软件设计基础

内容

1. 何为软件设计

- ✓设计任务
- ✓设计质量
- ✓设计过程
- ✓设计元素

2. 软件设计原则

3. 面向对象软件设计方法学

- ✓基本思想、过程和工具

4. 软件设计输出及评审

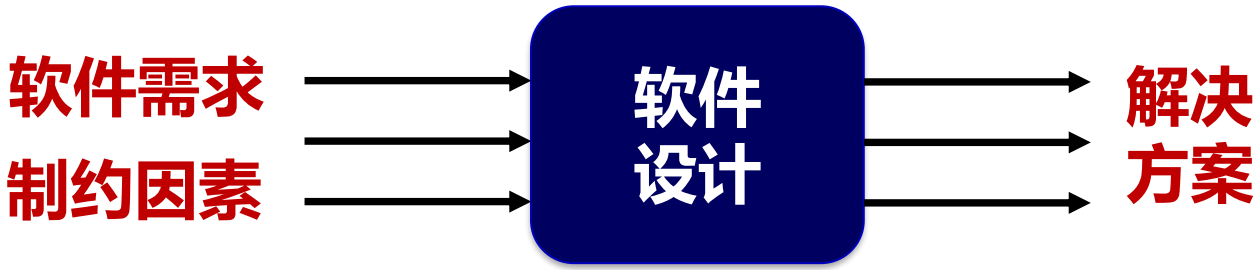
- ✓软件设计软件制品、软件设计缺陷及评审要求

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



❑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

何为软件实现的解决方案?

- 解决方案就是根据需求，对未来**软件**产品的**蓝图规划**，它描述未来**软件组成的元素**
  - ✓模块构成和接口
  - ✓模块之间的交互
  - ✓模块内部的算法
  - ✓人机交互界面
  - ✓数据和数据库结构

需求是站在**用户**视角

设计是站在**工程师**视角

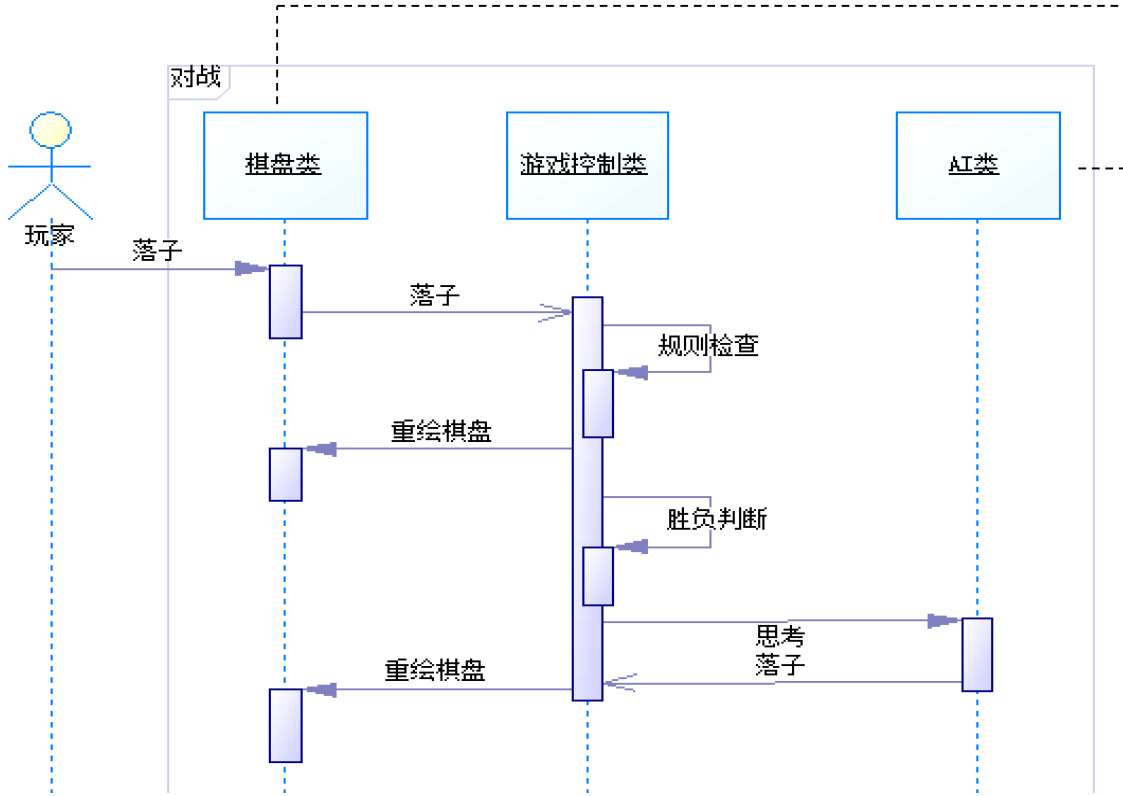
- 软件实现的解决方案类似于“建筑施工图”

南立面图

需求与设计的关系

❑需求 → 设计

- ✓需求回答**做什么**
- ✓设计回答**如何做** → **设计图纸**



需求模型

设计模型

1）界面设计

→ 棋盘类作为界面，应该如何布局和交互?

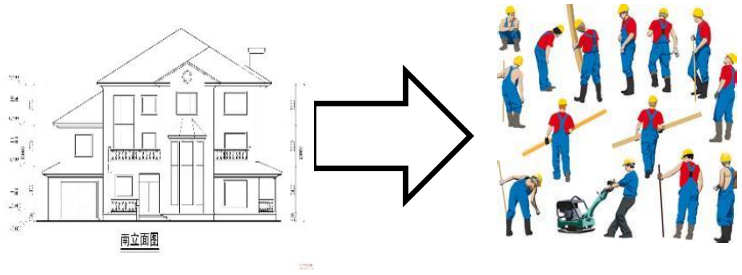
2）详细设计

AI类的职责之一“策略思考”具体应该采用什么算法?

