软件工程与实践

软件工程与实践课程组

电子科技大学信息与软件工程学院

❑第四章 软件设计

- ✓第一部分 软件设计基础
- ✓第二部分 软件体系结构设计
- ✓第三部分 软件详细设计

软件设计基础

1. 何为软件设计

✓设计的对象、任务和过程以及产生的设计元素

2. 软件设计原则

✓抽象、求精、模块、隐藏、多视点、分离

3. 软件设计方法

✓结构化设计方法

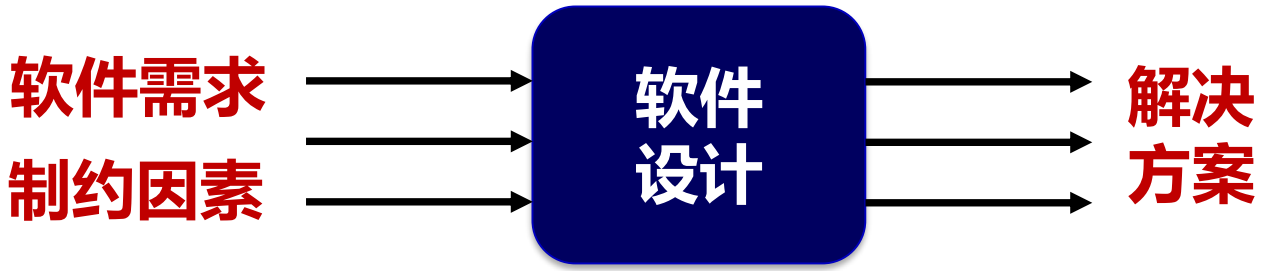
✓面向对象设计方法

4. 软件设计输出及评审

✓软件设计软件制品、软件设计缺陷及评审要求

1.1 何为软件设计?

- **软件设计**
  - ✓针对**软件需求**，综合考虑各种**制约因素**，探究软件实现的**解决方案**
- **设计前提：软件需求**
  - ✓定义了要做什么样的软件
- **设计考虑：制约因素**
  - ✓**资源**：时间、人力、财力、开发辅助工具
  - ✓**技术**：技术平台，如DBMS还是文件系统

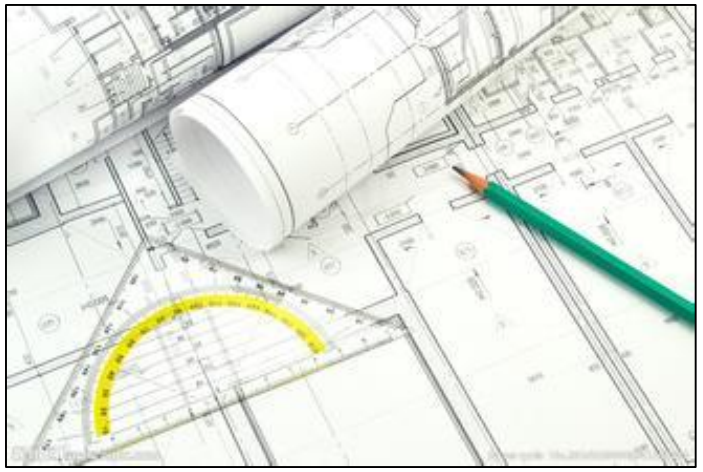


软件设计是要给出软件需求的实现解决方案

何为软件系统的解决方案?

- **描述了如何来构造和实现软件系统**
  - ✓模块的组织
  - ✓模块的功能和接口
  - ✓模块间的交互
  - ✓模块内部的算法
  - ✓人机交互的界面和方式
  - ✓数据结构设计
  - ✓数据库的设计和组织 ……

- **不同设计内容**
- **不同设计层次**
- **不同设计视角**



- **软件系统的解决方案类似于软件实现图纸**
- **从实现的角度，软件设计方案应该是什么样的?**

从需求到设计和编码

- **需求 → 设计**
  - ✓回答**如何做（How）→ 设计图纸**
  - ✓根据需求来进行设计，确保设计的**质量**
- **设计 → 实现**
  - ✓基于设计来指导**施工和实现**
  - ✓设计的好坏直接决定了最终产品的好坏！

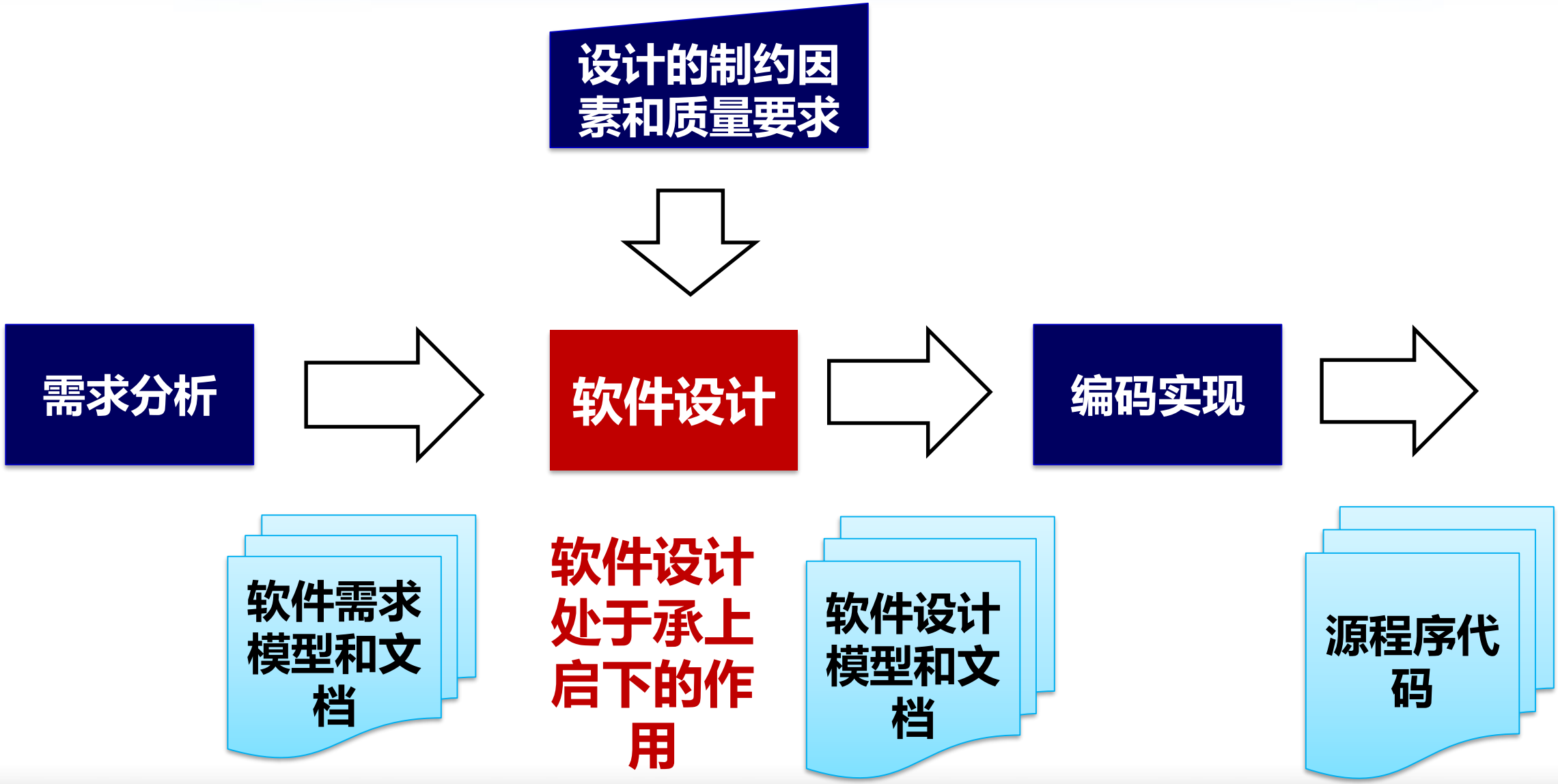
软件设计关注于软件需求的实现问题

软件需求

软件设计

编码实现

1.2 需求分析、软件设计、软件实现间的关系



软件设计是需求分析和软件实现间的桥梁

软件需求

要做什么，明确问题和目标

软件设计

如何做，绘制图纸

软件代码

做出来，开展施工

思考和讨论

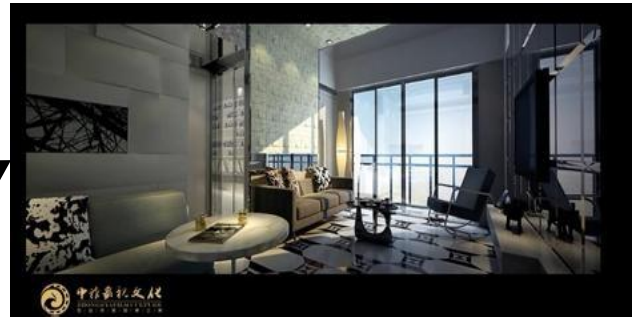
- 直接根据软件需求来编写代码行吗？为什么？

1.3 设计的多样性和差异性：质量

- ✓ 如何区分设计的差异性?
- ✓ 如何评价设计的优劣?
- ✓ 除了满足需求之外，设计还需要注意哪些要素?

