《软件体系结构与设计》
第五章
作业 1

班　　级：111171
学　　号：20171000970
学 生 姓 名：董安宁
指 导 教 师：尚建嘎

中国地质大学地理与信息工程学院软件工程系

2019 年 10 月

第 5 章 软件体系结构风格

作业

一 属性驱动设计方法 ADD、基于模式的设计方法中哪些步骤会涉及构件级（模块化）设计方法？

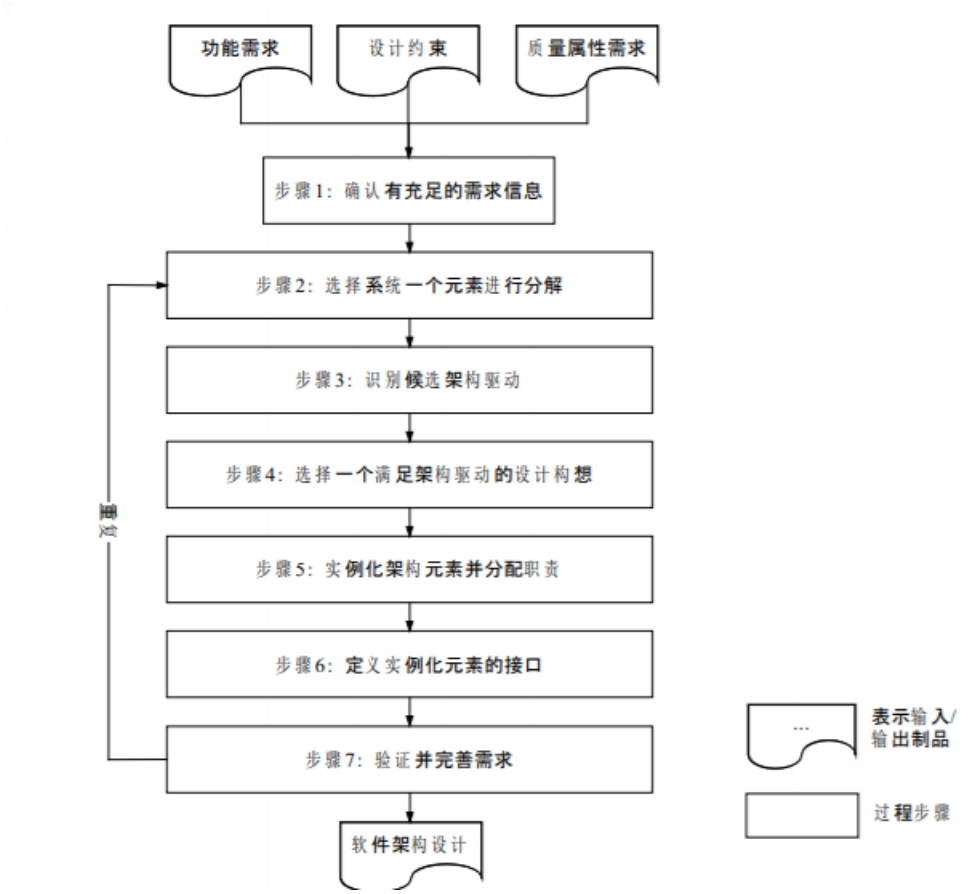

图 1 ADD 设计的步骤总览

我认为每一步都被设计成了一个构件，在进行设计的时候按照这种模块化的方式接续进行就可以完成设计。不过我认为步骤 2，步骤 6，步骤 8 被封装的最好。

步骤 2 是选择系统一个元素进行分解，选择的步骤从来都是一样的，即将一个大整体分为两部分，一部分是被选择出来的，一部分是未进一步选择的，在递归进行 ADD 设计的时候，这一步可以无需多少改动，作为一个模块插入到分解步骤中，因此是一个模块。

