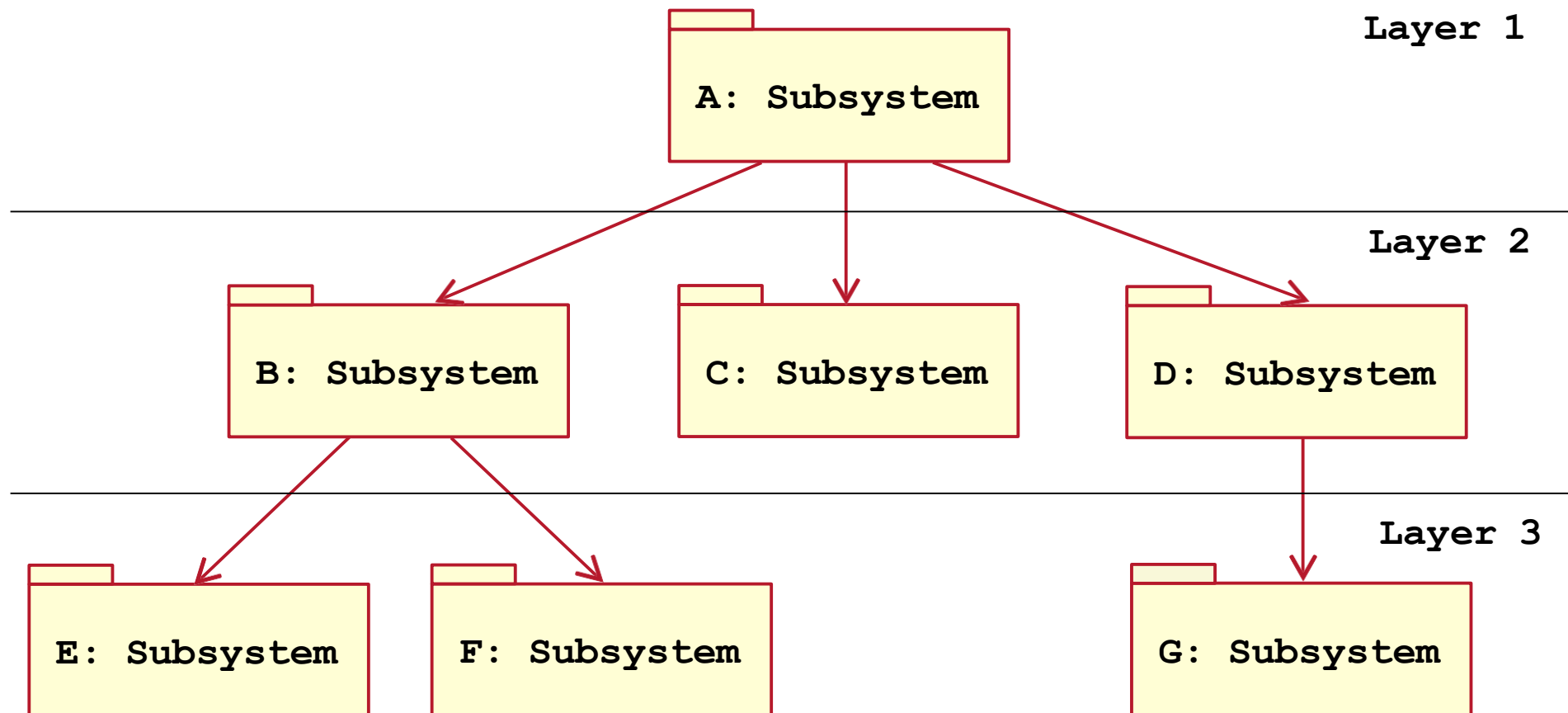
(3) 逐步求精

逐步求精的思想： 对一个事物的认识是一个从高层次抽象到向低层次抽象逐步转化和过渡的过程，首先一般性、抽象的，然后才是具体和详细的



人的认知过程遵守Miller法则：只能同时注意 7 ± 2 个知识块

(3) 逐步求精



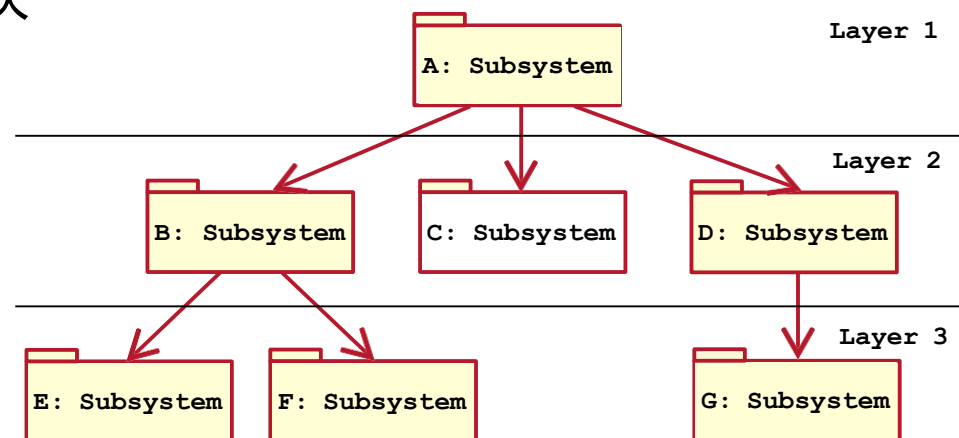
