第一章：

软件危机

**软件危机定义：**在计算机软件的开发和维护过程中所遇到的一系列严重问题。

**软件危机的典型表现：**

1.对软件开发成本和进度的估计常常很不准确

2.用户对“已完成的”软件系统不满意

3.软件产品的质量往往靠不住

4.软件常常是不可维护的

5.软件通常没有适当的文档资料

6.软件成本在计算机系统总成本占比逐年上升

7.供不应求

**产生软件危机的原因：**

1.与软件本身特点有关

2.软件开发与维护的方法不正确有关

**消除软件危机的途径：**

1.对计算机软件有一个正确的认识。

2.充分认识到软件开发是各类人员协同配合，共同完成的工程项目。

3.使用在实践中总结出来的开发软件的成功的技术和方法，并且研究探索更好更有效的技术和方法。

4.应该开发和使用更好的软件工具

**软件工程的七条基本原理**

1.用分阶段的生命周期计划严格管理

2.坚持进行阶段评审

3.实行严格的产品控制

4.采用现代程序设计技术

5.结果应能清楚地审查

6.开发小组的人员应该少而精

7.承认不断改进软件工程实践的必要性

**软件工程方法学三要素**

1.方法：完成软件开发的各项任务的技术方法，回答“怎样做”的问题。

2.工具：为运用方法而提供的自动的或半自动的软件工程支撑环境。

3.过程：为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

质量：方法，工具和过程必须围绕质量开展。