- 强化学习
- 决策树

设计与实现的关系

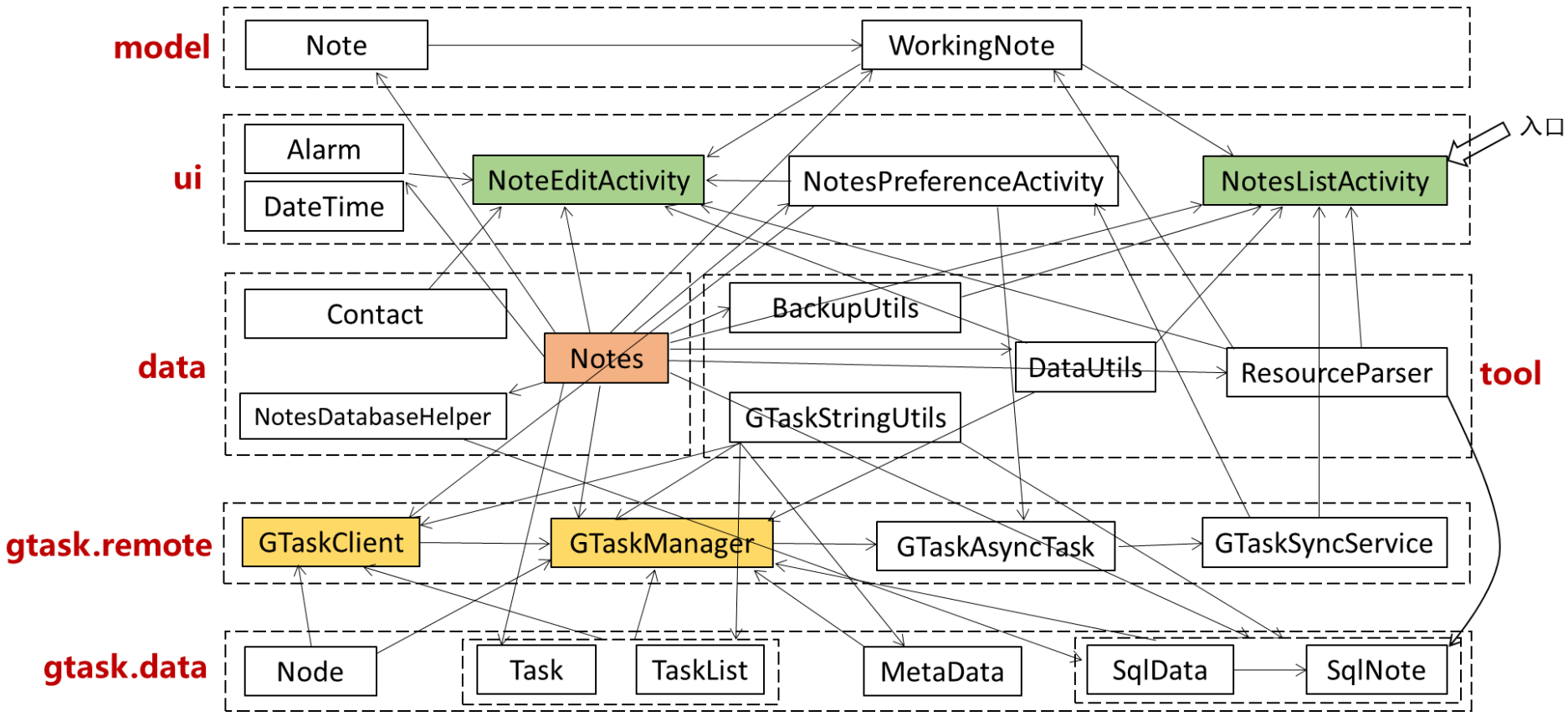
❑设计 ➔ 实现



- ✓基于设计来指导**实现（按图施工）**
- ✓**设计的质量直接决定了软件产品的质量**
  - 如：AI决策算法设计不好，可能推理效率低，同时决策结果差
    UI设计不好，则可能操作繁琐，使用不便

```
private void displayBoard() {
    for (int[] row : board) {
        for (int value : row) {
            System.out.print(value == 0 ? "+" : (value == 1 ? "O" : "X") + " ");
        }
        System.out.println();
    }
}
```

示例：“小米便签”的软件实现解决方案



- ✓ 模块
- ✓ 组织
- ✓ 接口
- ✓ 交互
- ✓ 算法
- ✓ 数据
- ✓ ……

系统架构设计

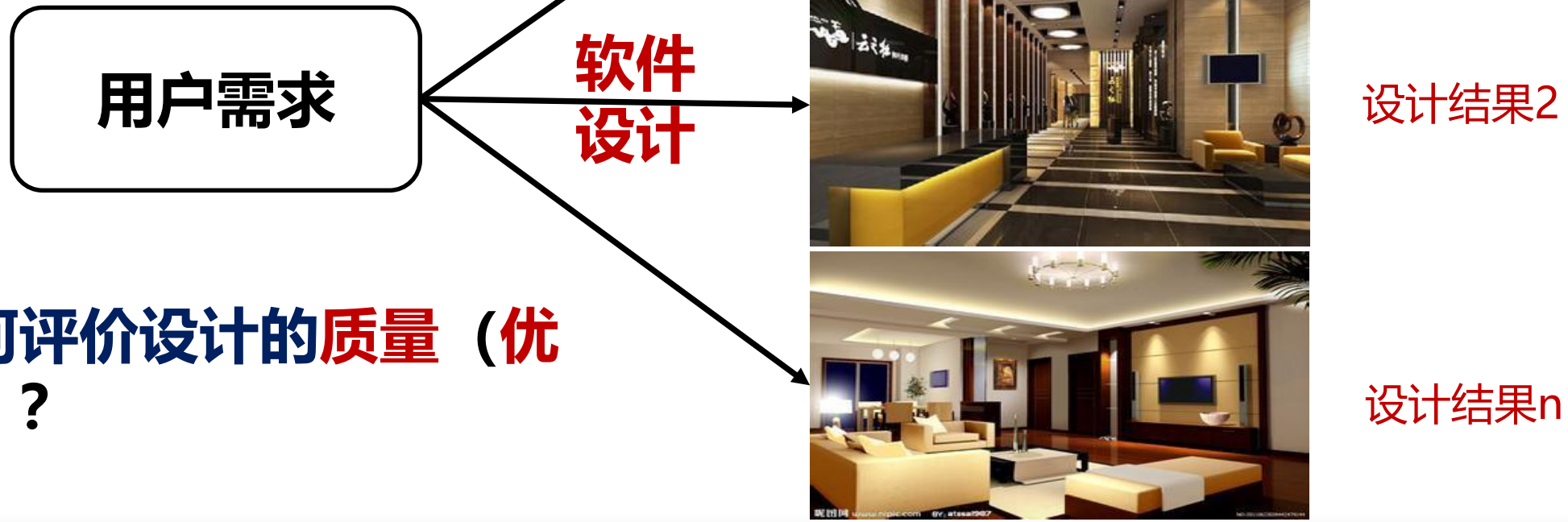
https://github.com/MiCode/Notes

思考和讨论

- 直接根据软件需求来编写代码行吗？为什么？

1.2 设计质量与多样性

- 软件设计是一个创作的过程，展示工程师的创造性
- 因此，同一个需求也会有多种软件设计方案



✓ **如何评价设计的质量（优劣）？**

软件设计的质量要求

- **满足需求**
  - ✓设计必须实现所有利益相关者的需求
- **遵循约束**
  - ✓设计必须满足软件项目的实现约束，确保其后期可以通过编码实现
  - ✓思考：公交微信支付 设计上有什么约束?
- **充分优化**
  - ✓根据设计优化原则，权衡多种设计方案，确保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、可靠性和可修改性等
- **足够详细**
  - ✓提供完整、全面、细致的设计内容。例如包括体系结构、接口、详细设计、数据设计等。
- **通俗易懂**
  - ✓确保设计其对后续的编码、测试以及维护人员，是可读、可理解的指南。