(3) 逐步求精

分层 (Layering)

- 每一层可以访问下层，不能访问上层
- 封闭式结构：每一层只能访问与其相邻的下一层
- 开放式结构：每一层还可以访问下面更低的层次
- 层次数目不应超过 7 ± 2 层

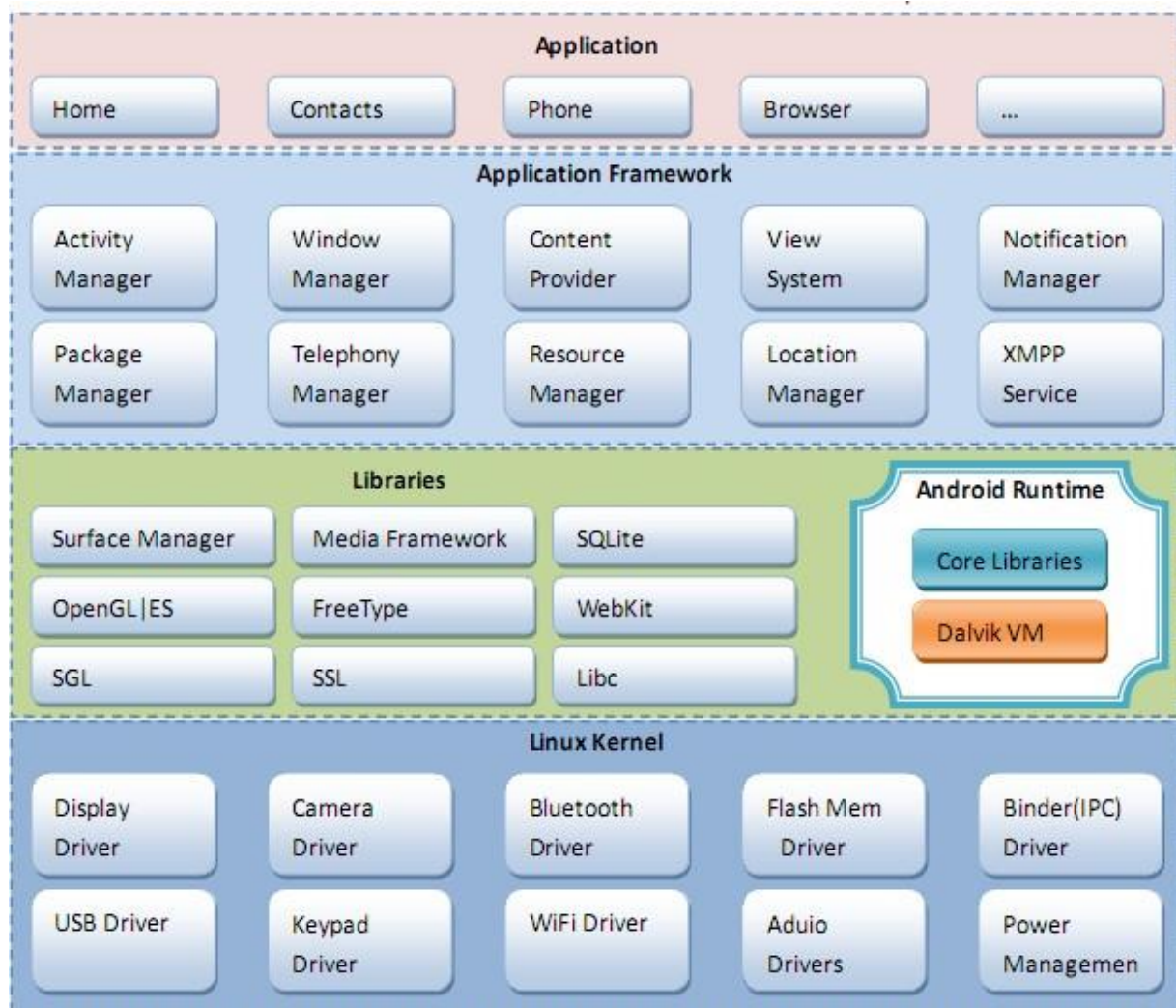
划分 (Partitioning)