用户需求

软件设计



设计结果1

设计结果2

设计结果n

- ✓ 软件设计是一个创作的过程
- ✓ 一个软件需求会有多种软件设计方案

软件设计的质量要求

- **正确性**
  - ✓ 正确实现所有的软件需求项；设计元素间无逻辑冲突；在技术平台和软件项目的可用资源约束条件下，采用程序设计语言可完整地实现设计模型
- **充分性**
  - ✓ 所有的设计元素已充分细化，模型易于理解，编程人员勿需再面对影响软件功能和质量的技术抉择或权衡
- **优化性**
  - ✓ 以合理的、充分优化的方式实现软件需求模型，目标软件产品能够表现出良好的软件质量属性，尤其是正确性、有效性、可靠性和可修改性
- **简单性**
  - ✓ 模型中的模块的功能或职责尽可能简明易懂，模块间的关系简单直观，模型的结构尽可能自然地反映待解软件问题的结构

高质量软件设计的特点

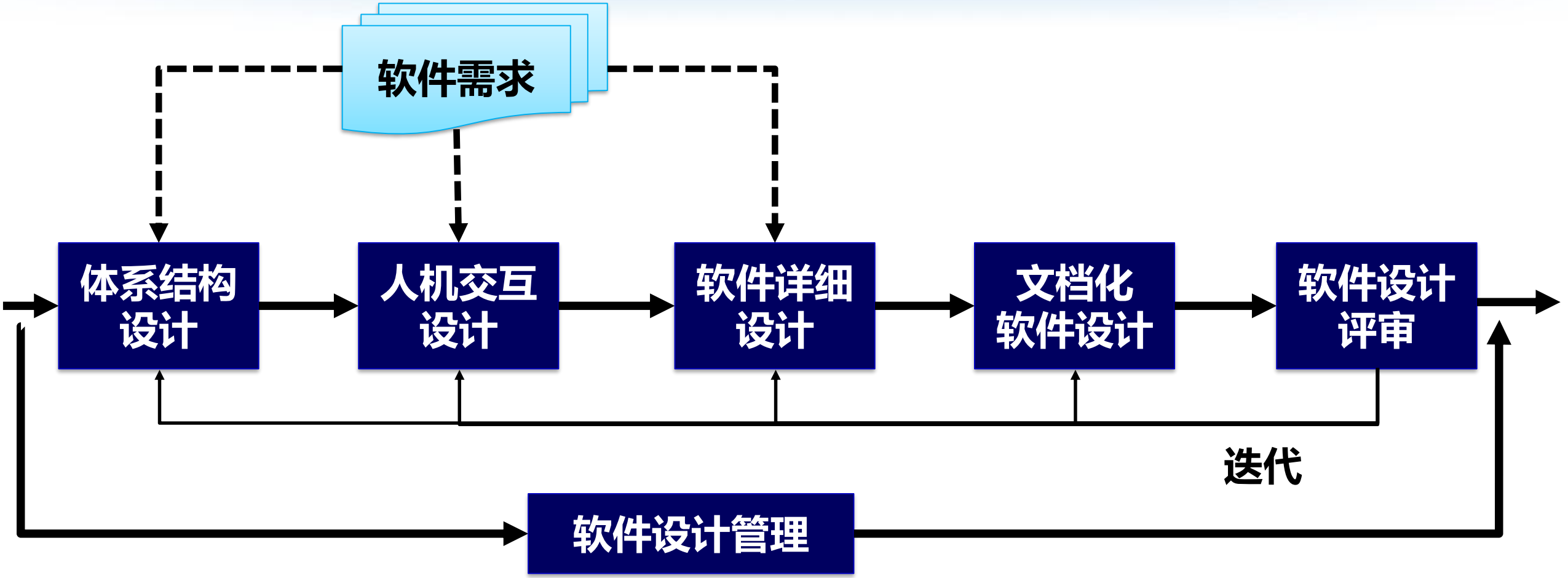
- 正确性
- 可靠性
- 可维护性
- 可重用性
- 可追踪性
- 可移植性
- 可互操作性
- 有效性
- 安全性 ……

- 设计不仅要满足需求，还要有好的质量！
- 要从多个利益相关者的角度来理解设计的“质量”
  - ✓ 用户、开发者、维护人员等
- 设计要内外兼修
  - ✓ 内在质量和外在质量

思考和讨论

- 为什么要关注设计的质量?
- 如果软件设计质量不高，会带来什么样的问题?
- 如何来确保软件设计的质量?

1.4 软件设计过程

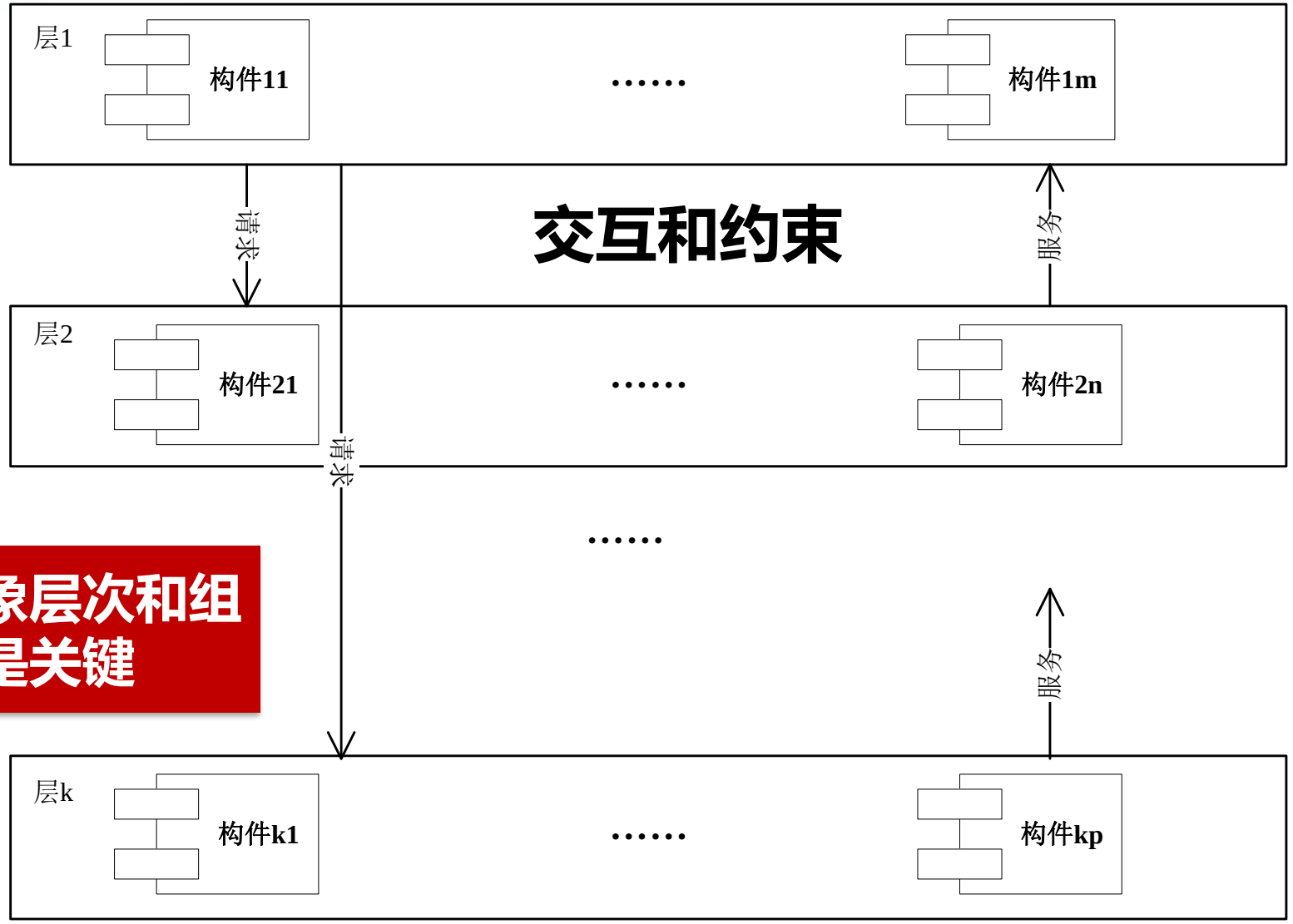


软件设计过程 – 软件体系结构设计

- **从全局和宏观视角、站在最高抽象层次来设计软件系统**
  - ✓构成要素及其关系
  - ✓职责分派、接口定义
  - ✓相互交互及协作行为
- **每个模块为“黑盒子”**
- **设计关注的质量要素**
  - ✓可扩展、可维护、可重用、可移植、可互操作等等