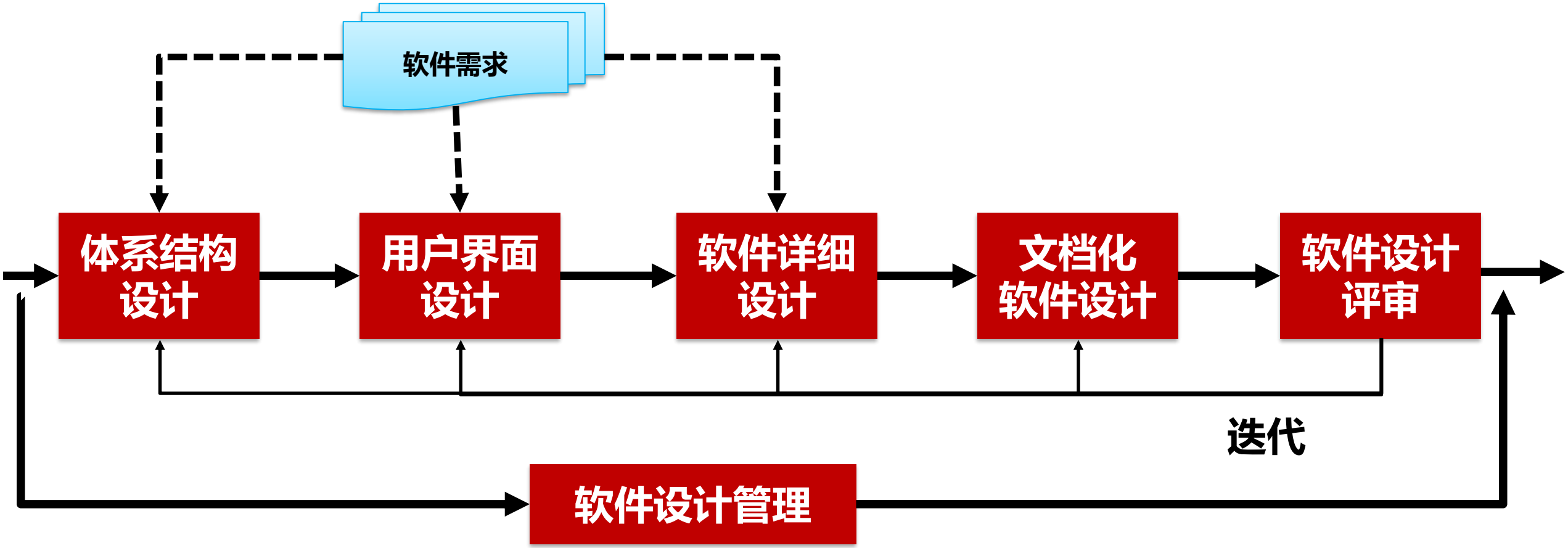
思考和讨论

- 如果设计质量不高，会带来什么样的问题?

“编写一段能工作的代码是一回事，设计能支持某个长久业务的东西则完全是另一回事”

- 更改设计和更改代码哪个更容易?

1.3 软件设计过程

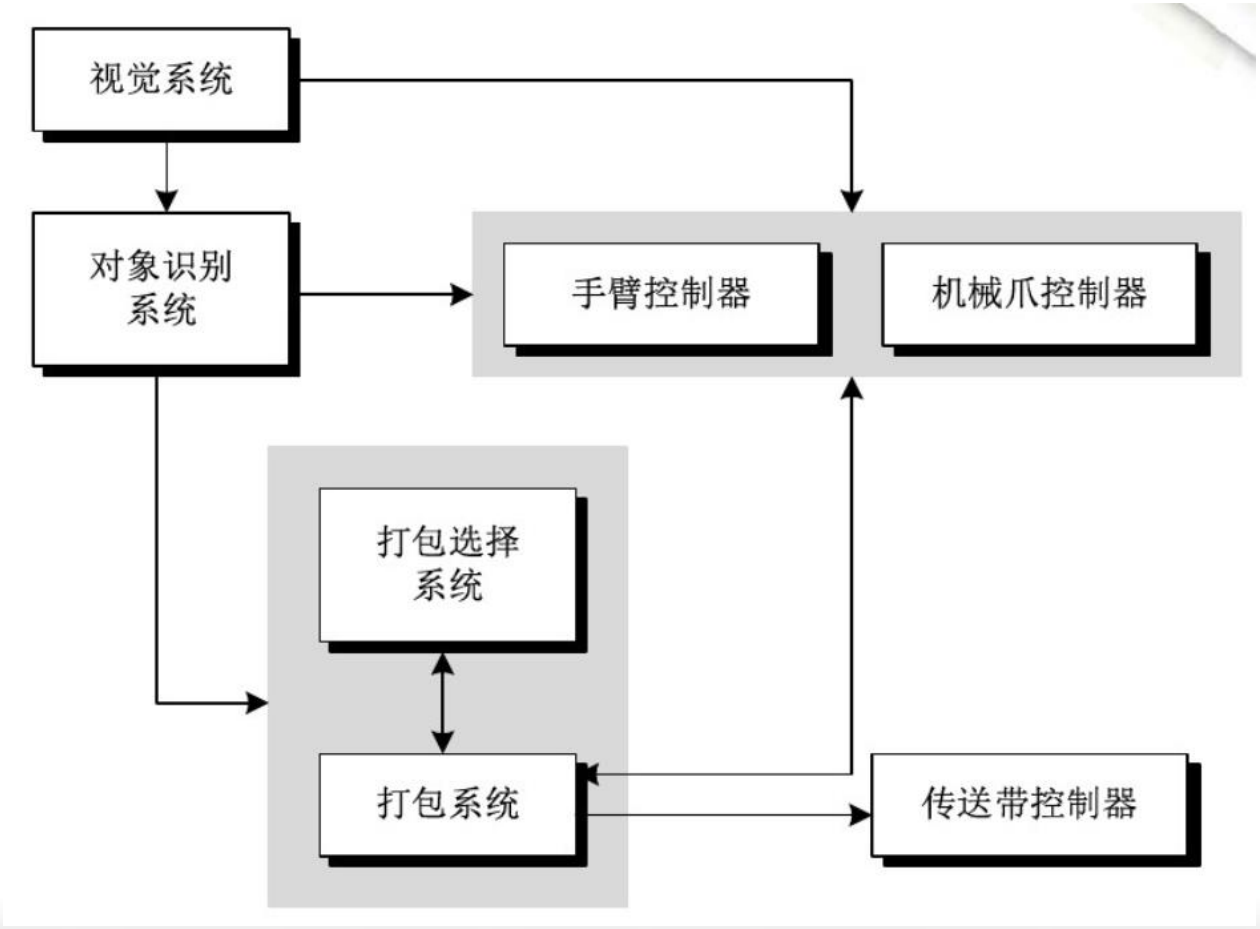


软件设计过程(1) – 软件体系结构设计

- **从全局和宏观视角、站在抽象层次，设计目标系统的构成元素，如**
  - ✓包括哪些子系统或构件
  - ✓它们对外的接口定义
  - ✓它们之间的关系
- **将每个构成元素视为“黑盒子”**
- **该设计关注的质量要素包括**
  - ✓可扩展、可维护、可重用、可移植、可互操作等

示例：打包机器人系统体系结构

- 机器人使用视觉子系统获取传送带上的对象，识别对象类型并选择正确的打包方式，然后从传送带取下对象，打包，再送往另一个传送带。



软件设计过程(2) – 用户界面设计

- 设计软件与用户之间**交互**的界面
  - ✓**输出**：告诉给用户的信息
  - ✓**输入**：用户提供的信息

- 该设计关注的质量要素包括
  - ✓直观、友好、易于操作和理解等

软件设计过程(3) – 软件详细设计

- **对体系结构和UI设计成果进行精细化，获得充分细化的设计模型**
  - ✓**用例设计**：基于设计元素，细化用例交互模型
  - ✓**子系统和构件设计**：细化子系统和构件的内部设计元素
  - ✓**类设计**：提供类的细节，如属性、操作、接口、具体算法
  - ✓**数据设计**：数据结构、描述、存取方式等

- **目标：为下一步编码、实现提供指南**

示例：子系统和构件设计

- 例如：打包机器人中的“对象识别子系统”可以被细化为哪些设计类？

- 例如：订单处理构件，可以被细化为多个设计类？
  - ✓Order类：
    - 存储订单的基本信息，如订单号、客户ID、订单状态、总金额等。
  - ✓OrderItem类：
    - 代表订单中的一个具体商品项（如商品ID、名称、价格、数量等）。
  - ✓OrderRepository类：
    - 负责订单数据的持久化，与数据库交互。

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

思考和讨论

- 不按照上述过程来进行软件设计行吗?
- 先进行详细设计，再进行体系结构设计会产生什么样的问题?