- 系统被分解成相互对等的若干模块单元
- 每个模块之间依赖较少，可以独立运行



注意：模块单元增加了处理开销，过度分层或划分会增加复杂性。

示例：安卓操作系统层次结构



应用层：运行在虚拟机上的Java应用程序。

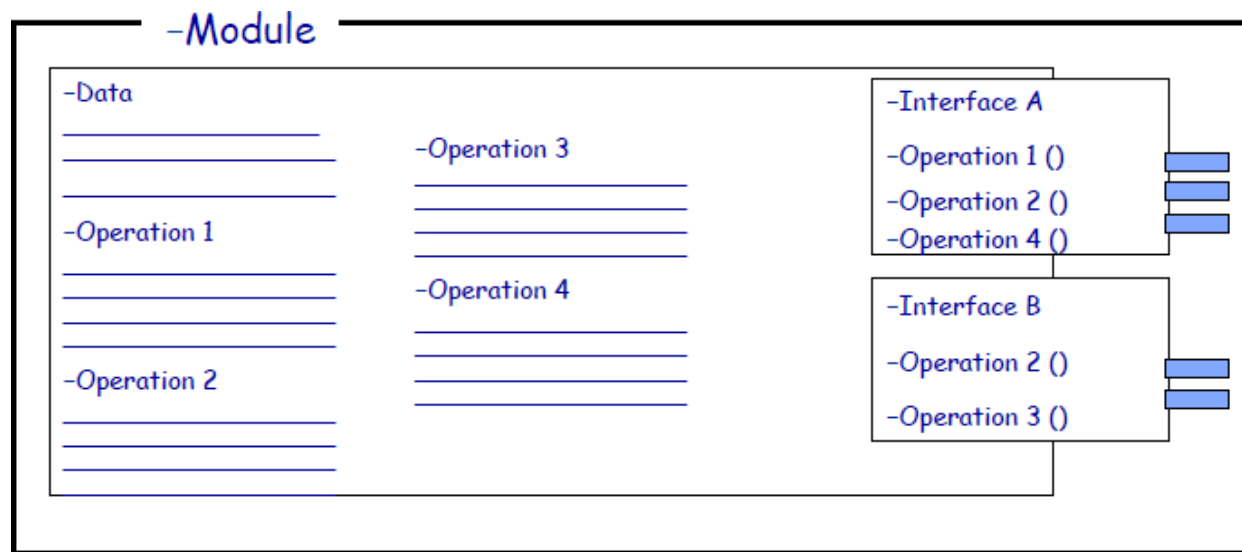
应用框架层：支持第三方开发者之间的交互，使其能够通过抽象方式访问所开发的应用程序需要的关键资源。