- **要素：函数、方法、类、程序包**
- **关系：依赖、交互**
- **区别：粒度**

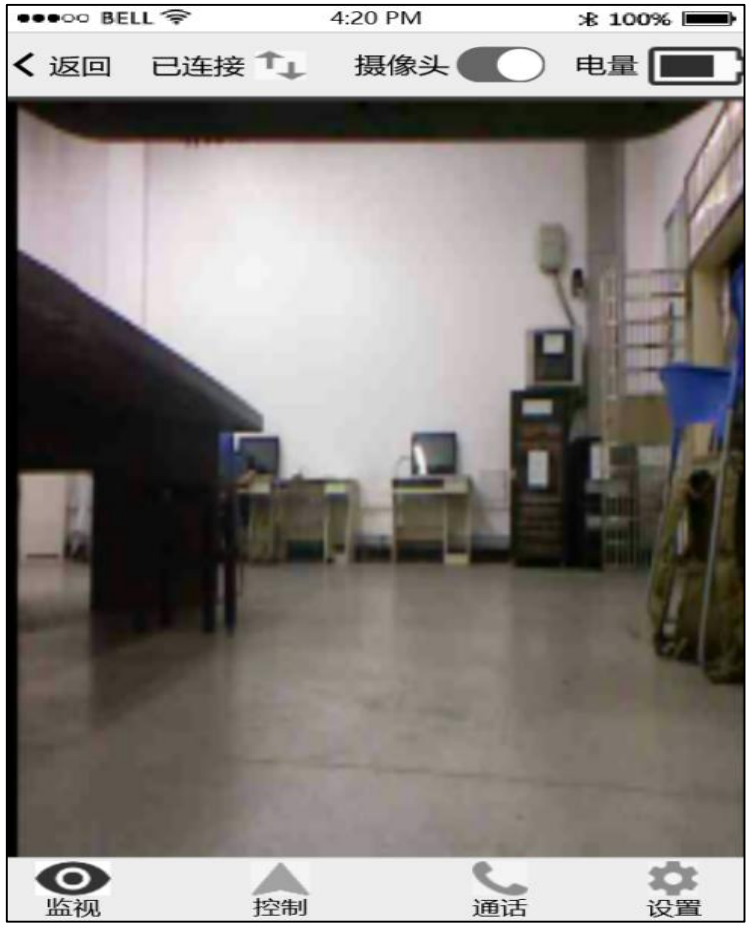
示例：分层的软件体系结构



合理地设计抽象层次和组织软构件是关键

软件设计过程 – 用户界面设计

- 设计软件对外展示以及与用户进行交互的界面，关注软件如何与用户进行交互
  - ✓**输出**：告诉给用户的信息
  - ✓**输入**：需要用户提供的信息



- 设计关注的质量要素
  - ✓直观、友好、易于操作和理解等

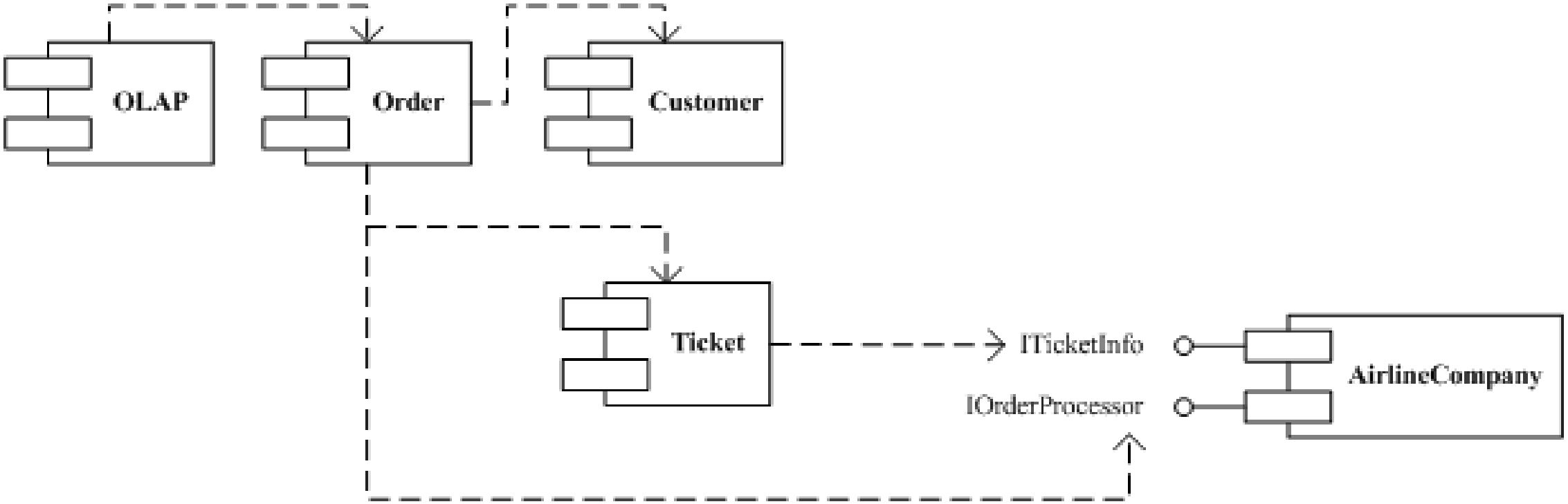
软件设计过程 – 软件详细设计

- **对体系结构设计和人机交互设计成果进行细化和精化，获得高质量的、充分细化的软件设计模型**
  - ✓**构件和类设计**：细化各个构件和类设计，如属性、操作、状态等
  - ✓**接口设计**：构件和类等提供的交互接口
  - ✓**算法设计**：实现特定功能的具体执行流程和算法
  - ✓**数据设计**：信息描述 → 计算机可以处理的数据描述

- **设计关注的质量要素**
  - ✓有效、高效、可靠、易于维护等等

示例：软构件及接口设计

❑软构件及其之间的关系



示例：类设计

```
public class Contact {
```

- ✓private static HashMap<String, String> sContactCache;
- ✓private static final String TAG = "Contact";
- ✓private static final String CALLER_ID_SELECTION;
- ✓public static String getContact(Context context, String phoneNumber)
- ✓……

```
}
```

- 给出类层次的设计信息
  - 属性
  - 方法及其算法等

软件设计过程 – 其它工作

❑撰写设计文档

- ✓基于软件设计及其成果，按照软件设计规格说明书的规范和要求，撰写软件设计文档，详细记录软件设计的具体信息

❑评审软件设计

- ✓对软件设计制品（包括设计模型和文档）进行评审，验证软件设计是否实现了软件需求，分析软件设计的质量，发现软件设计中存在的缺陷和问题，并与多方人员一起协商加以解决

❑软件设计管理

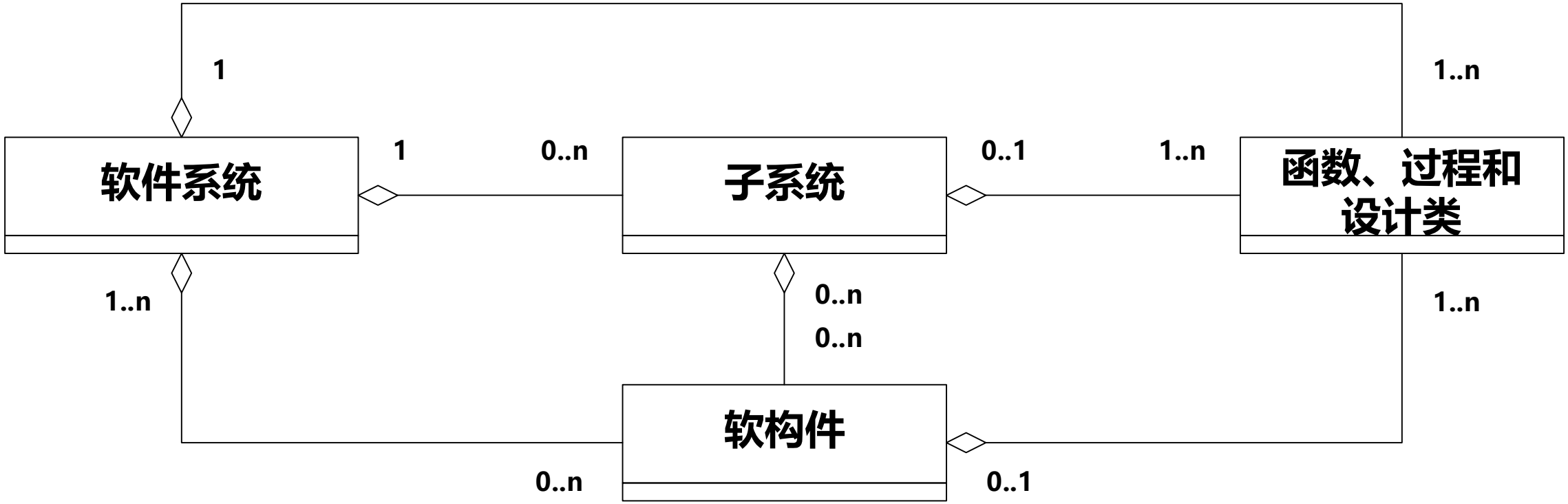
- ✓对软件设计变化以及相应的软件设计制品进行有效的管理，包括追踪软件设计变化、分析和评估软件设计变化所产生的影响、对变化后的软件设计制品进行配置管理等等