1.4 软件设计元素

❑子系统

- ✓完成特定功能、逻辑上相互关联的一组构件集合，如**软件包。**
- ✓子系统通常是为实现系统的部分功能而设计，如微信是一个系统，聊天，朋友圈就是子系统

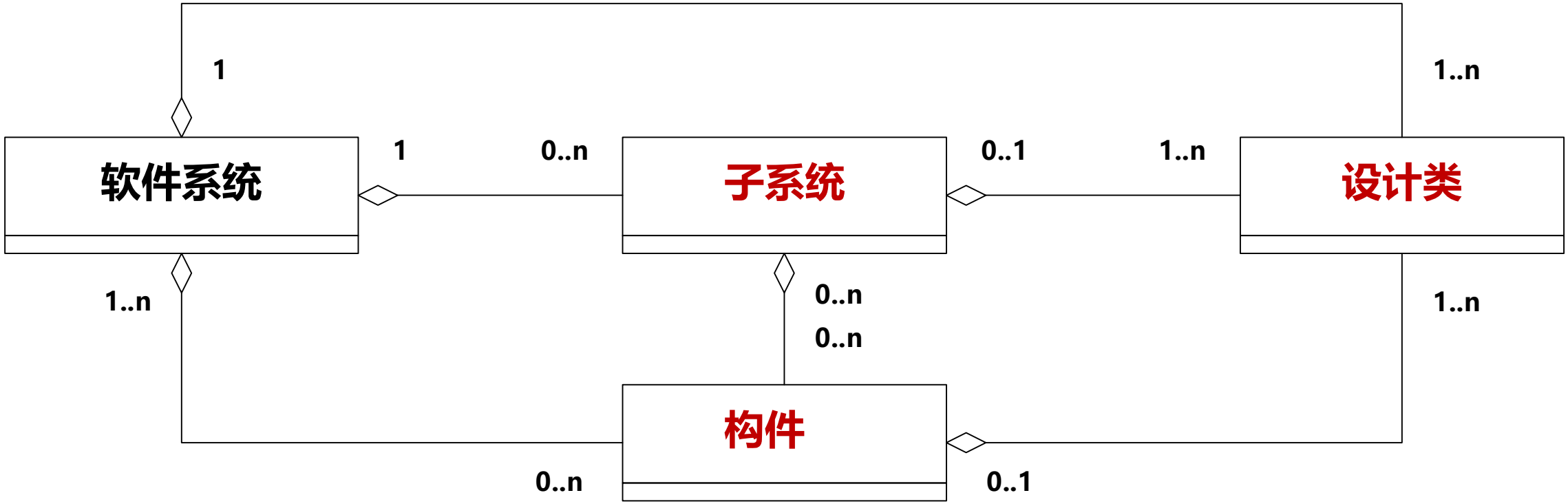
❑构件

- ✓可重用的一类设计元素，封装了某个职责。
- ✓每个构件均有相应的接口，并通过接口对外提供服务。
- ✓构件是构成软件(子)系统的基础构建块。
- ✓如**动态链接库**、**JAR包**、**微服务**等就属于构件

❑设计类

- ✓类是最细粒度的设计元素，构件通常由一组相互协作的类构成。

软件设计元素之间的关系



内容

1. 何为软件设计
   - ✓设计任务
   - ✓设计质量
   - ✓设计过程
   - ✓设计元素
2. **软件设计原则**
3. **面向对象软件设计方法学**
   - ✓基本思想、过程和工具
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

