软件设计师

--软件设计概述

第十三章 软件设计概述

- 13.1 软件设计基本原则
- 13.2 结构化设计方法
- 13.3 面向对象设计
- 13.4 用户界面设计
- 13.5 设计评审
- 13.6 设计模式

一、软件设计基本原则

1．模块

是指执行某一特定任务的数据结构和程序代码。

- 将模块的接口和功能定义为其外部特性
- 将模块的局部数据和实现该模块的程序代码称为内部特性。

在模块设计时，最重要的原则就是实现信息隐蔽和模块独立。

2．信息隐蔽

将每个程序的成分隐蔽或封装在一个单一的设计模块中，并且尽可能少地暴露其内部的处理过程。

信息隐蔽可以提高软件的可修改性、可测试性和可移植性。

3．模块独立

模块独立是指每个模块完成一个相对独立的特定子功能，并且与其他模块之间的联系最简单。

通常用耦合（模块之间联系的紧密程度）和内聚（模块内部各元素之间联系的紧密程度）两个标准来衡量，我们的目标是“高内聚、低耦合”。

4．内聚

指模块内部各元素之间联系的紧密程度。模块的内聚类型分为7种，根据内聚度从高到低的排序。

内聚类型	描述
功能内聚	完成一个单一功能，各个部分协同工作，缺一不可
顺序内聚	处理元素相关，而且必须顺序执行
通信内聚	所有处理元素集中在一个数据结构的区域上
过程内聚	处理元素相关，而且必须按特定的次序执行
瞬时内聚	所包含的任务必须在同一时间间隔内执行（如初始化模块）
逻辑内聚	完成逻辑上相关的一组任务
偶然内聚	完成一组没有关系或松散关系的任务

(1)功能内聚：指模块内所有元素共同完成某一功能，联系紧密，缺一不可，是最强的内聚类型。

(2)顺序内聚：指一个模块中各个处理元素都密切相关于同一功能且必须顺序执行，前一功能元素输出是下一功能元素的输入。即一个模块完成多个功能，这些模块又必须顺序执行。

(3)通信内聚：指模块内所有处理元素都在同一个数据结构上操作，或者指各处理使用相同的输入数据或者产生相同的输出数据。

(4) 过程内聚：构件或者操作的组合方式是，允许在调用前面的构件或操作之后，马上调用后面的构件或操作，即使两者之间没有数据进行传递。

(5)时间内聚：把需要同时执行的动作组合在一起形成的模块为时间内聚模块，所有的动作需在同一个时间段内执行。

(6)逻辑内聚：把几种相关的功能组合在一起， 每次被调用时，由传送给模块参数来确定该模块应完成哪一种功能

(7)偶然内聚：模块内各部分之间没有联系，或者有联系，这种联系也很松散，是内聚度最低的模块。

例：某模块实现两个功能：向某个数据结构区域写数据和从该区域读数据。该模块的内聚类型为（ D ）内聚。

A. 过程　　B. 时间　　C. 逻辑　D. 通信

5 .耦合

指模块之间联系的紧密程度。模块的耦合类型分为7种，根据耦合度从低到高排序。

耦合类型	描述
非直接耦合	没有直接联系，互相不依赖对方
数据耦合	借助参数表传递简单数据
标记耦合	一个数据结构的一部分借助于模块接口被传递
控制耦合	模块间传递的信息中包含用于控制模块内部逻辑的信息
外部耦合	与软件以外的环境有关
公共耦合	多个模块引用同一个全局数据区
内容耦合	一个模块访问另一个模块的内部数据；一个模块不通过正常入口转到另一模块的内部；两个模块有一部分程序代码重叠；一个模块有多个入口