1.5 三类软件设计元素

- **过程、函数和设计类**
  - ✓它们既是最基本的设计单元，也是最基本的模块单元
- **软构件**
  - ✓可分离、可独立部署和执行、可单独重用的一类设计元素
  - ✓如**动态链接库**（.DLL）、可运行的**Java JAR包**、**微服务镜像**等就属于软构件
- **子系统**
  - ✓完成特定功能、逻辑上相互关联的一组模块集合
  - ✓有助于管理软件系统的复杂度，简化软件设计和实现，如**软件包**

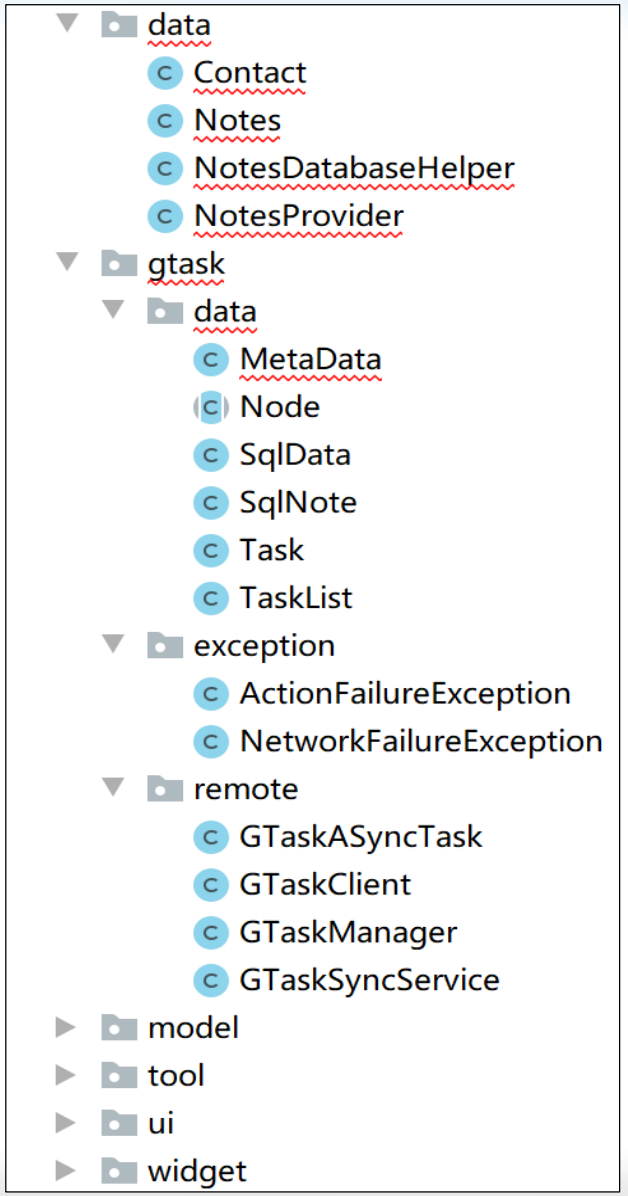
所有的设计元素在编码实现时都有相应的对应物

软件设计元素之间的关系



示例：软件设计元素

小米便签软件体现了哪些设计元素?




```
package net.micode.notes.data;


import ...


public class NotesProvider extends ContentProvider {
    private static final UriMatcher mMatcher;

    private NotesDatabaseHelper mHelper;

    private static final String TAG = "NotesProvider";

    private static final int URI_NOTE            = 1;
    private static final int URI_NOTE_ITEM       = 2;
    private static final int URI_DATA            = 3;
    private static final int URI_DATA_ITEM       = 4;

    private static final int URI_SEARCH          = 5;
    private static final int URI_SEARCH_SUGGEST  = 6;

    static {
        mMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        mMatcher.addURI(Notes.AUTHORITY, "note", URI_NOTE);
        mMatcher.addURI(Notes.AUTHORITY, "note/#", URI_NOTE_ITEM);
```

内容

1. 软件设计概述
   - ✓设计元素、任务和过程
2. **软件设计原则**
   - ✓抽象、求精、模块、隐藏、多视点、分离
3. **软件设计方法**
   - ✓结构化设计方法
   - ✓面向对象设计方法
4. **软件设计输出及评审**
   - ✓软件设计软件制品、软件设计缺陷及评审要求

2.1 软件设计要考虑的因素

- **满足需求**
  - ✓正确、一致、可行、完整、无冗余等
- **权衡抉择**
  - ✓多种设计方案，明确优缺点，综合考虑多方因素
  - ✓关注质量要求
- **应对变化**
  - ✓易于理解和扩展，高效处理等

如何才能得到高质量的软件设计呢？==》软件设计原则

2.2 软件设计的基本原则

① 抽象与逐步求精
② 模块化，高内聚度、低耦合度
③ 信息隐藏
④ 多视点和关注点分离
⑤ 软件重用
⑥ 迭代设计
⑦ 可追踪性

1. 抽象原则

❑何为抽象?

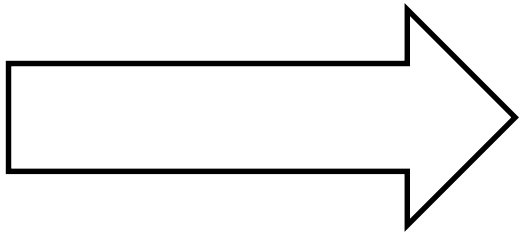
- ✓在认识事物、分析和解决问题的过程中，**忽略那些与当前研究目标不相关的部分**，以便将注意力集中于**与当前目标相关的方面**
- ✓抽象是管理和控制复杂性的基本策略
- ✓如在架构设计时不考虑实现细节

❑抽象在软件设计中的应用

- ✓软件开发就是一个**从高层抽象到低层抽象逐步过渡过程**
- ✓**高层设计(架构设计) → 底层设计(详细设计)**，“逐步求精”过程

软件设计抽象层次的变化

- 结构性
- 全局性
- 关键性



- 过程性
- 局部性
- 细节性

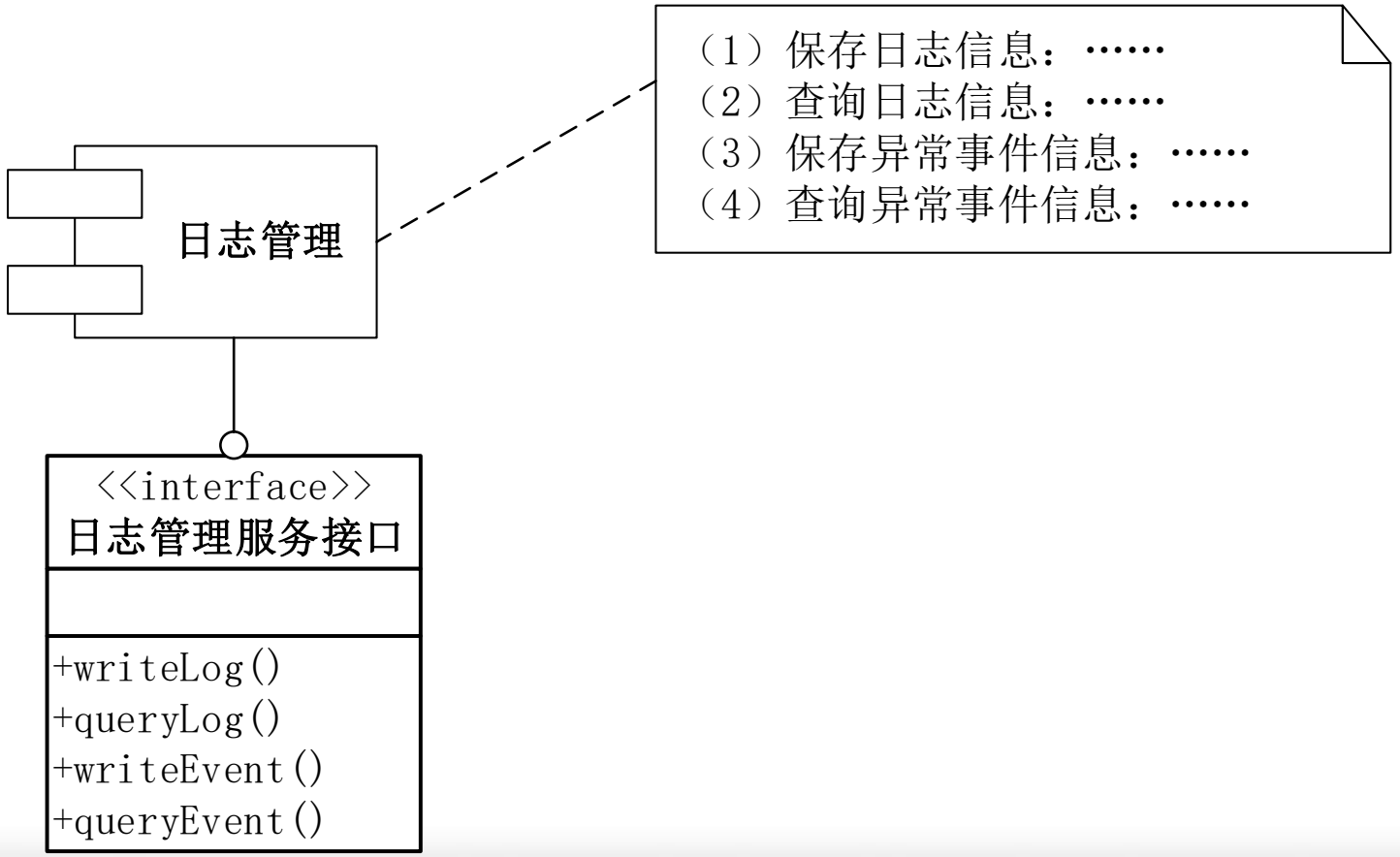
体系结构设计抽象 → 模块抽象(类、函数) → 算法设计抽象

逐步求精

软件设计过程中不采用抽象的原则会产生什么样的结果?