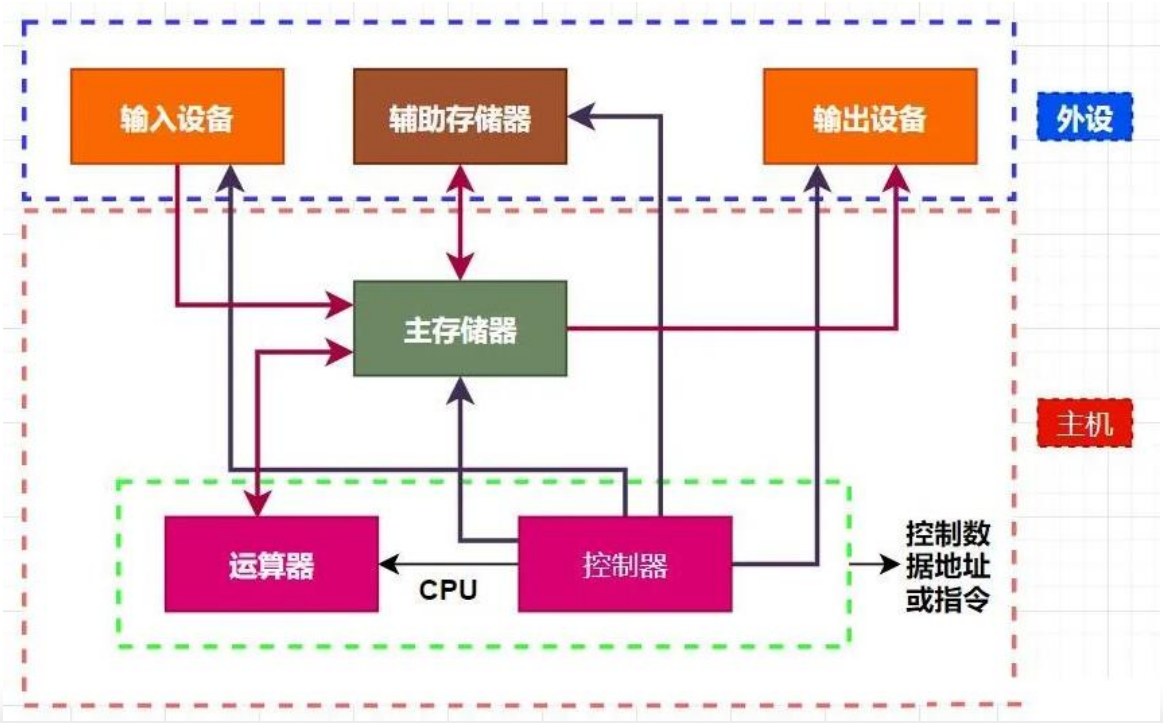
2.1 软件设计的基本原则

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

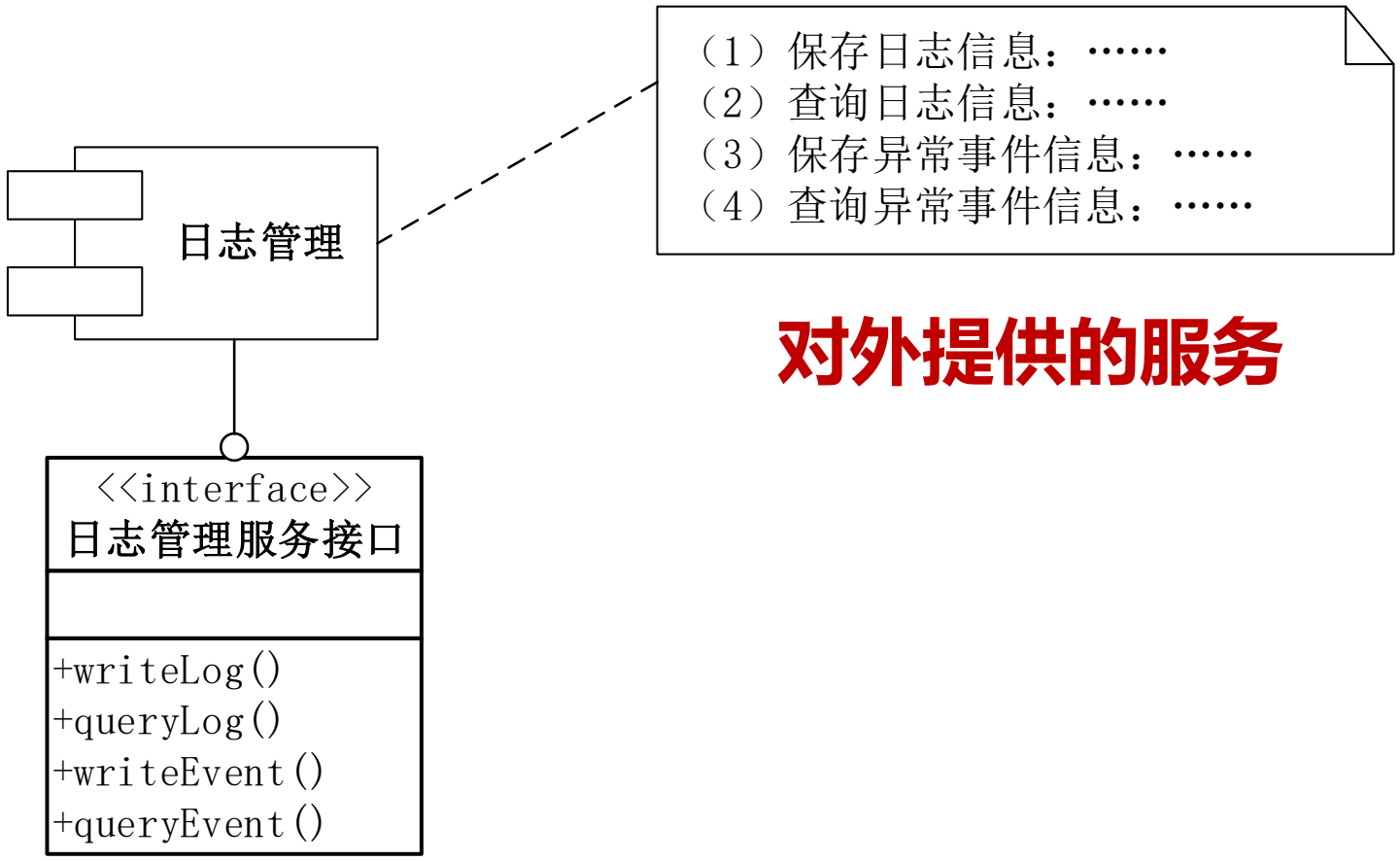
❑何为抽象?

- ✓**忽略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**
- ✓抽象是控制复杂性的基本策略
- ✓如：架构设计时不考虑实现细节



示例：体系结构层次的设计抽象

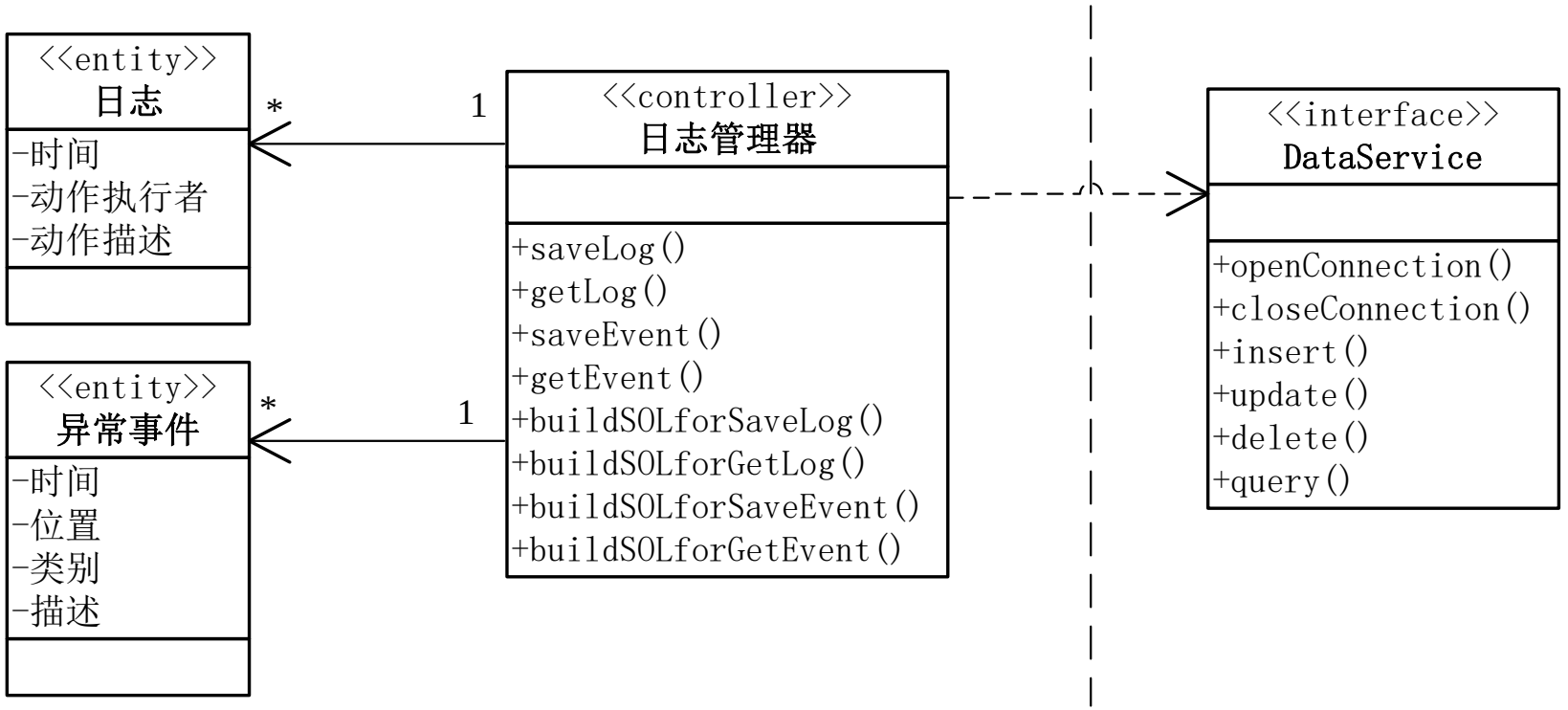
- 关注构件的**职责和接口**，无需关注构件**内部的细节**



对外提供的服务

示例：构件层次的设计抽象

- 关注构件**内部设计元素构成，无需关注每个类的内部细节**



构件内部的设计类和关系

2. 模块化、高内聚和低耦合原则

- **模块化是指将软件系统分解为一组相对独立的模块**
  - ✓模块：**包、子系统、构件、类等**
  - ✓**每个模块实现单一的功能，**并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

- **模块化应该遵循**
  - ✓模块内部强内聚
  - ✓模块之间低耦合

低耦合度原则

❑何为模块间的耦合度?