ADD 设计步骤 2——选择系统一个元素进行分解

在步骤 2 中，将所选择的系统元素作为后续步骤的 设计重点

当首次执行该步骤时，唯一可以分解的元素是系 统本身（大泥球），那么就可将所有需求分配给 该系统

已经将系统分为两个或者多个元素，也将需求分 配给这些元素。那么就需要从中选择一个元素作 为后续步骤的重点

架构师可根据这些质量属性需求选择合适的设计 模式和策略

步骤 6 也被模块化的很好，由于接口是在拆分的时候所呈现的，只需要发现接口所连接的两个构件之间的参数传递关系，就可以了，因此也可以比较高程度地封装成构件被使用。

ADD 设计步骤 6—— 定义实例化元素的接口

♣ 一个接口可能包括以下任何一项：

- 操作语法（如签名）
- 操作语义（如描述，前置条件，后置条件，约束）
- 信息交换（如信号事件，全局数据）
- 个体元素或操作的质量属性需求 • 错误处理

步骤 8 也是高度模块化的步骤，只需要记录上面 1~7 步骤执行的过程后，链接到下一次 1~7 的过程，起到了桥梁的作用，也因此被封装的很好。

ADD 设计步骤 8—— 分解系统其它元素，重复步骤 2 至 7

完成步骤 1-7，就已经将 1 个父元素分解成多个子 元素。每个子元素都是一个职责的集合，包括接口描述、功能性需求、质量属性需求以及设计约束

现在可以返回到步骤 2 的分解过程，继续选择下一 个元素进行分解

二 消除循环依赖通常有哪些方法？请举例说明。

循环依赖会导致在测试、维护、理解的时候出现各种无法解释的问题，如类似“先有鸡还是先有蛋”的这类问题，以及无法预测某个包的变化将会如何影响其他包，有以下两种方式消除循环依赖：

① 创建新包

② 利用 DIP(依赖倒置) 和 ISP(接口隔离)原则

1 若在逐层调用的时候，最下面的包要返回最上面的包消息，就会相乘依赖循环，可以通过添加消息管理的包，使最顶部的包和返回消息的包都对其产生依赖，这样就避免了循环，如下图所示：