示例：体系结构层次的设计抽象

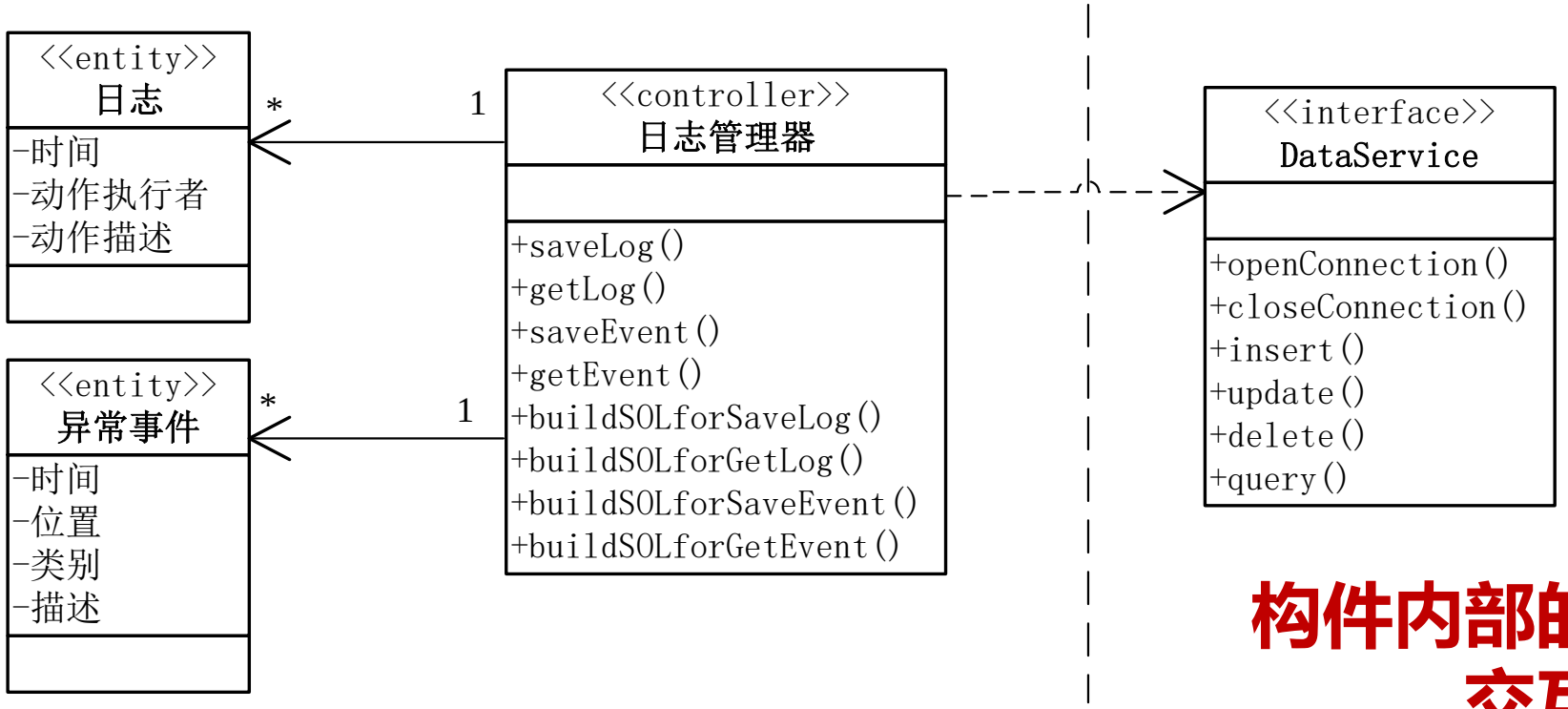
- 关注有哪些构件，它们的**职责和接口**（体系结构设计的抽象），无需关注构件**内部的细节**（构件设计的抽象）



外部的功能和服务

示例：构件层次的设计抽象

- 考虑构件**内部设计元素的职责和协同（构件设计的抽象）**，**无需关注每个类的实现细节（类设计的抽象）**



构件内部的结构和交互

2. 模块化、高内聚度和低耦合度原则

- **将软件系统的整体结构分解为一组相对独立的模块**
  - ✓模块：**包、子系统、构件、类、方法等等**
  - ✓**每个模块实现单一的功能**
  - ✓通过模块之间的交互来组装模块，形成整体框架
  - ✓体现了“分而治之”思想
- **如何达成模块化**
  - ✓模块内部强内聚
  - ✓模块之间低耦合

不按照模块化的原则来进行软件设计会怎样?

高内聚度原则

❑何为模块的内聚度?

- ✓指该模块内各成分间彼此结合的紧密程度，越高越好，高内聚

❑内聚度分类

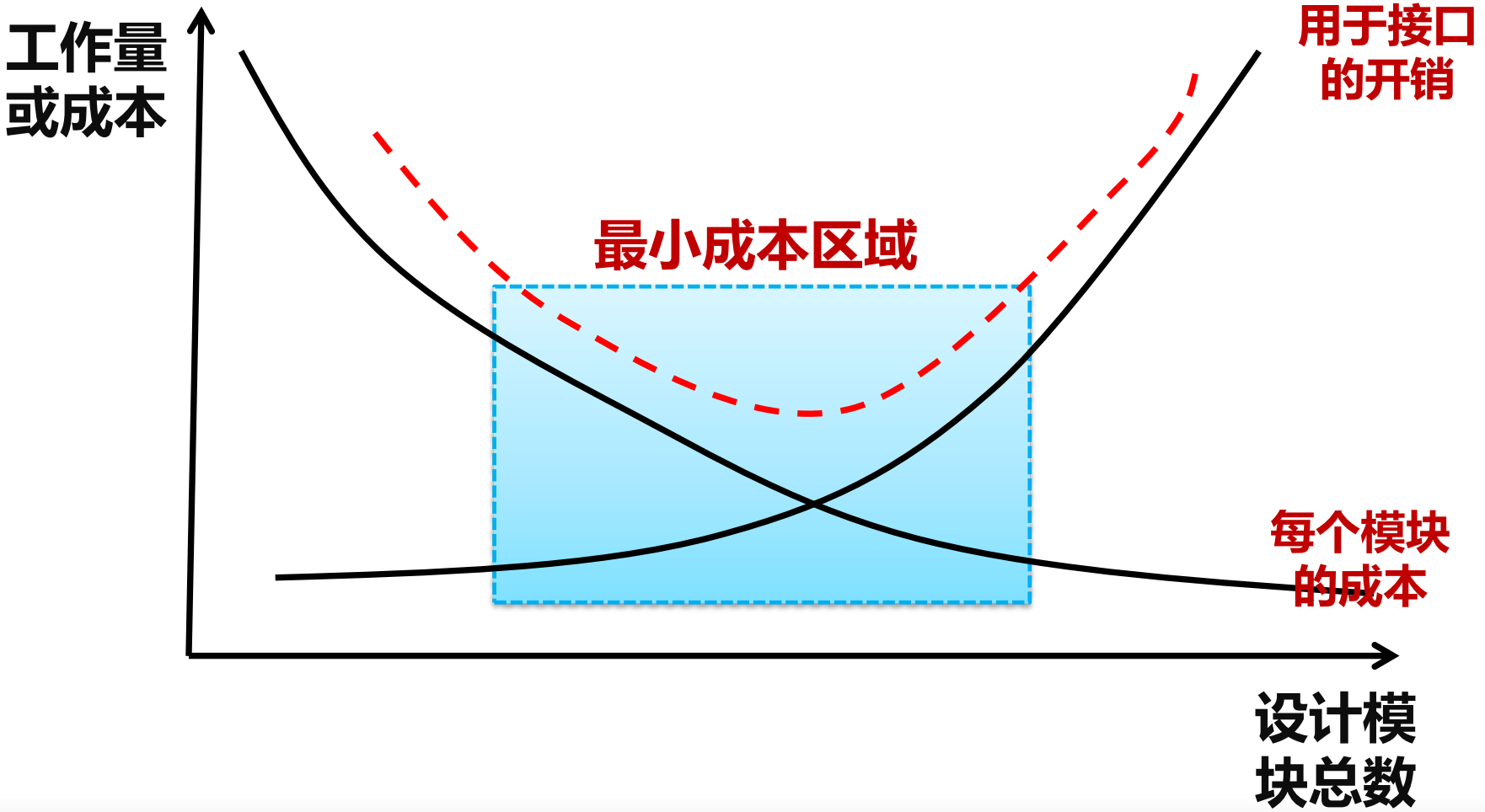
- ✓**偶然性内聚**：模块内各成分为完成一组功能而结合在一起，关系松散
- ✓逻辑性内聚：模块完成的诸任务逻辑上相关
- ✓时间性内聚：模块内诸任务必须在同一时间段内执行
- ✓过程性内聚：模块内各成分相关且必须按特定次序执行
- ✓通讯性内聚：模块内各成分对数据结构的同一区域操作
- ✓顺序性内聚：模块内各成分与同一功能相关且顺序执行
- ✓**功能性内聚：**模块内各成分是一整体，完成单个功能