软件生命周期

软件生命周期由软件定义、软件开发和运行维护(也称软件维护)3个时期组成，每个时期又进一步划分成若干个阶段。

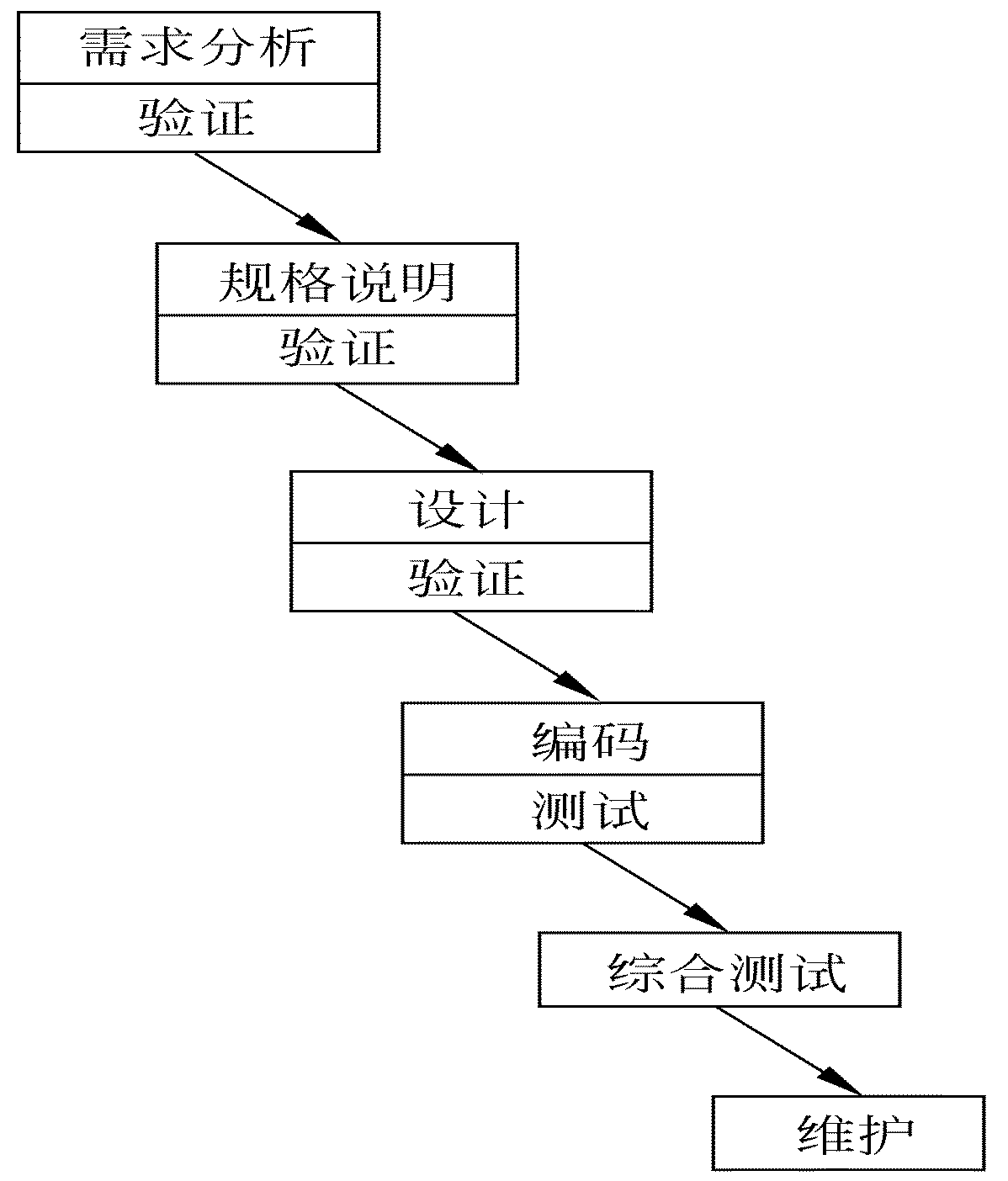
软件定义：问题定义、可行性研究、软件需求分析

软件开发：总体设计、详细设计、编码和单元测试、综合测试

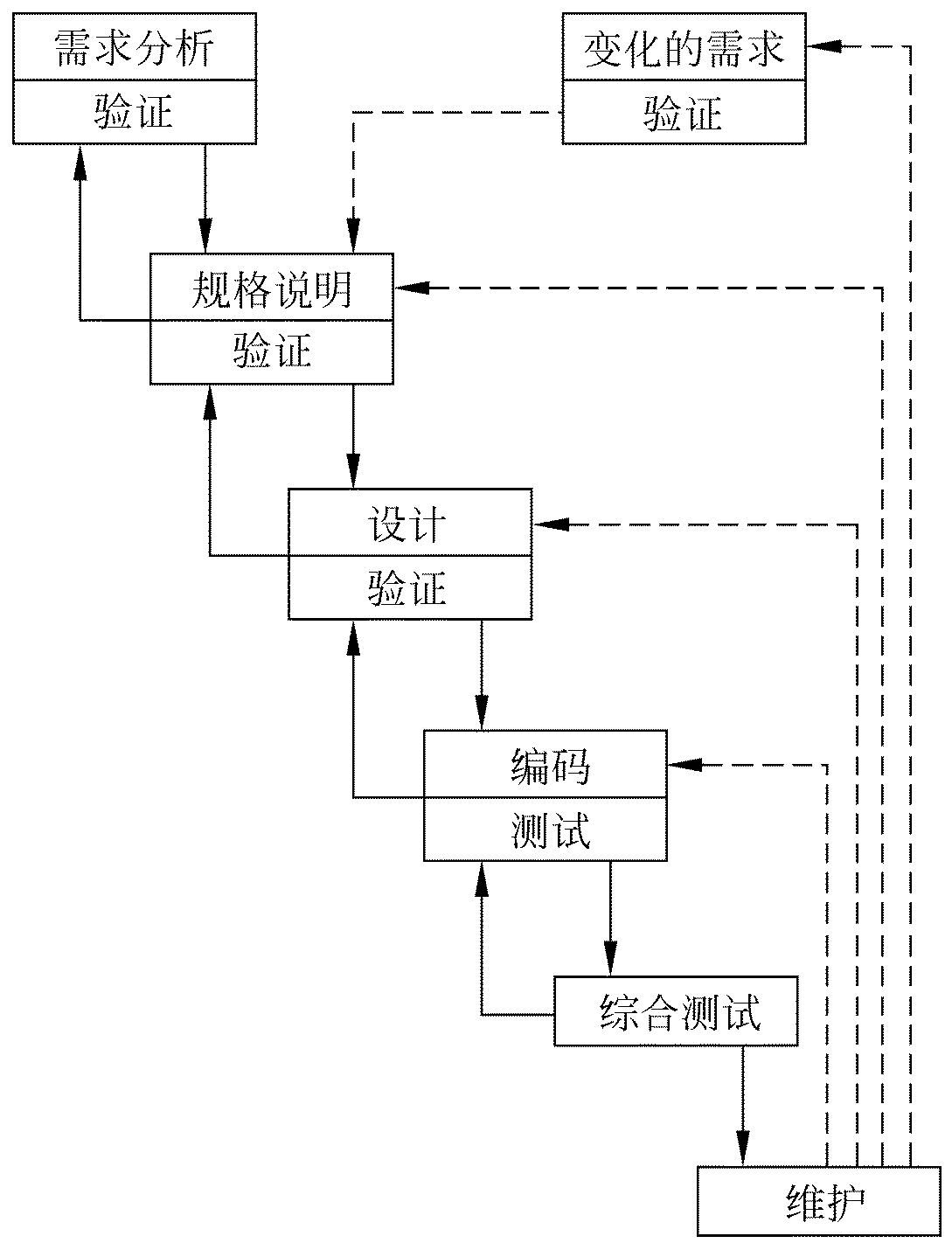
运行维护：维护

软件过程模型（除喷泉和微软模型）

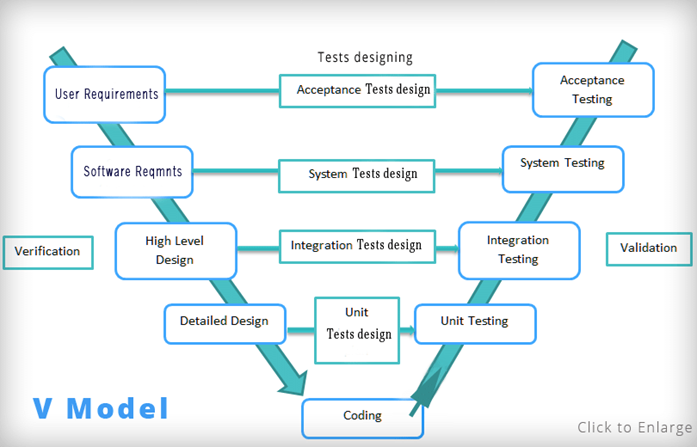
瀑布模型：



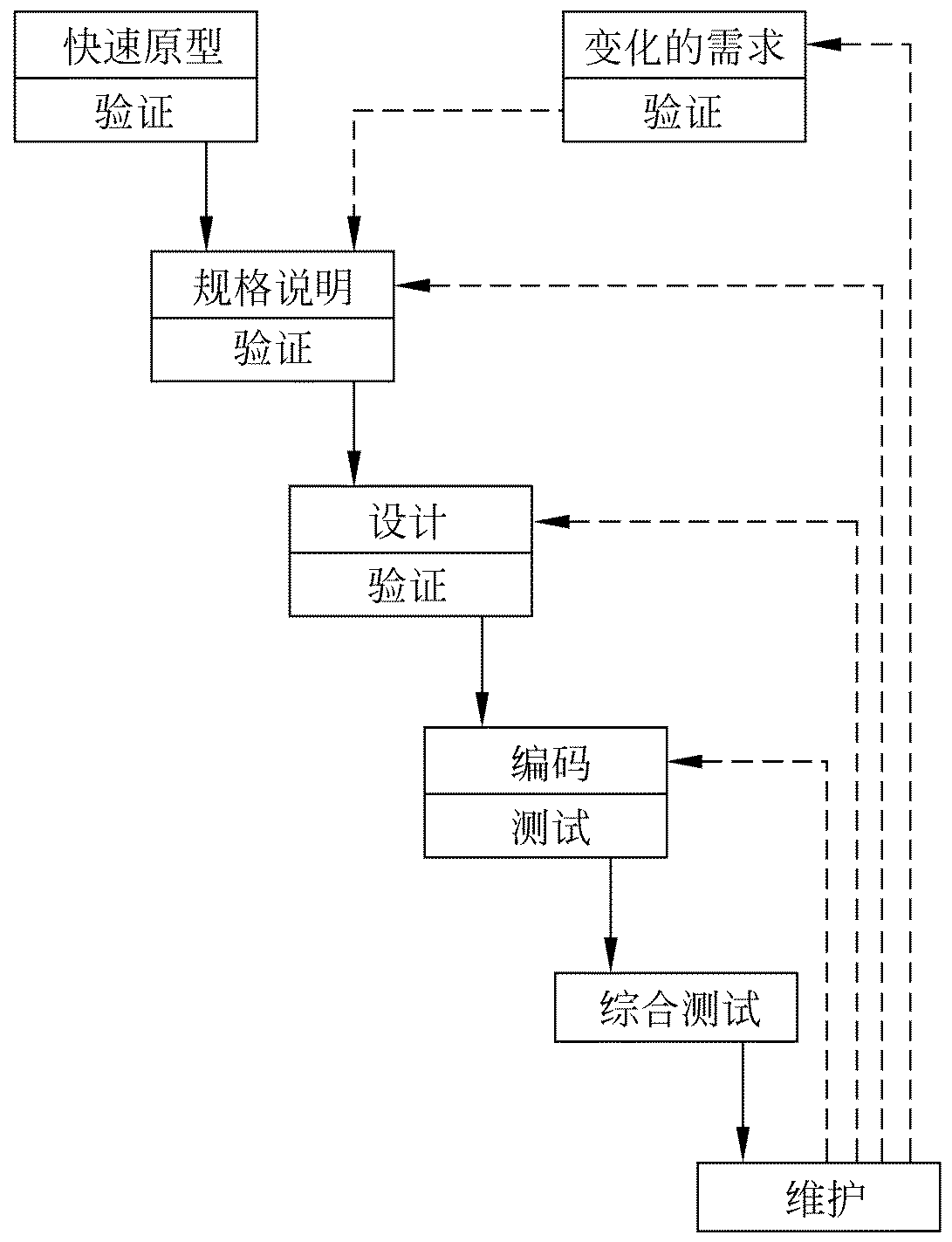
实际的瀑布模型：