(1)非直接耦合：两个模块之间没有直接关系，它们之间的联系完全是通过主模块的控制和调用来实现的。这种模块的耦合度最低、模块独立性最强。

(2)数据耦合：指两个模块之间有调用关系，传递的是简单的数据值，相当于高级语言的值传递。

(3)标记耦合：指两个模块之间传递的是数据结构，如高级语言中的数组名、记录名、文件名等这些名字即标记，其实传递的是这个数据结构的地址。

(4)控制耦合：指一个模块调用另一个模块时，传递的是控制变量（如开关、标志等），被调模块通过该控制变量的值有选择地执行模块内某一功能。

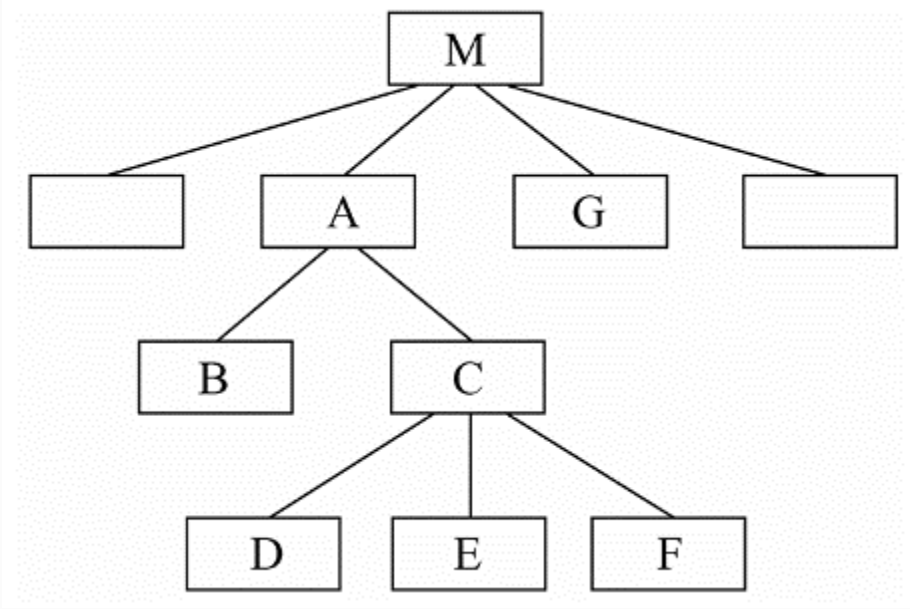
(5)外部耦合：一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息。