低耦合度原则

- **何为模块间的耦合度?**
  - ✓模块间的相关程度，越低越好，低耦合
- **耦合度分类**
  - ✓**非直接耦合**：二个模块都不依赖对方而独立存在
  - ✓数据耦合：二个模块通过参数交换信息且仅限于数据
  - ✓控制耦合：二个模块通过参数交换信息包含控制信息
  - ✓特征耦合：介于数据耦合和控制耦合之间
  - ✓外部耦合：二个模块与同一外部环境相关联(文件等)
  - ✓公共耦合：模块间通过全局数据环境相互作用
  - ✓**内容耦合**：一个模块使用另一模块内的数据和控制信息，或者直接转移到另一模块内执行

模块分解与开发成本之间的关系

按照模块化原则开展软件设计，有助于得到模块数量适中、开发工作量和成本较低、易于维护的设计结果



3. 信息隐藏原则

❑何为信息隐藏?

- ✓模块应该设计得使其所含的信息对那些不需要这些**信息的模块不可访问**；模块间仅仅交换那些为完成系统功能所必需交换的信息

❑信息隐藏的优点

- ✓模块的独立性更好
- ✓支持模块的并行开发（设计和编码）
- ✓减少错误向外传播，便于测试和维护
- ✓便于增加新的功能

信息隐藏示例

- 模块只提供**对外接口**，不提供内部**实现细节**
  - ✓**Public 方法对外可访问**
- 某些方法或属性设计为不可访问
  - ✓**Private不可访问**

```
package net.micode.notes.data;


import ...


public class NotesProvider extends ContentProvider {
    private static final UriMatcher mMatcher;

    private NotesDatabaseHelper mHelper;

    private static final String TAG = "NotesProvider";

    private static final int URI_NOTE            = 1;
    private static final int URI_NOTE_ITEM       = 2;
    private static final int URI_DATA            = 3;
    private static final int URI_DATA_ITEM       = 4;

    private static final int URI_SEARCH          = 5;
    private static final int URI_SEARCH_SUGGEST  = 6;

    static {
        mMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        mMatcher.addURI(Notes.AUTHORITY, "note", URI_NOTE);
        mMatcher.addURI(Notes.AUTHORITY, "note/#", URI_NOTE_ITEM);
```

面向对象方法学如何支持信息隐藏?

4. 关注点分离原则

❑何为关注点

- ✓针对概念、任务和目标的某个部分或者侧面的聚焦
- ✓关注点：结构、行为等

❑何为关注点分离

- ✓设计师将若干性质不同的关注点分离开来，以便在适当的时间处理不同的关注点，随后将这些关注点整合起来，形成局部或者全局性的设计结果
- ✓防止“胡子眉毛一把抓”

关注点不分离会产生什么样的后果?

5. 软件重用原则

- 尽可能地**重用**已有的**软件资产**来实现软件系统的功能，同时要确保所开发的软件系统**易于重用**
- 可被重用的软件资产
  - ✓**代码形式**：代码片段、过程、函数、类、软构件、开源软件
  - ✓**其他形式**：软件设计模式、软件开发知识
- 支持软件重用的技术手段
  - ✓封装、接口、继承、多态等

为什么软件重用可以提高软件设计的质量和开发效率?

6. 软件设计的其它原则

- 设计**可追溯**到分析模型
- 经常关注待建系统的**架构**
- **数据设计**和功能设计同样重要
- 必须设计**接口**
- **用户界面**设计必须符合最终用户要求
- 设计表述要尽可能**易于理解**
- 设计应该**迭代**进行

内容

1. 软件设计概述
   - ✓设计元素、任务和过程
2. 软件设计原则
   - ✓抽象、求精、模块、隐藏、多视点、分离
3. **软件设计方法学**
   - ✓结构化设计方法
   - ✓面向对象设计方法
4. **软件设计输出及评审**
   - ✓软件设计软件制品、软件设计缺陷及评审要求

何为软件设计方法?

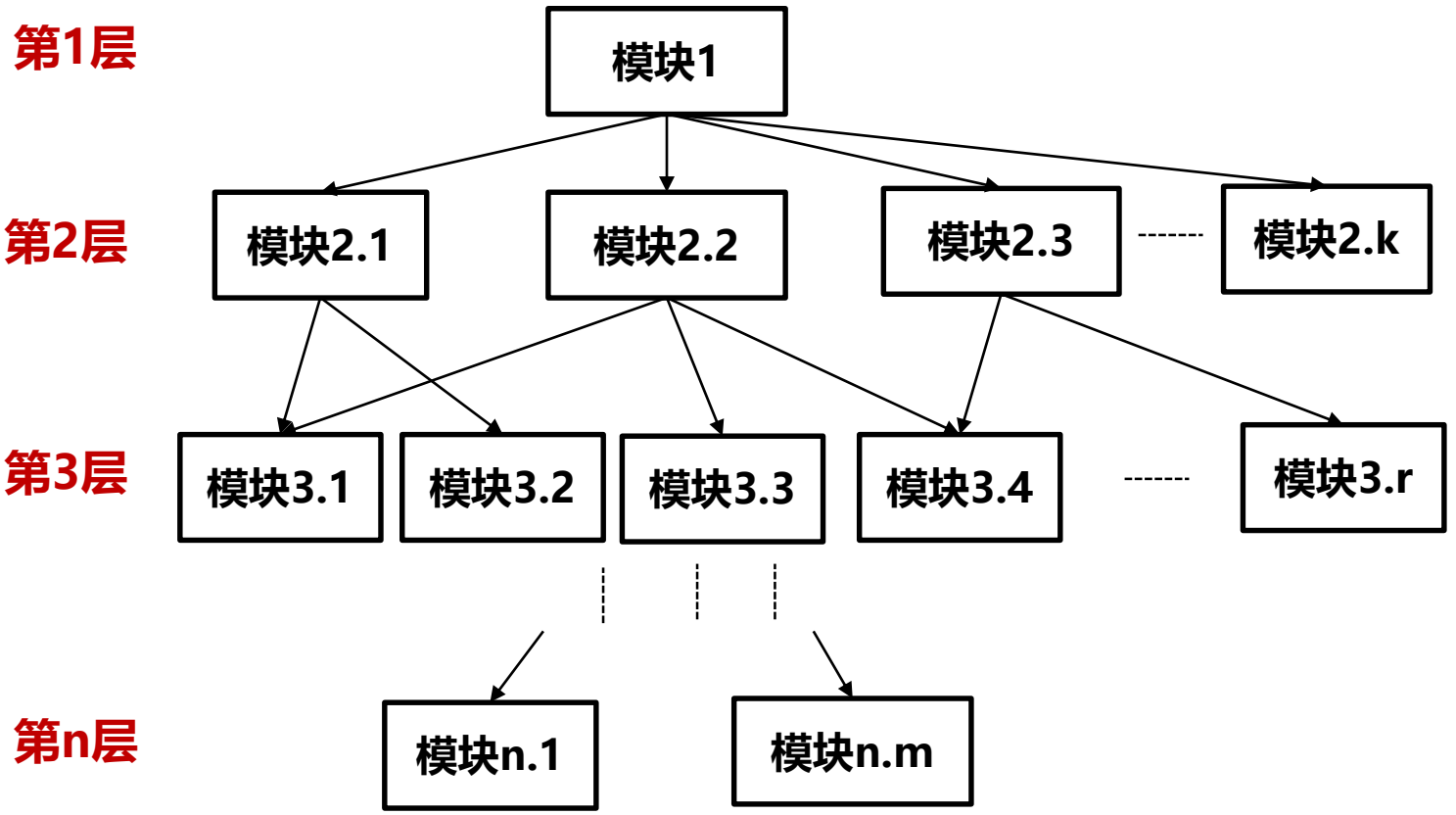
- **软件设计方法明确了按照什么样的思想、理念和技术来开展软件设计**
  - ✓抽象：模块及交互
  - ✓过程：设计步骤
  - ✓语言：设计描述
  - ✓工具：设计支撑
- **有哪些软件设计方法**
  - ✓**结构化软件设计方法**
  - ✓**面向对象软件设计方法**