系统运行库层：为开发者和类似终端设备所有者提供需要的核心功能。

Linux内核层：提供启动和管理硬件以及Android应用程序的最基本的软件。

(4) 信息隐藏和局部化

信息隐藏和局部化是指每个软件单元对其他所有单元都隐藏自己的设计决策，各个单元的特性通过其外部可见的接口来描述



要求： 应将单元接口设计得尽可能简单，并将单元对于环境的假设和要求降至最低。

(5) 模块独立

模块独立的概念是模块化、抽象、信息隐藏和局部化的直接结果

模块独立：使得每个模块完成一个相对独立的特定子功能，并且和其它模块之间的关系尽量简单，其重要性表现为：

模块化的软件容易开发

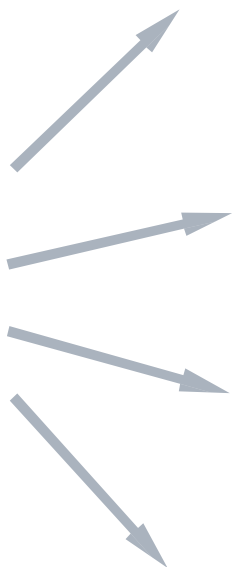
独立的模块容易测试和维护

模块独立程度可以由两个定性标准度量

耦合：衡量不同模块彼此间互相依赖的紧密程度

内聚：衡量模块内部各个元素彼此结合的紧密程度

耦合的种类



数据耦合

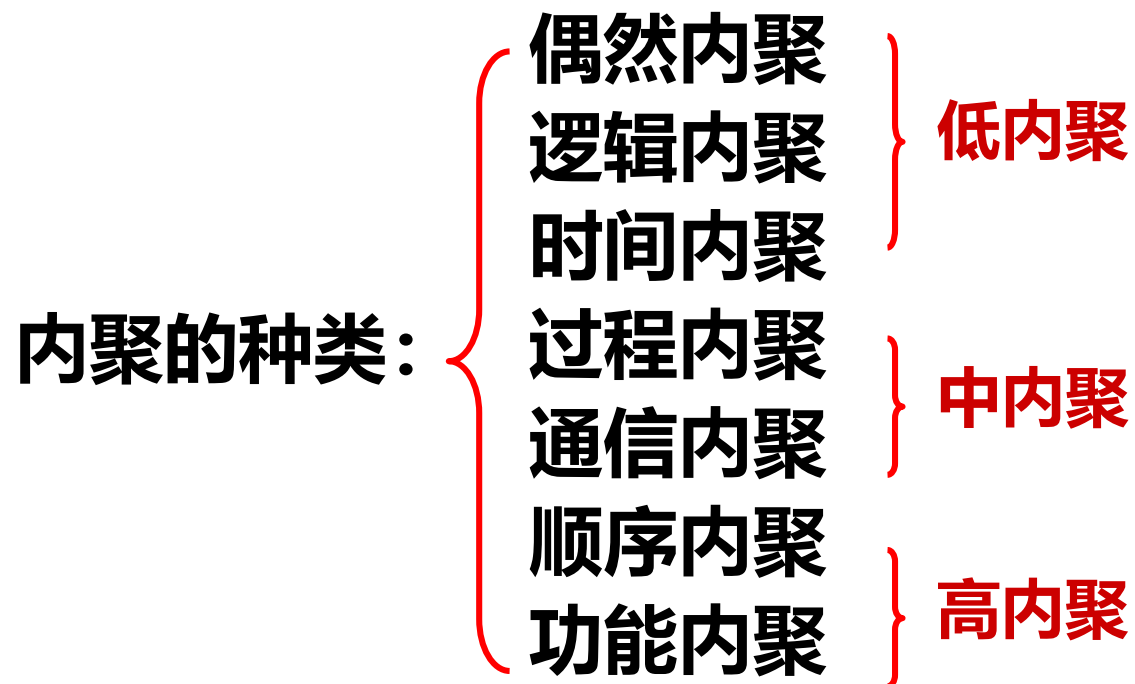
控制耦合

公共环境耦合

内容耦合

**尽量使用数据耦合，少用控制耦合，限制公共环境耦合的范围，
完全不用内容耦合**

内聚



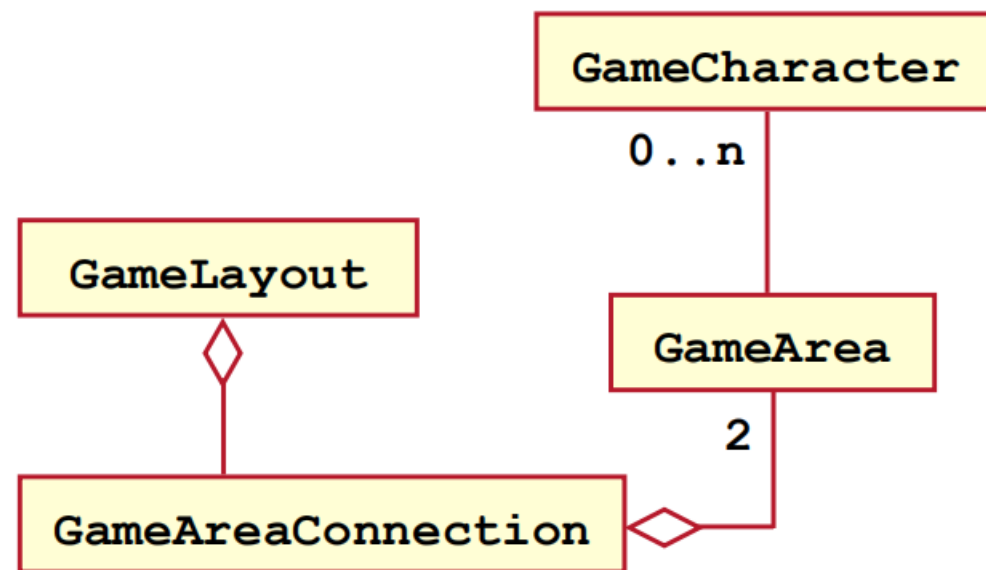
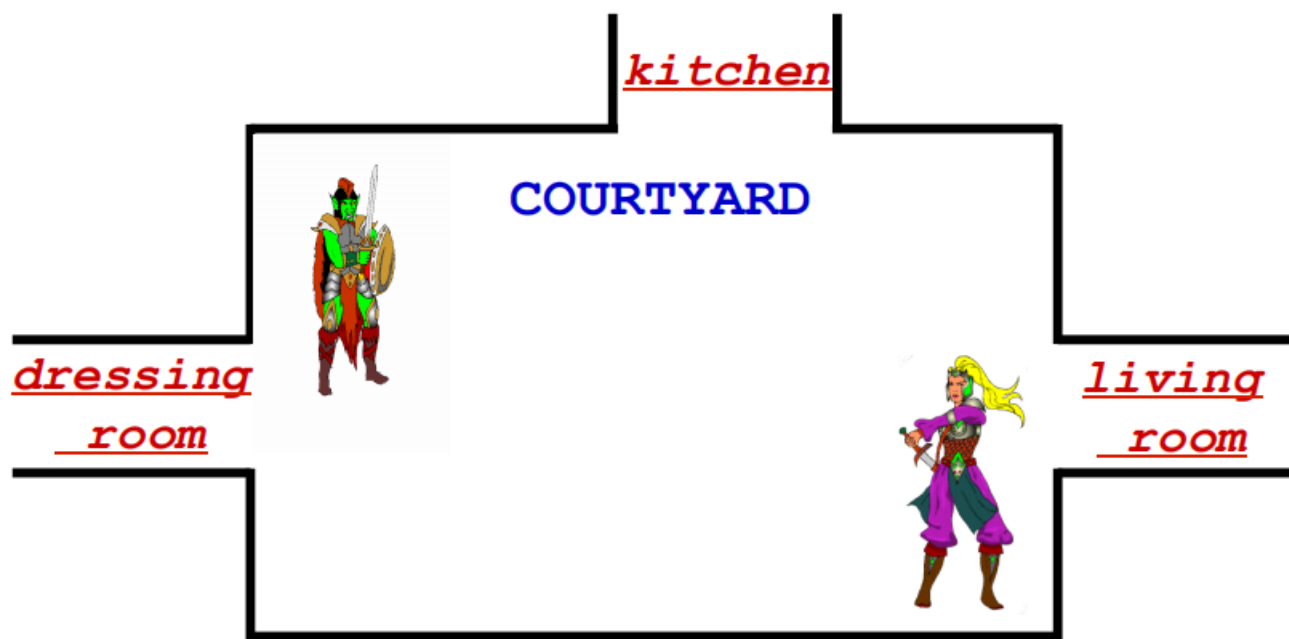
内聚