(6)公共耦合：指一组模块都访问同一个公共数据环境，如全局数据结构，共享通信区。

(7)内容耦合：一个模块直接使用另一个模块的内部数据，或通过非正常入口而转入另一个模块内部。是最差的耦合。

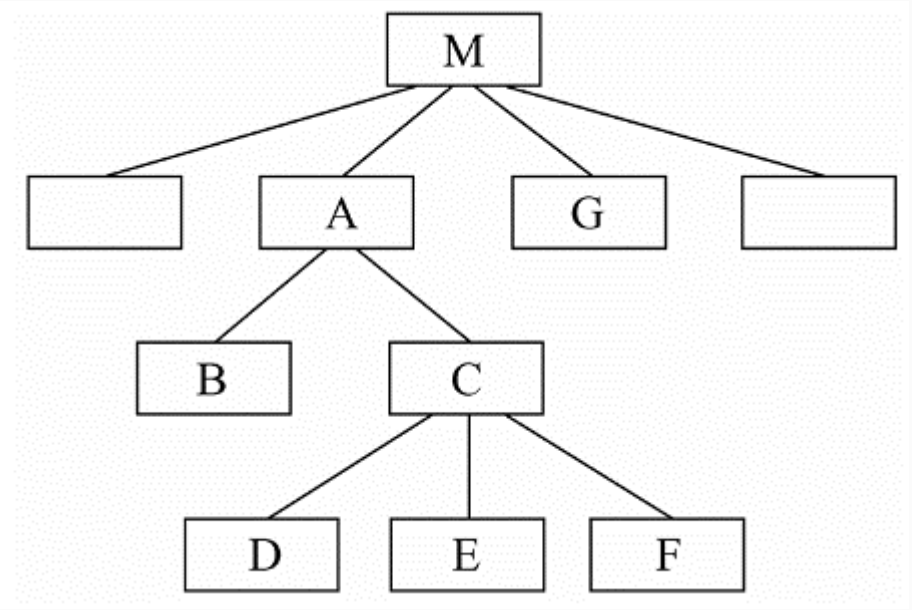
6 .深度

表示软件结构中控制的层数，它往往能粗略地标志一个系统的大小和复杂程度。如果层数过多则应该考虑是否有许多管理模块过分简单，能否适当合并。



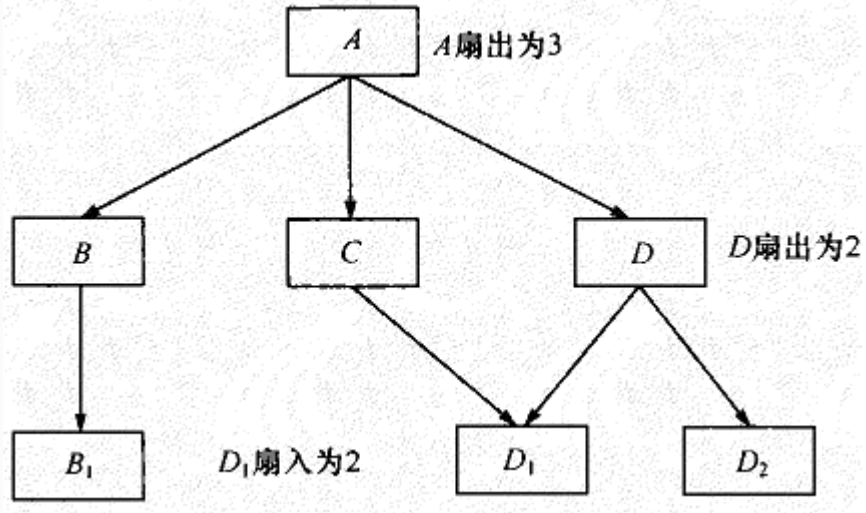
7 .宽度

是软件结构内同一个层次上的模块总数的最大值。宽度越大系统越复杂。对宽度影响最大的因素是模块的扇出。



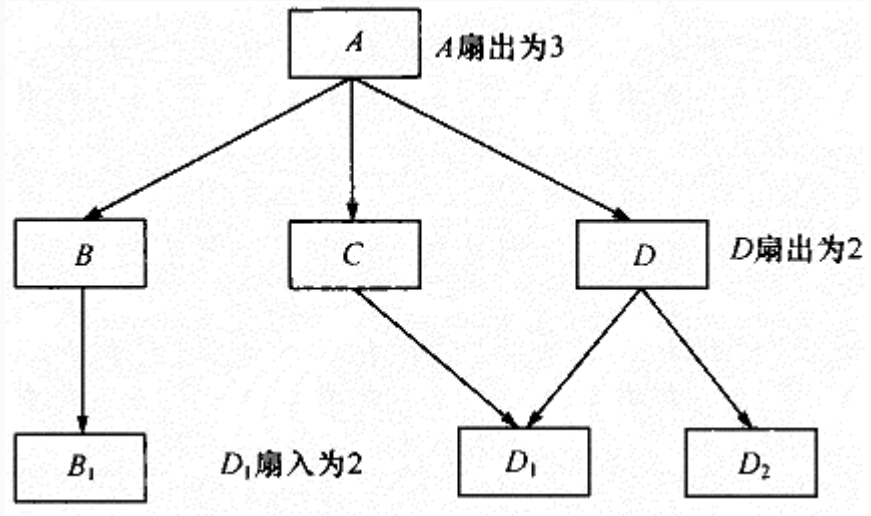
8 .扇出

模块的扇出是指一个模块直接控制(调用)的下层模块数目。扇出过大意味着模块过分复杂，需要控制和协调过多的下级模块；扇出过小也不好。设计得好的系统平均扇出是3或4。



9 .扇入

是指有多少个上级模块调用它，扇入越大则共享该模块的上级模块数目越多。



在模块分解时需要注意：

- 保持模块的大小适中
- 尽可能减少调用的深度
- 直接调用该模块的次数应该尽最多，但调用其他模块的次数则不宜过多(扇入大，扇出小)。好的软件设计结构顶层高扇出，中间扇出较少，底层高扇入。
- 保证模块是单入口、单出口的
- 模块的作用域应该在模块之内
- 功能应该是可预测的