3.1 结构化软件设计

- **结构化软件设计方法学（Structured Design Methodology）产生于二十世纪七十年代，代表性成果是面向数据流的软件设计方法学**
- **基本思想**
  - ✓基于**结构化需求分析结果**（如数据流图），将其设计为**以功能模块为核心的软件设计模型**（如模块化、层次化的设计模型），最后交由**结构化程序设计语言**（如C、Fortran等）加以实现
- **面向数据流的设计方法学主要用于支持软件的体系结构设计，其思想简单、技术成熟，在软件产业界得到广泛应**

结构化软件设计的输出结果

❑模块层次图 – 软件体系结构



- 模块作为软件体系结构的基本单元
- 模块之间的交互表现为模块调用
- 软件体系结构变现为层次化的形式

结构化软件设计方法的特点

- 以数据流为输入，以层次化的软件体系结构为输出
- 采用转换和映射的方法
- 得到的是软件体系结构，属于软件体系结构设计工作
- 在此基础上可以进一步开展详细设计工作

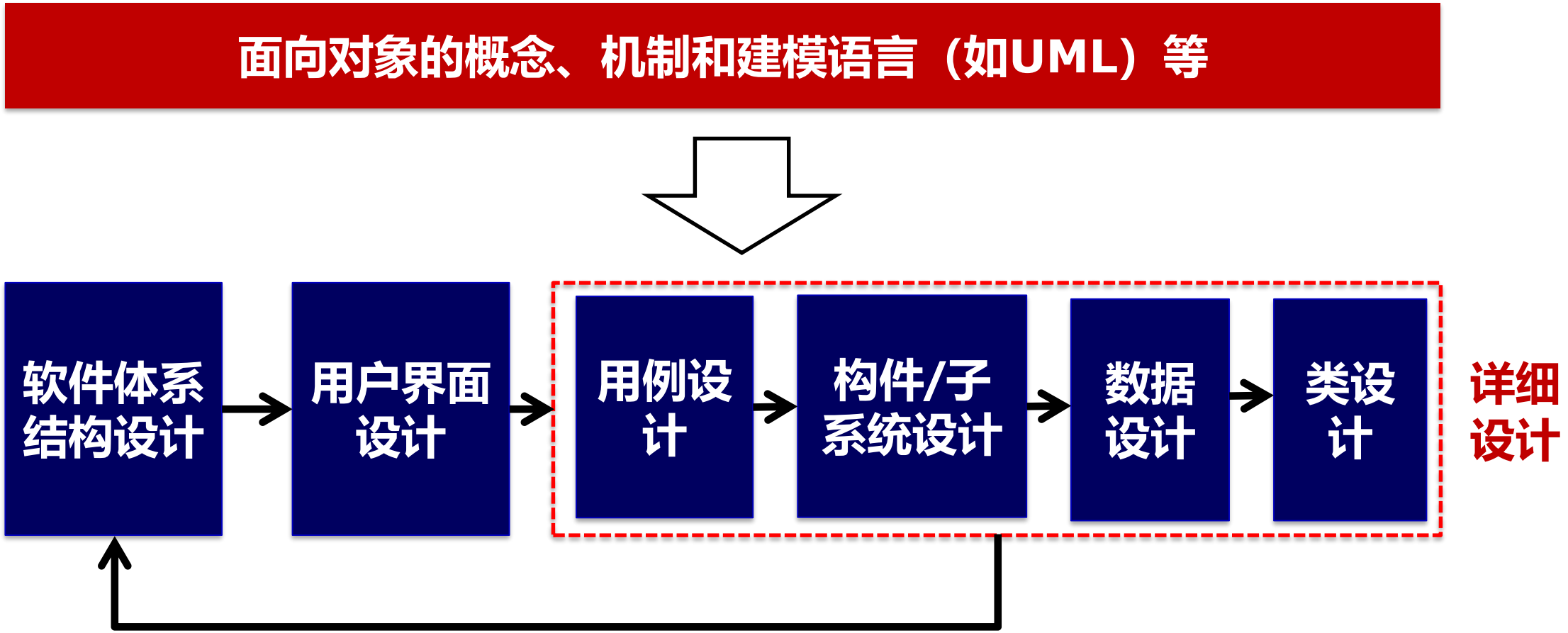
3.2 面向对象软件设计方法学

- 针对面向对象需求分析所得到的**软件需求模型**（如用例图、交互图、分析类图），对其进行不断**精化**（而非转换），获得软件系统的**各类软件设计元素**，如子系统、构件、设计类等，产生不同视角、不同抽象层次的**软件设计模型**，如软件体系结构图、用例设计交互图、设计类图、活动图等，形成软件系统**完整和详尽的设计方案**