✓模块间的相关程度。低耦合意味着模块间关系简单，便于独立开发。

❑ 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

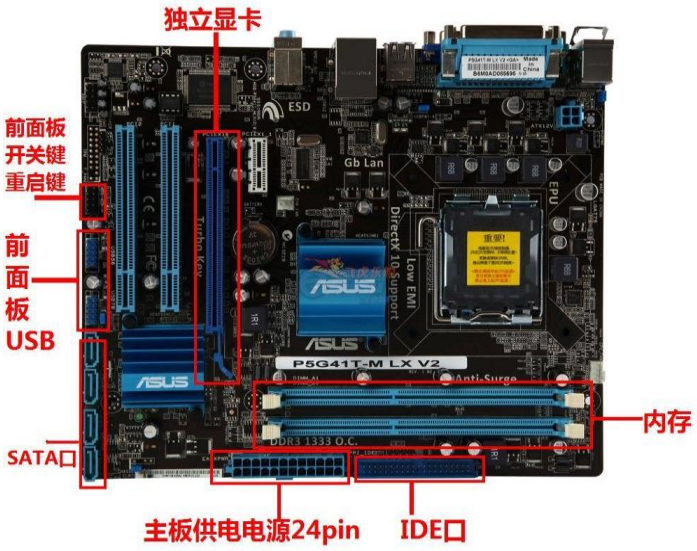
✓ 控制耦合：二个模块通过参数交换信息包含控制信息

✓ 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



低耦合意味着替换显卡，不会影响其它组件

高内聚度原则

❑何为模块的内聚度?

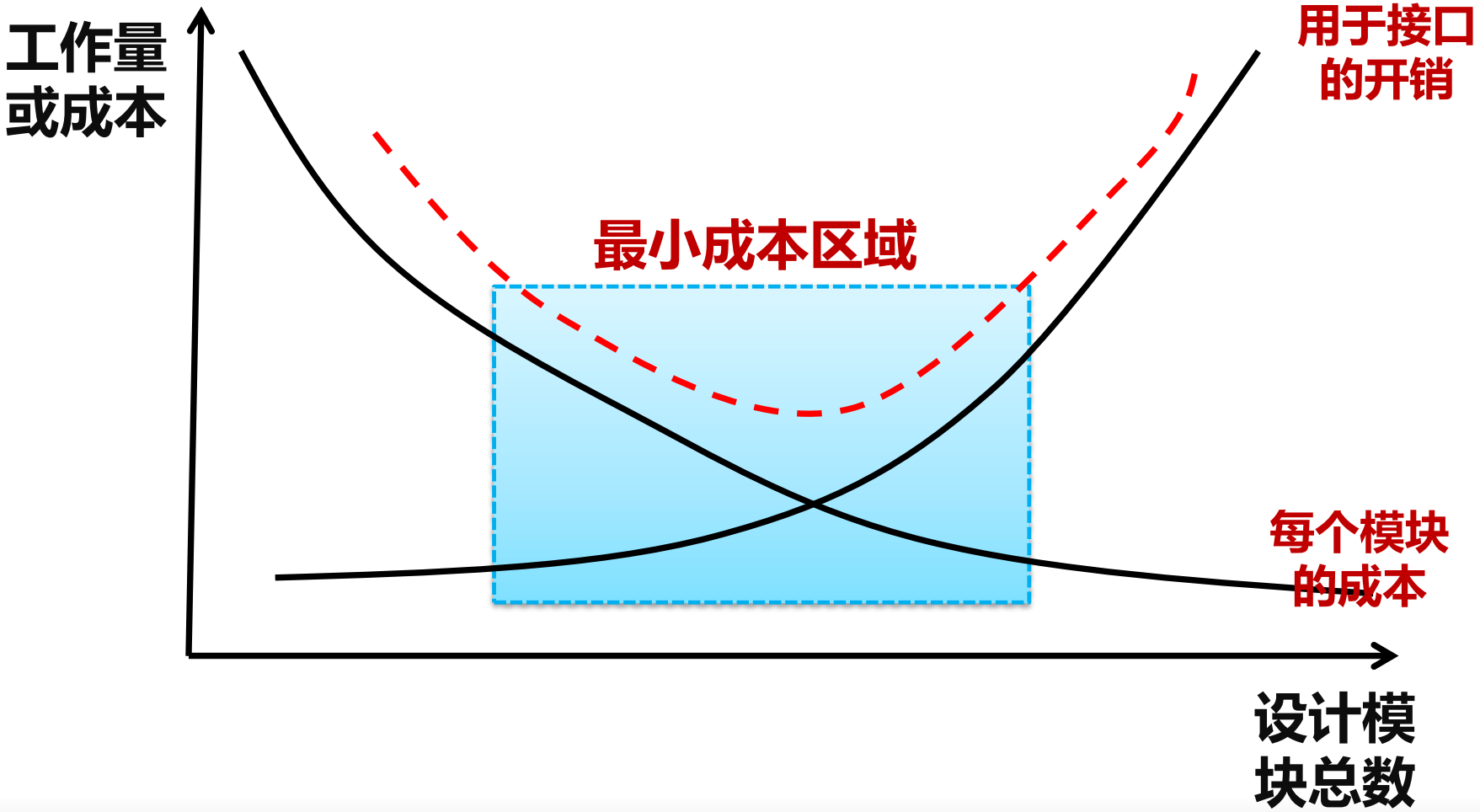
- ✓指该模块内各成分间彼此结合的紧密程度。
- ✓高内聚意味着模块内部只围绕一个特定的功能或职责进行组织。

❑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

模块分解与开发成本之间的关系

模块数量不是越多越好，要适中



3. 信息隐藏原则

❑何为信息隐藏?