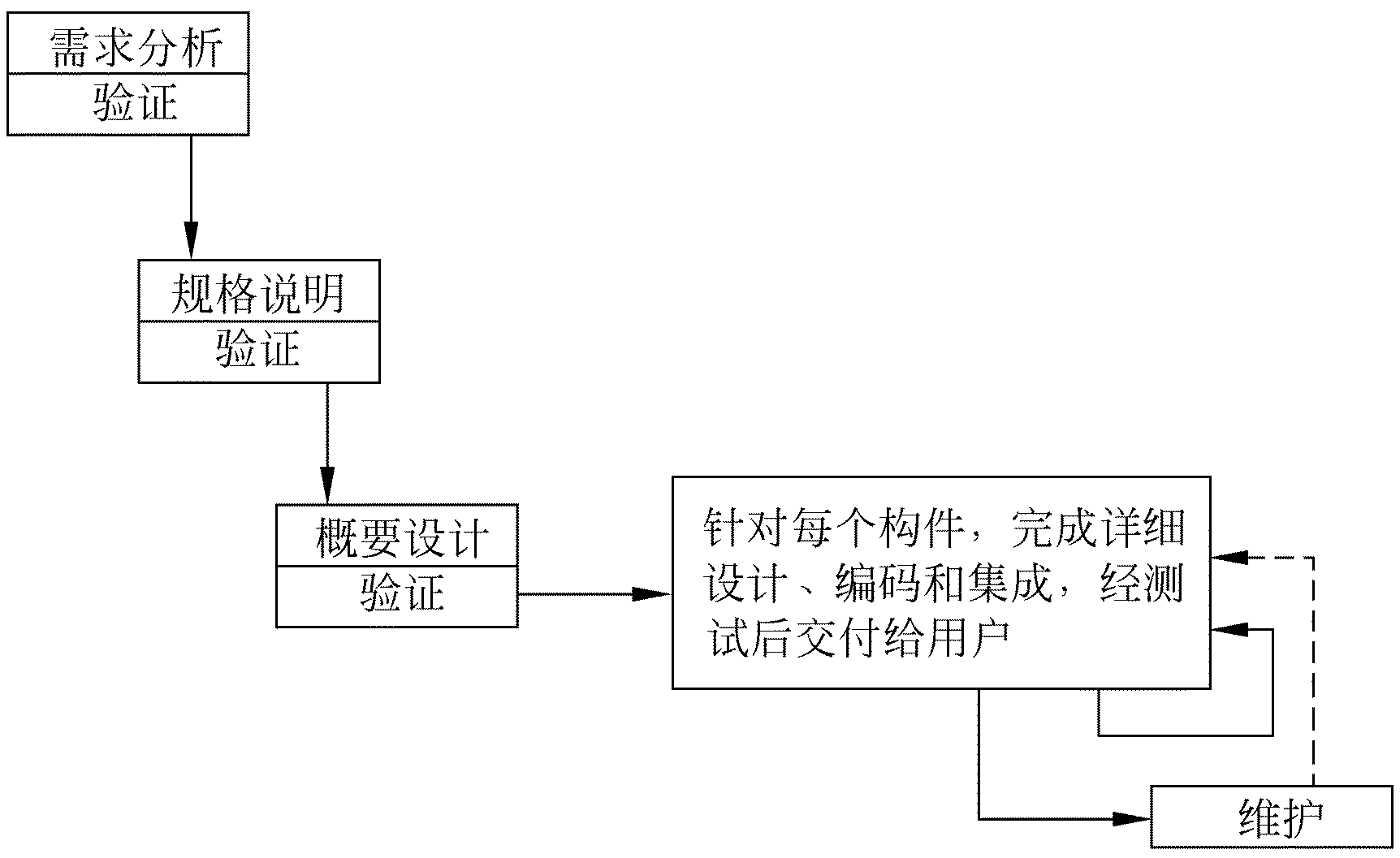
瀑布模型变形：V模型



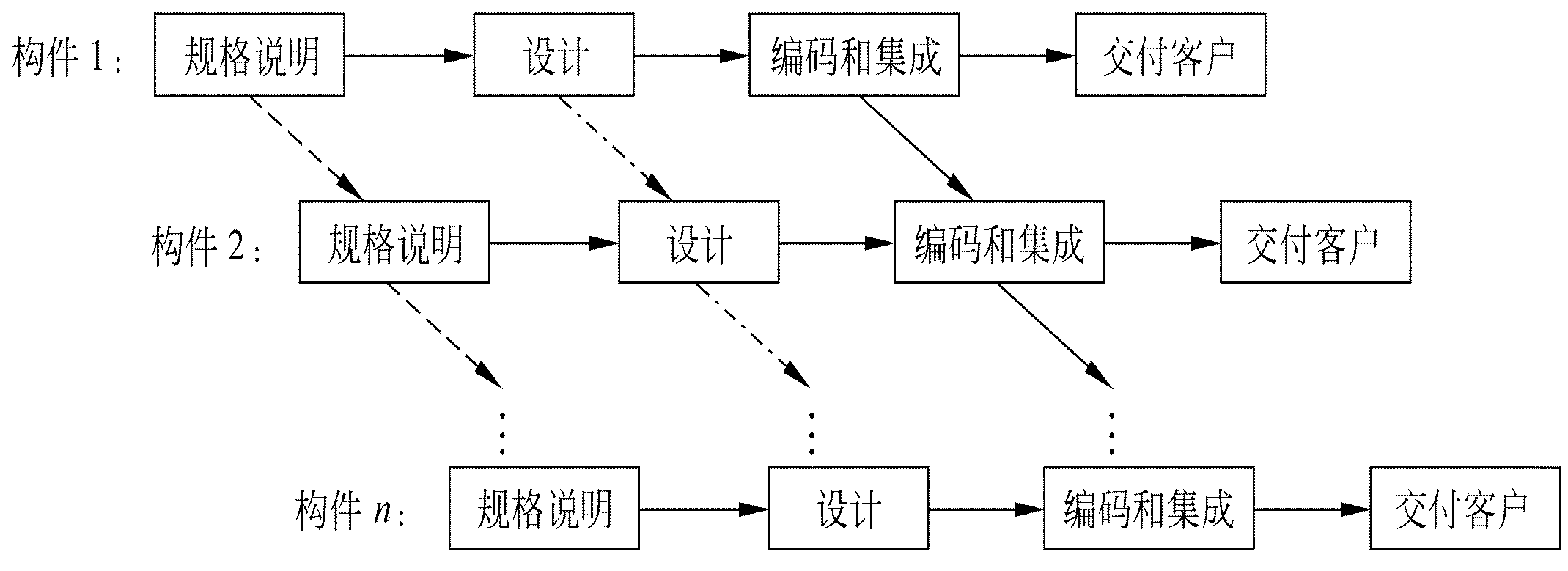
快速原型模型：



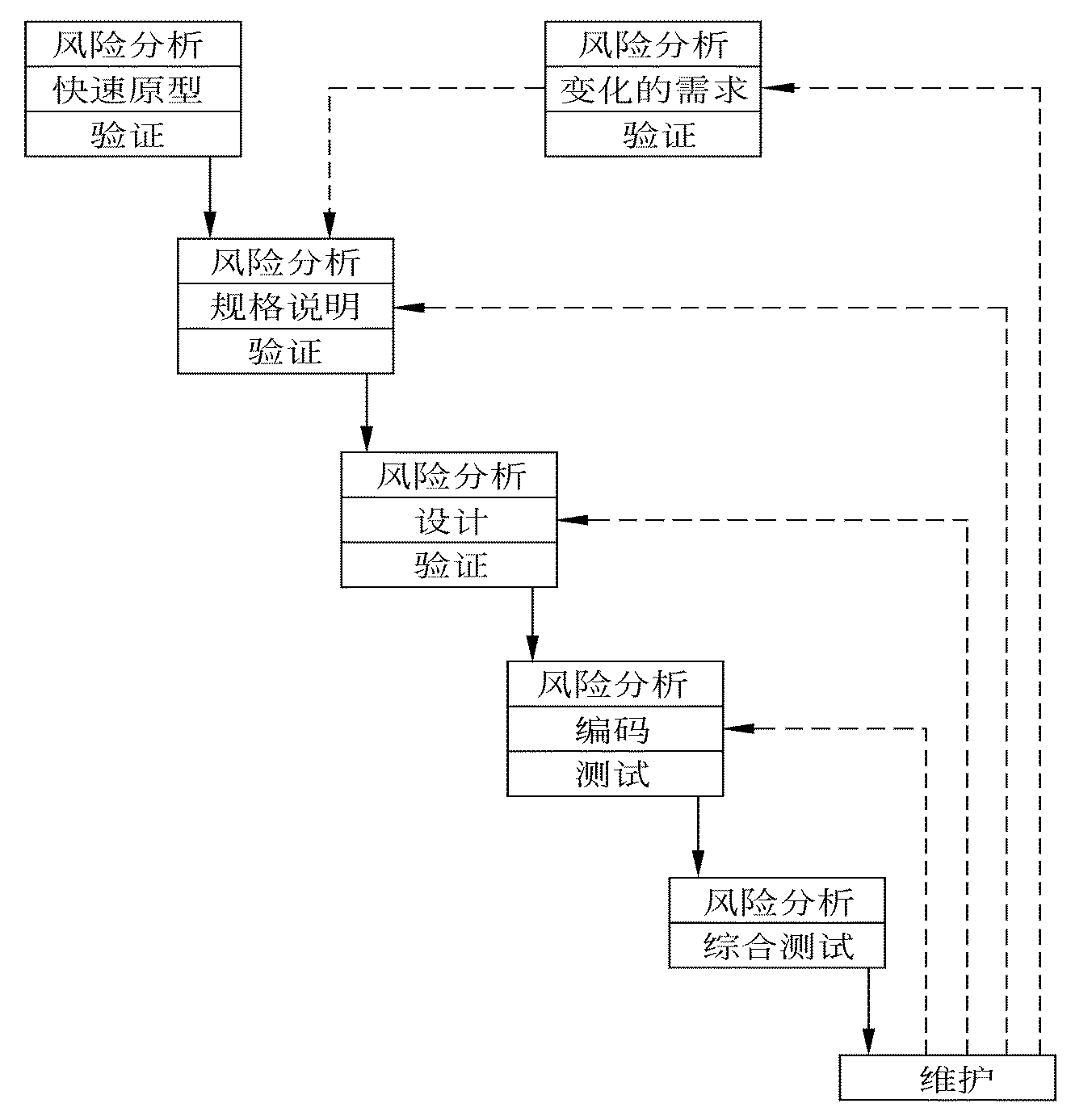
增量模型：



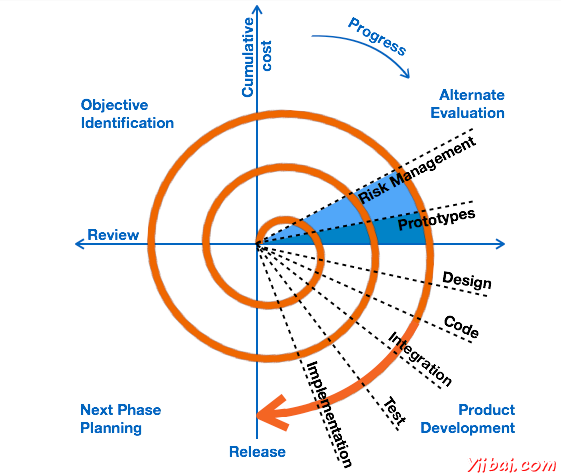
风险更大的增量模型：



螺旋模型（快速原型模型+风险分析）：



完整的螺旋模型：



第二章：

可行性研究

1.技术可行性：相关技术的发展是否支持这个系统。

2.经济可行性：进行开发成本以及效益的估算，确定项目是否值得投资开发。