面向对象软件设计方法学

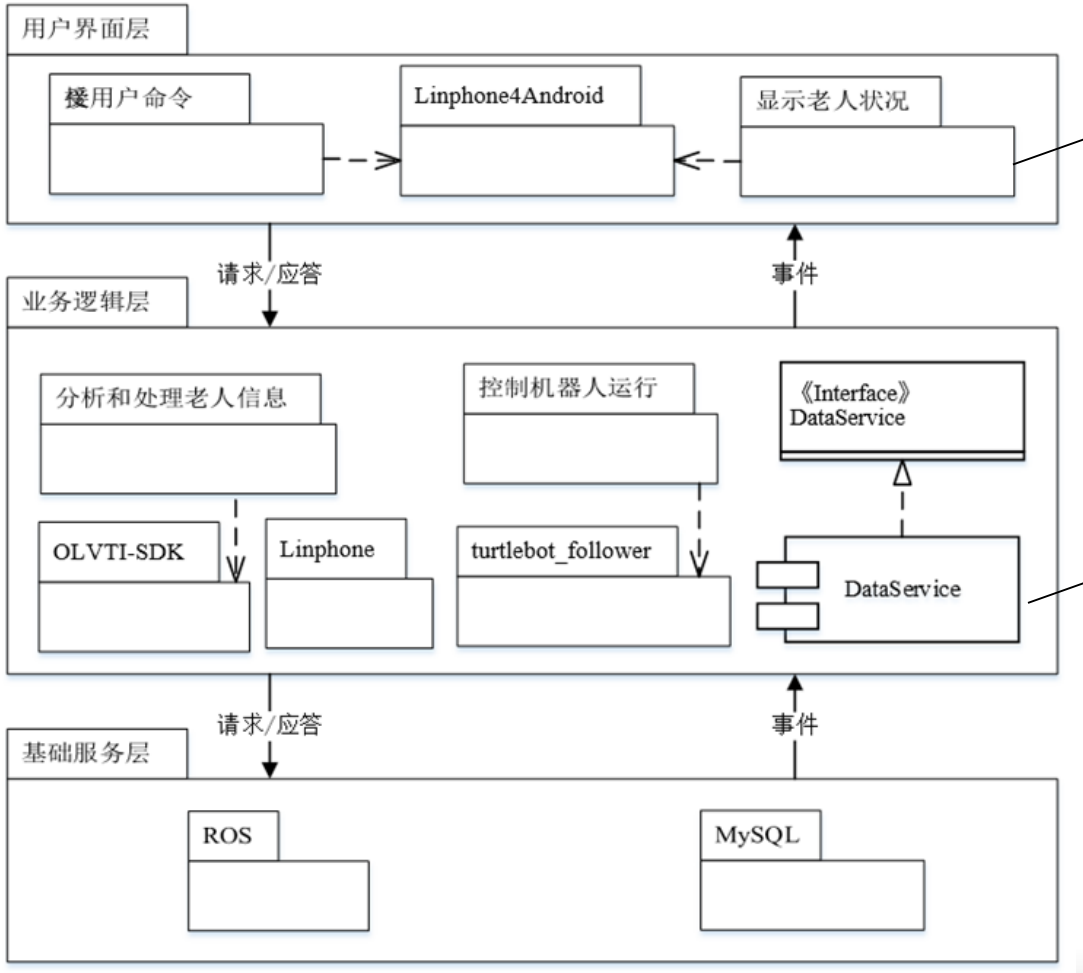
提供了概念、机制、过程、策略等来支持OO软件设计，产生高质量软件设计

面向对象软件设计过程


面向对象的概念、机制和建模语言（如UML）等
软件体系结构设计
用户界面设计
用例设计
构件/子系统设计
数据设计
类设计
详细设计

面向对象软件设计建模 - 包图

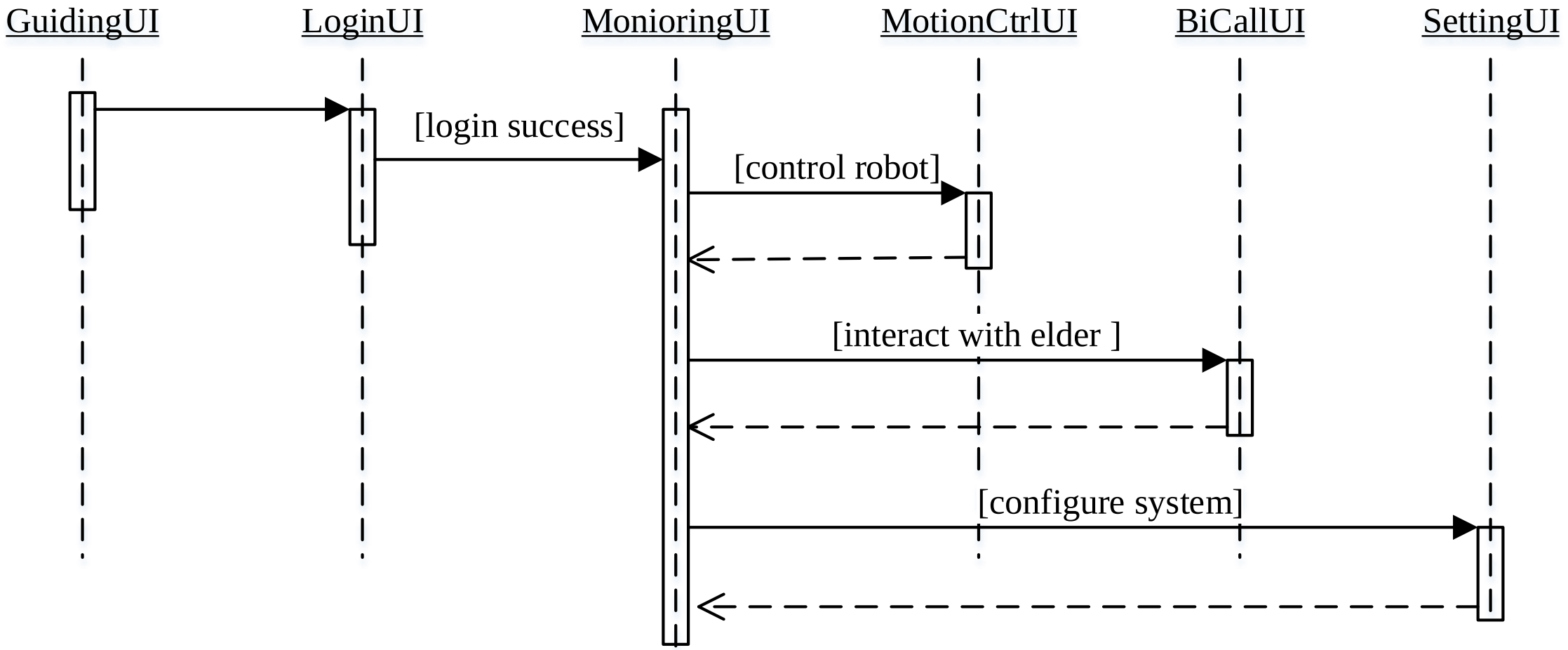
❑用包图表示的软件体系结构设计



包来表示体系结构要素

软构件来表示体系结构要素

面向对象软件设计建模 - 顺序图



顺序图用于表示对象之间的交互时序及内容

面向对象软件设计的优势（1/2）

❑高层抽象和自然过渡

- ✓面向对象概念更加**贴近于现实世界**，有助于对应用问题以及软件系统的直观理解和建模
- ✓采用相同的一组抽象和概念来进行描述和分析，基于模型的精化手段来实现软件设计，极大简化了软件设计工作
- ✓面向对象模型更易于为人们所接受，可减少软件工程师与用户之间的交流鸿沟，有助于支持大型复杂软件系统的开发

❑多种形式和粗粒度的软件重用

- ✓提供了多种方式来**支持软件重用**，进而有助于提高软件开发的效率和质量

面向对象软件设计的优势（2/2）

❑系统化的软件设计

- ✓系统地支持软件设计阶段的所有工作，包括**体系结构设计、用户界面设计、数据设计、软构件设计、子系统设计、用例设计、类设计等等**

❑支持软件的扩展和变更

- ✓提供了**接口、抽象类、继承、实现**等多种机制，可以设计出易于扩展和变更的软件设计模型

3.3 软件设计的CASE工具

- **软件设计文档撰写工具**
  - ✓如借助于Microsoft Office、WPS等
- **软件设计模型绘制工具**
  - ✓如Microsoft Visio、StarUML、Argo UML等工具
- **软件设计分析和转换工具**
  - ✓如IBM Rational Rose等软件工具
- **配置管理工具和平台**
  - ✓如Git、Github、Gitlab、PVCS、Microsoft SourceSafe等，支持软件需求制品（如模型、文档等）的配置、版本管理、变化跟踪等

3.4 软件设计工程师

- 负责软件设计的各项工作，包括**体系结构设计、数据设计、用户界面设计、详细设计等**
  - ✓包括**架构师、数据设计工程师、用户界面设计工程师**等等
  - ✓创新能力
  - ✓抽象和建模能力
  - ✓质量保证能力
  - ✓组织、沟通和协调能力
  - ✓权衡抉择能力

内容

1. 软件设计概述
   - ✓设计元素、任务和过程
2. 软件设计原则
   - ✓抽象、求精、模块、隐藏、多视点、分离
3. 软件设计方法
   - ✓结构化设计方法
   - ✓面向对象设计方法
4. **软件设计输出及评审**
   - ✓软件设计软件制品、软件设计缺陷及评审要求

4.1 软件设计的输出

❑软件设计模型

- ✓它从多个不同的视角、不同的抽象层次描述了软件的设计信息，并采用诸如UML、模块图、层次图等图形化的方式来加以刻画

❑软件设计文档

- ✓它采用自然语言的形式，结合软件设计模型，详细描述软件系统的各项设计，包括体系结构设计、子系统和构件设计、用户界面设计、用例设计、数据设计等等

4.2 软件设计文档规范及其内容

- 文档概述
- 系统概述
- 设计目标和原则
- 设计约束和现实限制
- 体系结构设计
- 用户界面设计
- 子系统/构件设计
- 用例设计
- 类设计
- 数据设计
- 接口设计

软件设计文档采用图文并茂的方式详细描述软件设计的具体内容

4.3 软件设计中的缺陷

- **设计不满足需求**
  - ✓对软件需求的理解存在偏差，未能正确地理解用户的软件需求，导致所设计的软件无法满足用户的需要
- **设计质量低下**
  - ✓设计过程中未能遵循设计原则、缺乏设计经验，导致软件设计质量低下，如设计的软件不易于维护和扩展
- **设计存在不一致**
  - ✓不同软件设计制品对同一个设计有不同的描述，或者存在不一致甚至相冲突的设计内容；多个不同软件设计要素之间存在不一致
- **设计不够详尽**
  - ✓未能提供设计细节性信息，导致程序员无法根据设计来开展编码工作

4.4 软件设计的评审

- **目的：发现软件设计模型和文档中的缺陷**
- **谁参与评审**
  - ✓设计工程师、程序员、测试工程师、用户、质量保证人员等
- **评审什么内容**
  - ✓**文档规范性**，软件设计文档是否符合软件设计规格说明书
  - ✓**设计制品的可理解性**，是否简洁、易于理解
  - ✓**设计内容的合法性**，设计结果是否符合相关的标准、法律和法规
  - ✓**设计的质量水平**，软件设计是否遵循设计原则，质量如何
  - ✓**设计是否满足需求**，设计是否完整和正确地实现了软件需求
  - ✓**设计优化性**，软件设计是否还有待优化的内容

4.5 软件设计的管理

- **软件设计的变更管理**
  - ✓明确哪些方面发生了变更、这些变化反应在软件设计模型和文档的哪些部分、导致软件设计模型和文档的版本发生了什么样变化
- **软件设计的追溯管理**
  - ✓搞清楚是什么原因导致了软件设计的变更，评估设计变更的影响域，评估设计变更对软件项目开发带来的影响
- **软件设计的基线管理**
  - ✓一旦软件设计模型和文档通过了评审，纳入到基线库中

小结

- **软件设计是要给出软件需求的实现解决方案**
  - ✓设计既要满足需求，也要关注质量；设计用于指导实现和编码
- **软件设计有其过程，要循序渐进地开展设计**
  - ✓从体系结构设计、用户界面设计、详细设计
- **软件设计要遵循一系列的基本原则**
  - ✓模块化、信息隐藏、逐步求精、多视点等
- **面向对象软件设计的特点**
  - ✓基于面向对象的概念和抽象，系统性的设计支持，具有多种优点
- **对软件设计结果进行文档化和评审**
  - ✓撰写软件设计文档，发现和纠正软件设计中存在的缺陷

思考和讨论

- 你们准备怎么设计你们的课程设计项目?
- 什么是软件设计？程序设计呢?

课后作业

- 7-1，7-2，7-3，7-7，7-13，7-14
- 用staruml或者类似软件完成7- 19