- **偶然性内聚:** 模块内各成分为完成一组功能而结合在一起，关系松散
- **逻辑性内聚:** 模块完成的诸任务逻辑上相关
- **时间性内聚:** 模块内诸任务必须在同一时间段内执行
- **过程性内聚:** 模块内各成分相关且必须按特定次序执行
- **通讯性内聚:** 模块内各成分对数据结构的同一区域操作
- **顺序性内聚:** 模块内各成分与同一功能相关且顺序执行
- **功能性内聚:** 模块内各成分是一整体，完成单个功能

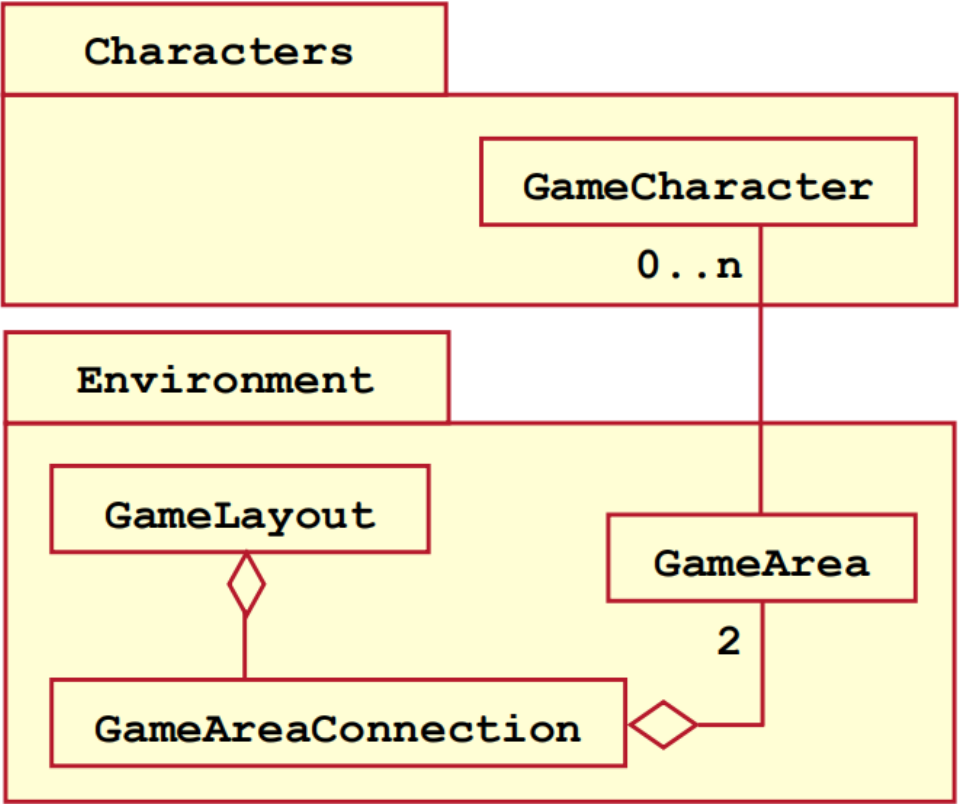
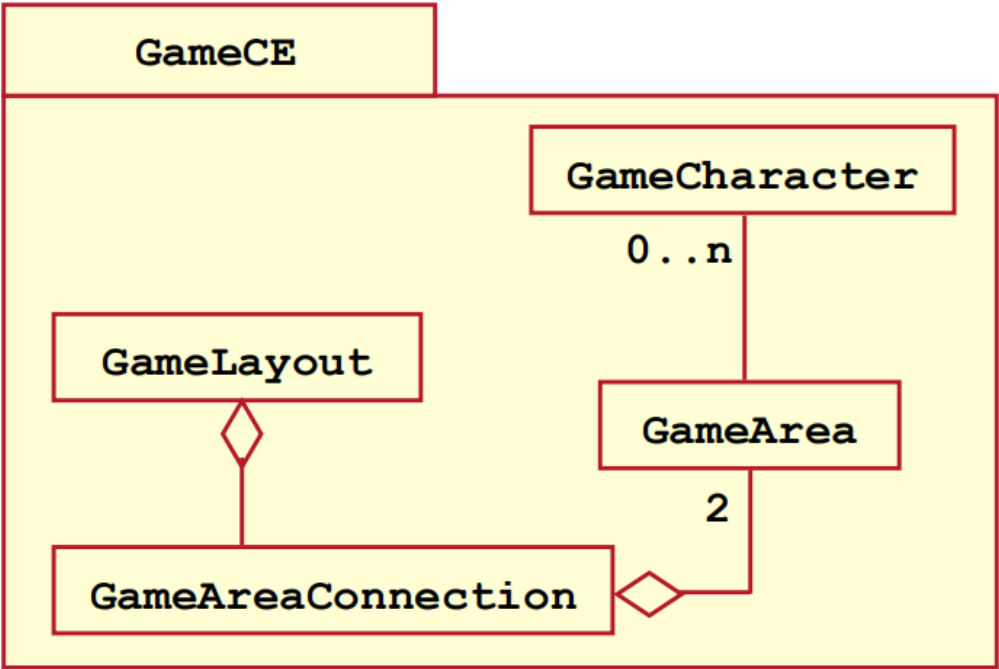
高内聚，低耦合