3.操作可行性：用户组织是否适合目标系统，现有的人员素质能否胜任对目标系统的操作。

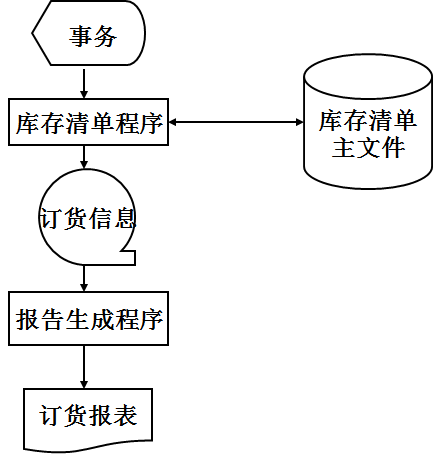
4.社会环境可行性：

（1）市场：市场又分为未成熟的市场、成熟的市场和将要消亡的市场。

（2）政策：政策对软件公司的生存与发展影响非常大

（3）法律：开发项目是否会引起侵权、破坏或其它责任问题。

系统流程图（主要用于描绘物理系统，注意与程序流程图的不同）



显示 处理 输入/输出 数据流 文档

数据流图（注意分层细化的规则）

四种基本符号：

1.矩形：数据的源点和终点

2.圆角矩形或圆形：数据处理

3.右侧无竖边的开口矩形或两条平行横线：数据存储

4.箭头：数据流