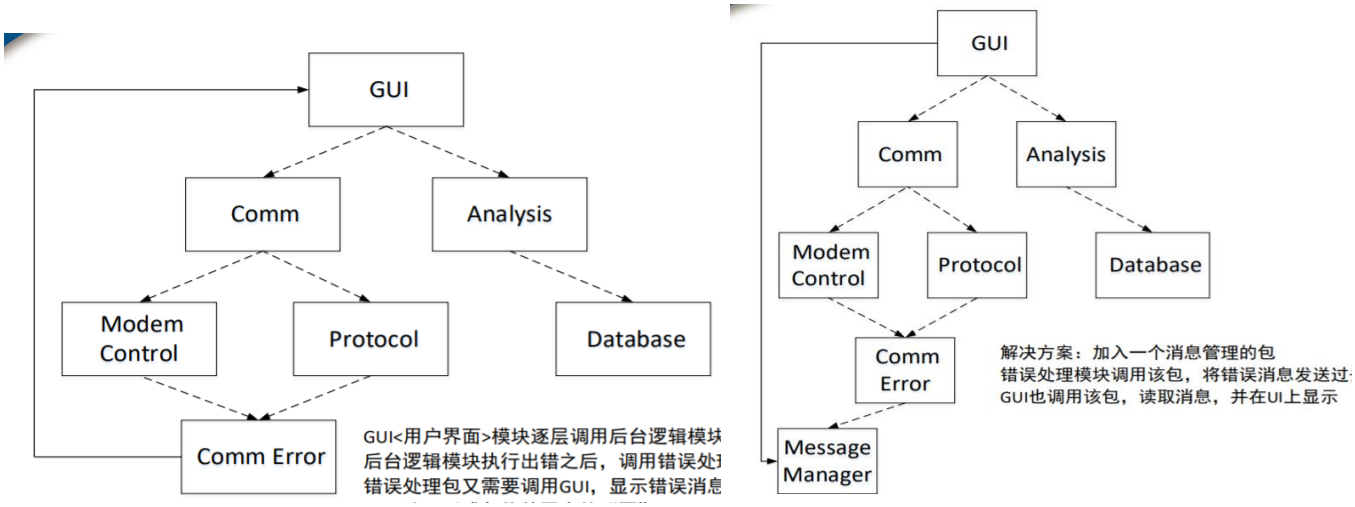

图 2 添加包解决循环

2 若存在双向依赖，使得依赖 C 和 B 的控件 A 又经过 C 依赖了 B，可以通过添加抽象类的方式，将依赖进行反转，从而消除圈，如下图所示：

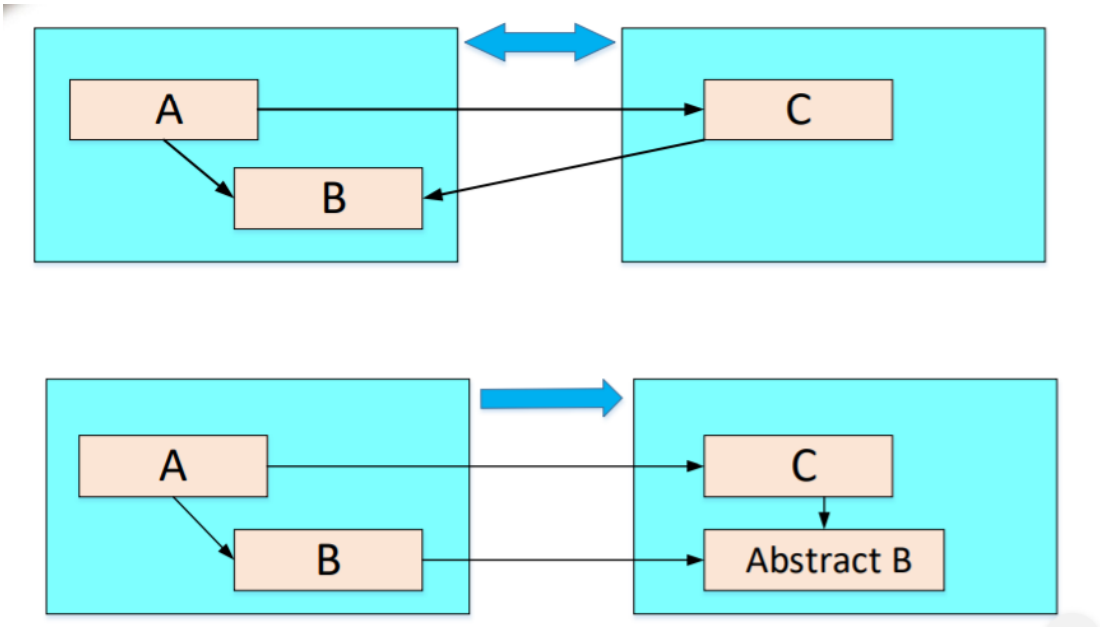

图 3 创建抽象类解决循环

三 回顾您曾经参与的或熟悉的一个软件项目，谈谈软件设计尤其是体系结构设计方面存在什么问题？针对软件的一部分或全部，选择属性驱动设计方法、基于模式的设计方法、模块化设计方法中的一种或多种的结合进行软件体系结构设计练习？

我曾经做过的基于 B/S 的三层架构（即软件体系结构实习二），在设计结构的过程中，对于不同层之间的通信设计有问题。我使用的是基于 SQS 队列的通信，由于队列消息需要主动接收，导致每一层之间的通信都是一种异步的格式，最坏的情况下五秒后才会收到消息。这样就降低了用户对响应的反馈。

用基于模式的设计方法对该问题进行分析：

1. **选择一个需要细化设计的构件：**

   层与层之间的通信构件。

2. **为该构件定义需求**

   该构件将部署在每一层，交互过程为：每一层的系统都需要传入指定格式的数据，该构件便会向指定的层发送消息；当构件收到来自别的层的消息的时候，也会将数据传递给所在层。

3. **针对步骤 2 中定义的需求和交互，找出最合适的结构风格或模式**

   3.1 指定问题

   目前该构件运行符合基本要求，可以对传入的数据进行发送，以及接收数据，但是构件发送和接收的行为仍是异步通信，导致响应速度过慢。因此需要解决的问题就是发送和接收时过慢的问题。

3.2 选择模式类别

由于是构件级别的修改，因此需要寻找一种设计模式来解决该问题。

3.3 选择问题类别

该问题属于设计模式问题类别中的行为型。

3.4 比较问题描述

针对该构件在不同层之间的通信问题，可以缩小范围到进程间的通信方式上。

可选择的通信方式：

① HTTP 通信

② TCP/UDP 套接字通信

③ 消息队列通信

3.5 比较优点和不足

HTTP 通信是应用层的通信，是在 TCP 协议的基础上进行的通信，在稳定的基础上传输的信息也更容易封装，首选。

TCP 协议偏向于底层，对不同的数据可能需要分别设计不同的处理方式，虽然可能相比之下更能直接通信，但需要为此添加许多处理数据的代码，造成冗余。次选。

消息队列对传输信息的封装也很好，但是其接收和发送消息是异步的，在响应时间层面不如上面两种。次选

3.6 选择最佳模式变种

在上述的几个方法中可以通过增加或者减少功能，形成满足要求的变种模式。

3.7 选择其他的问题类别

暂时不需要。

4. 使用与问题相匹配的模式来指导类和构件的设计

在上述分析的基础上进行类和构件的设计，通过测试寻找最优的解决方案。

5. 在构件中进行迭代

为每个构件重复“定义需求”，“寻找风格”，“指导设计”这几个步骤，直到设计出合乎要求的构件。