✓软件设计中，将模块的内部实现细节隐藏起来，只暴露必要的公共接口给外部使用。

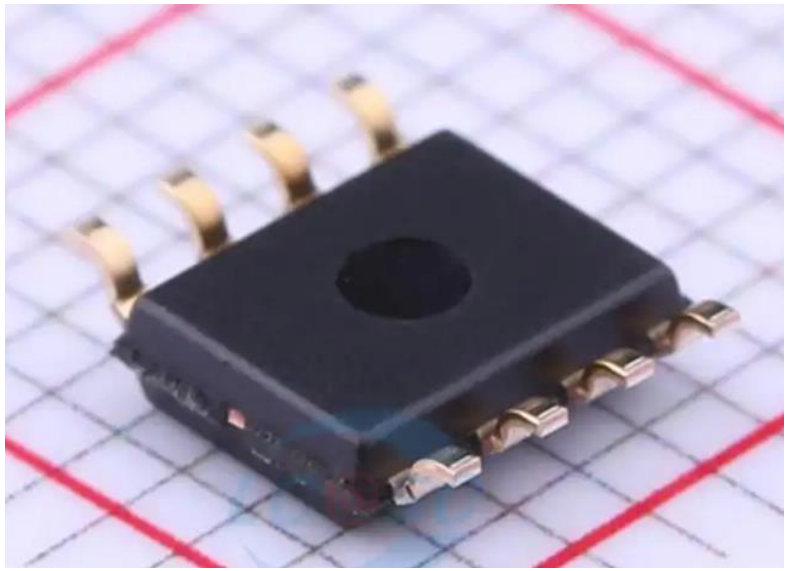
❑优点

✓降低模块之间的耦合性，提高独立性

✓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

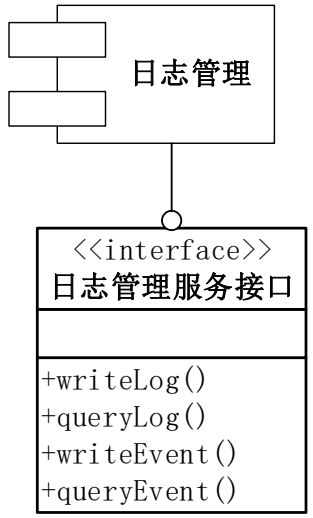
✓便于增加新的功能



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);
```



4. 关注点分离原则

❑何为关注点

- ✓关注点指的是系统设计时需要特别关注或考虑的某个方面。
- ✓软件系统具有**多面性的特点**，如既有结构特征（如软件的体系结构），也有安全特征（认证、授权、数据加密）；

❑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，以便在不同的时间处理不同的关注点，随后将这些关注点整合起来，形成全局性的设计结果
- ✓防止“胡子眉毛一把抓”

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

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. 何为软件设计
   - ✓软件任务
   - ✓设计质量
   - ✓设计过程
   - ✓设计元素
2. 软件设计原则
3. **面向对象软件设计方法学**
   - ✓基本思想、过程和工具
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

3.1 面向对象软件设计的基本思想

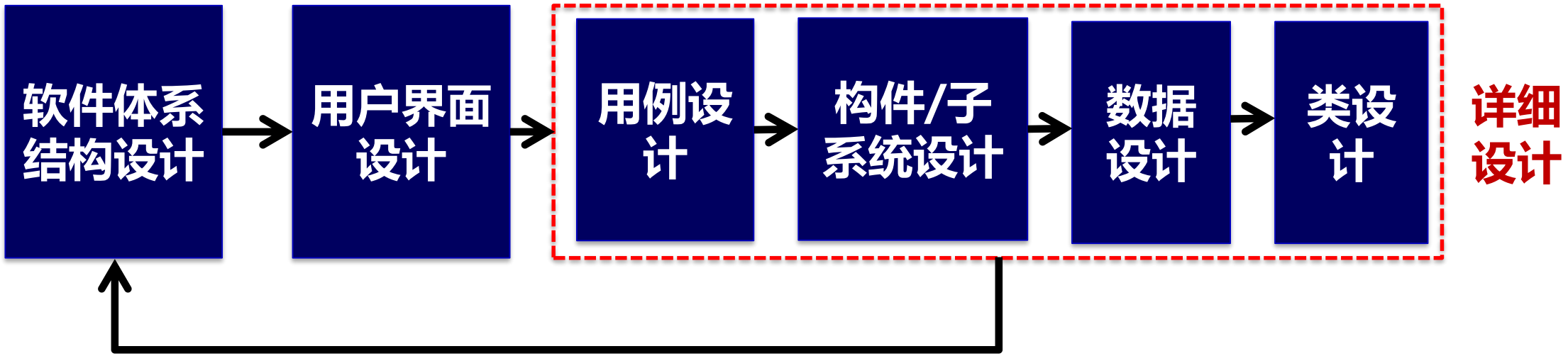
- 面向对象设计采用面向对象分析**相同的概念**（如**问题空间**与**设计空间**都是以类、对象为核心）

- 因此，设计与分析之间不存在鸿沟，而是对**分析模型**（如用例图、交互图、分析类图）的进一步**精化**（而非转换）

- 从而获得**软件系统解决方案**中关注的**各类设计元素**，如子系统、构件、设计类等。

3.2 面向对象软件设计过程

遵循**先整体后局部，先抽象后具体**的设计原则

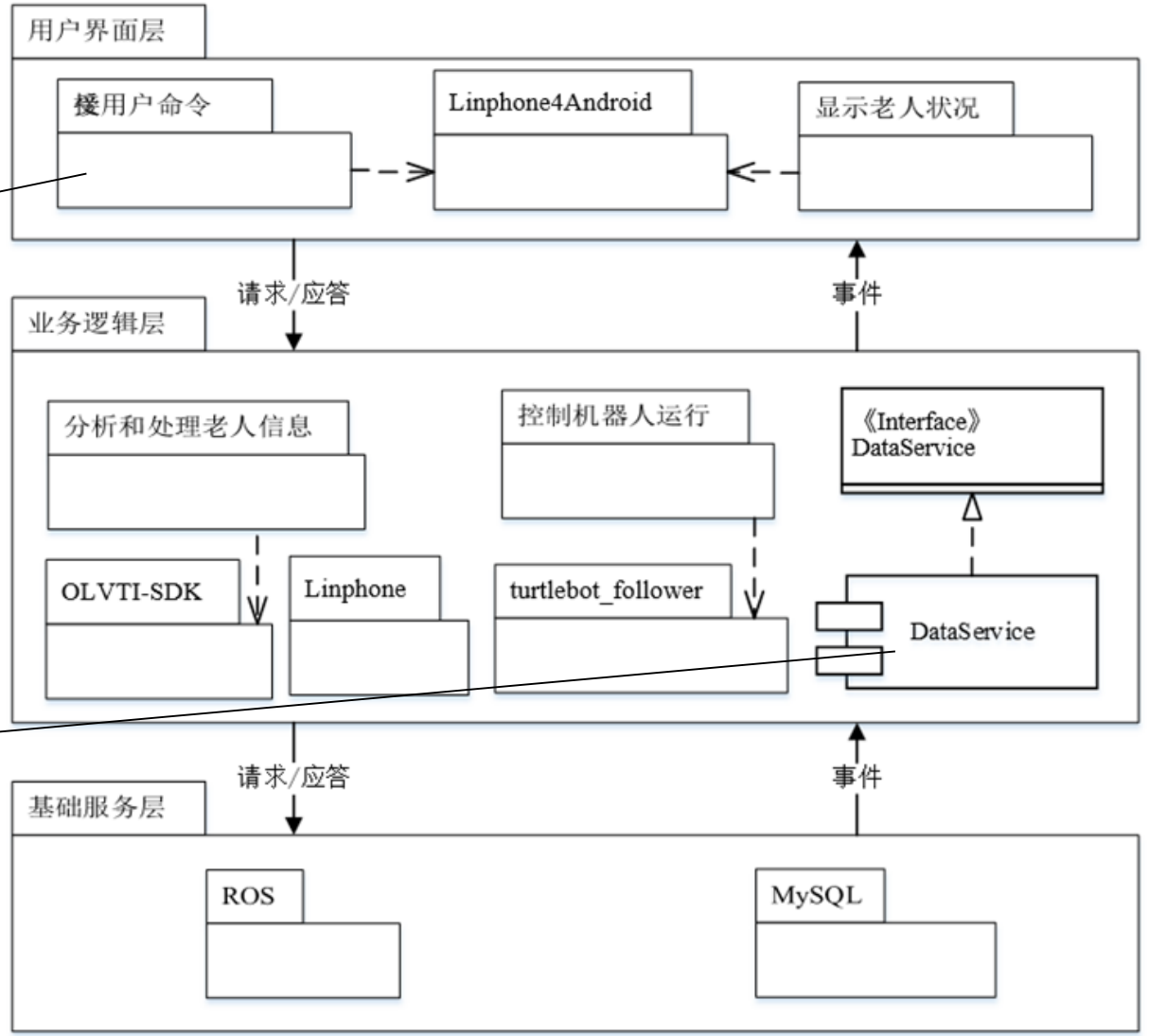