模块设计的原则

举例：某游戏软件的部分类图如下，其中GameCharacter代表人物，GameLayout代表布局，GameArea 代表一个区域，GameAreaConnection 代表两个区域的连接。

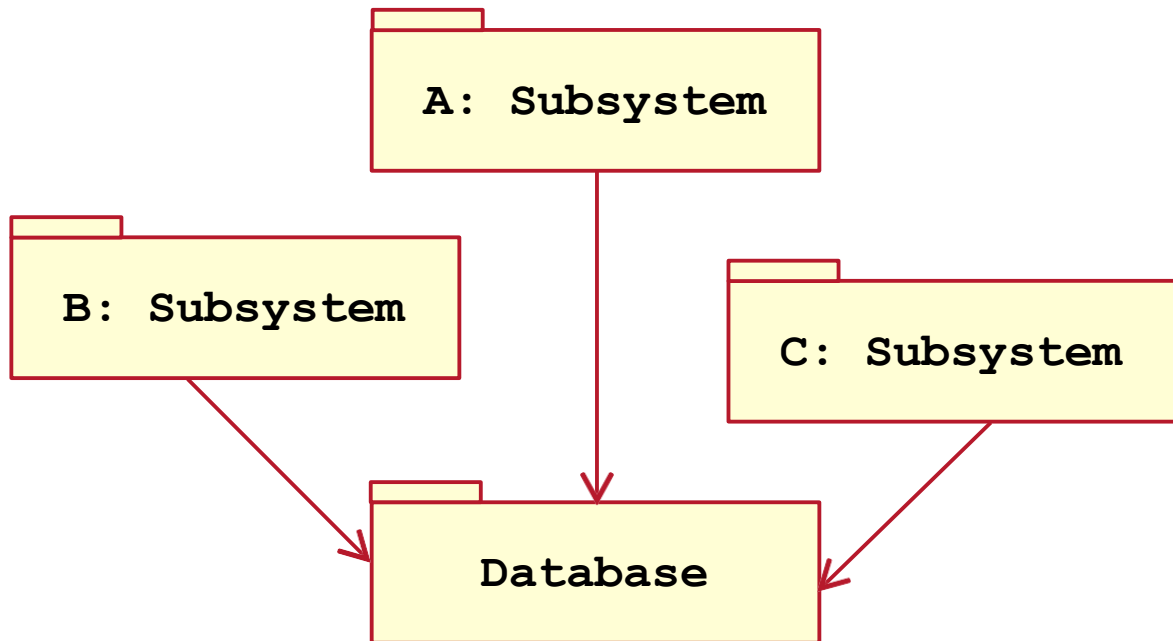


模块设计的原则



模块设计的原则

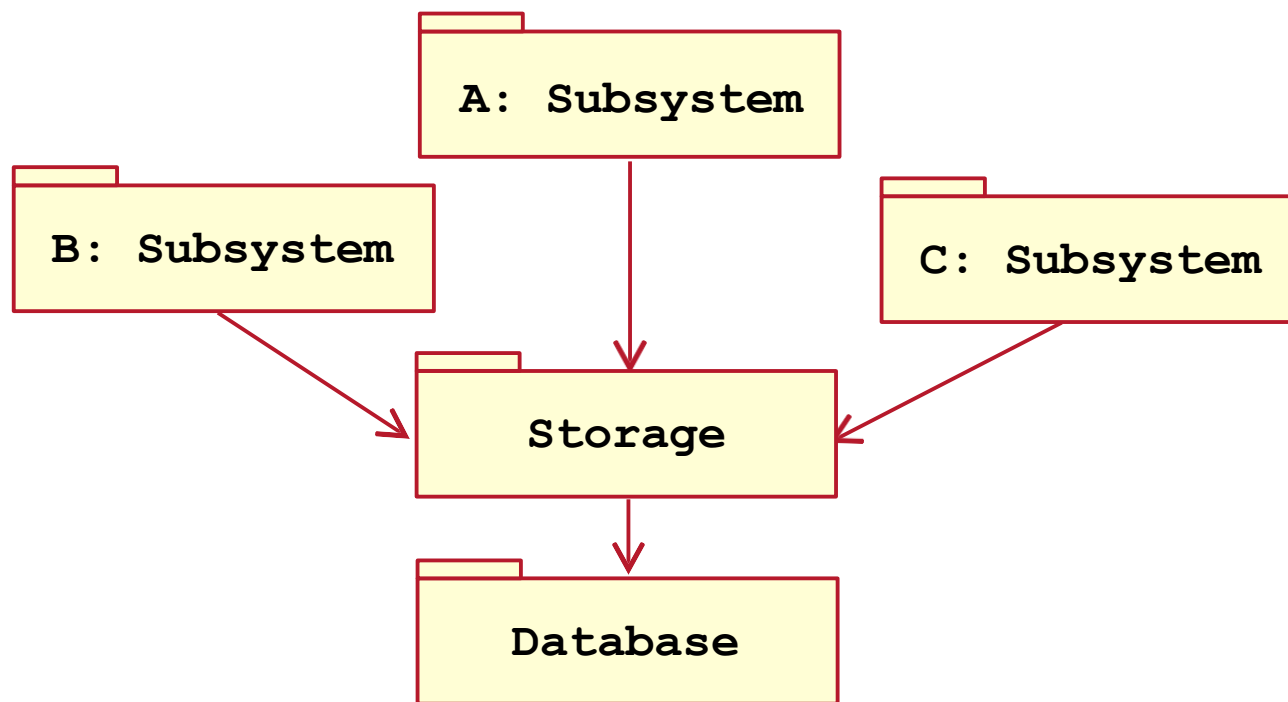
举例：在一个软件系统中，有三个子系统A、B、C都要访问一个关系数据库。



这个方案有什么问题？

模块设计的原则

改进： 在子系统和数据库之间增加一个存储子系统Storage，从而屏蔽了底层数据库的变化对上层子系统的影响。



一个设计良好的系统应具有（ ）的特征

- ☐ A 低内聚，低耦合
- ☒ B 高内聚，低耦合
- ☐ C 高内聚，高耦合
- ☐ D 低内聚，高耦合

提交

聆听

总结

分享

改进