面向对象的软件设计表示

- 不同设计阶段，构建不同的UML设计模型

用包表示的体系结构元素

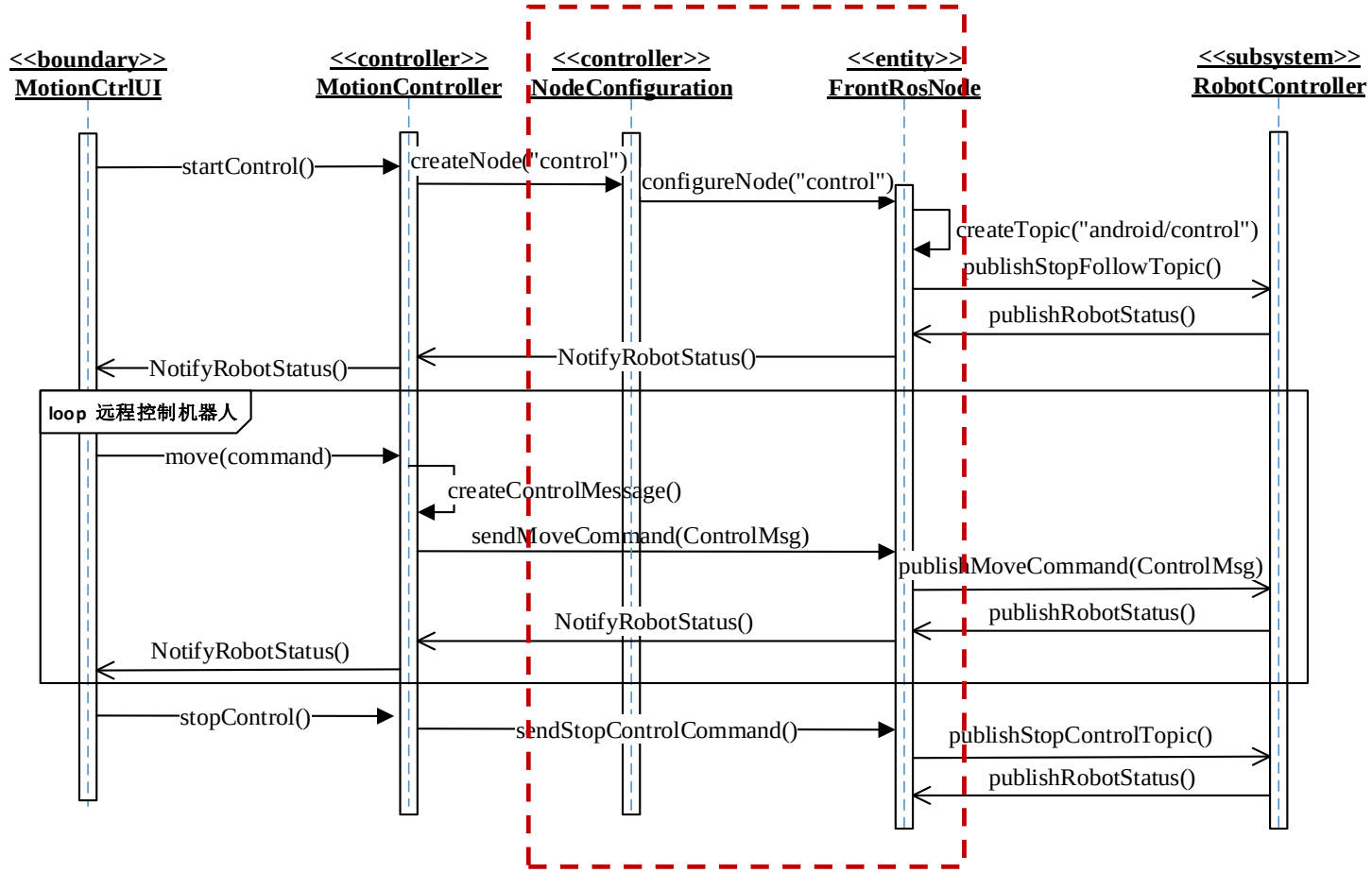
用构件表示的体系结构元素



UML包图表示的软件体系结构

面向对象的软件设计表示

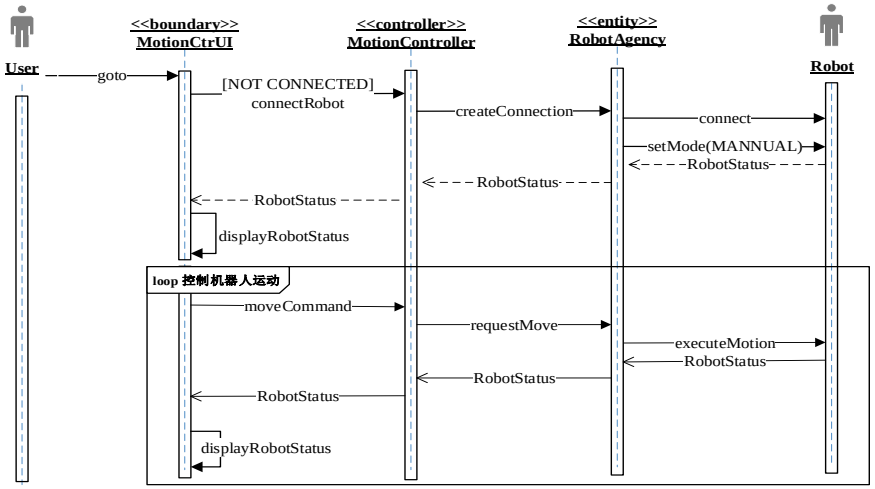
❑ UML交互图表示的用例实现模型



通过手机远程控制机器人的 用例实现模型

ROS（Robot Operating System）

节点：一个进程，可以通过网络与其他节点通信



分析阶段的 顺序图

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、Axure、PowerDesigner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等，支持软件设计制品（如模型、文档等）的配置、版本管理、变化跟踪等

内容

1. 何为软件设计
   - ✓软件任务
   - ✓设计质量
   - ✓设计过程
   - ✓设计元素
2. 软件设计原则
3. 面向对象软件设计方法学
   - ✓基本思想、过程和工具
4. **软件设计输出及评审**
   - ✓软件设计输出制品、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

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

综合实践一

- **任务：搜寻可支持新需求实现的开源软件或其它可重用软件资源**
- **方法**
  - ✓分析开源软件的实现技术和运行环境，到开源软件托管平台中寻找合适的开源软件
- **要求**
  - ✓深入理解软件设计需要满足的约束和限制，找到可有效支持新需求实现的可重用软件资源
- **结果：无**

综合实践二

- **任务：搜寻可有效支持软件实现的开源软件或其它的可重用软件资源**
- **方法**
  - ✓基于软件开发平台（如编程语言环境）所提供的软件开发包来寻找可重用的软件资源，或者到开源软件托管平台（如Github、Gitee）寻找合适的开源软件
- **要求**
  - ✓结合软件开发的各种约束和限制来理解软件设计的约束和限制，尽可能多的找到可支持软件实现的可重用软件资源
- **结果：无**

问题和讨论