附加符号：

1.\*：与

2.+：或

3.⊕：亦或

数据流命名：

1.代表整个数据流(或数据存储)的内容

2.不要使用缺乏具体含义的名字；

3.除流入流出数据存储的数据流可以命名相同之外，其他的数据流命名需要唯一，即便是同样内容。

数据处理命名：

1.先为数据流命名，再为处理命名，通常1个动词+宾语的命名方式

2.如果必须用两个动词才能描述整个处理的功能，则把这个处理再分解成两个处理可能更恰当些。

3.若处理的命名比较困难可能是处理包含太多功能

易错点：

1.外部实体之间不可能有数据流

2.外部实体不可能直接写数据到数据存储中

3.加工输入输出数据流不可同名

4.加工不可能只有输入或只有输出

数据字典

定义：是关于数据的信息集合，即对数据流图中包含的所有元素定义的集合。

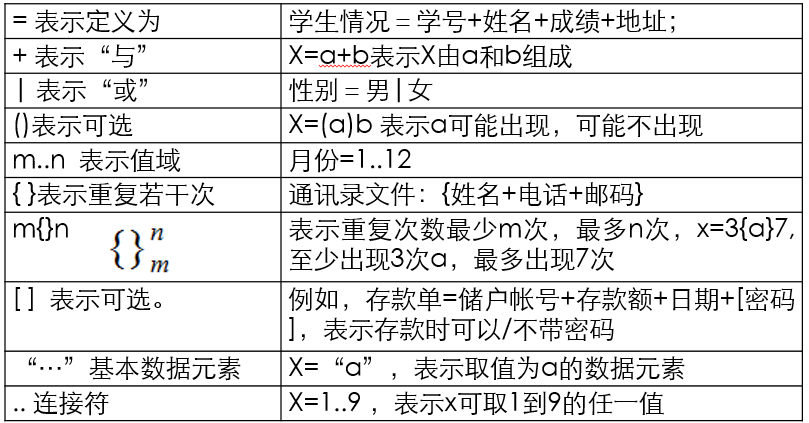
需要定义的四项内容：

1.数据流：来源、去向、组成、流通量

2.数据元素：名称、别名、取值范围、含义、数据长度、小数位数、简单描述

3.数据存储：数据结构及数据存放规则

4.数据处理：数据处理的逻辑功能和主要算法



成本效益分析（中等关注）

常用的成本估算技术

1.代码行技术

2.任务分解技术

3.自动估计成本技术

4.经验统计估计模型

第三章：

需求分析的任务

1.需求收集

2.需求整理

3.需求建模及软件规格说明

4.需求评审和验证

**需求分析应建立的模型**

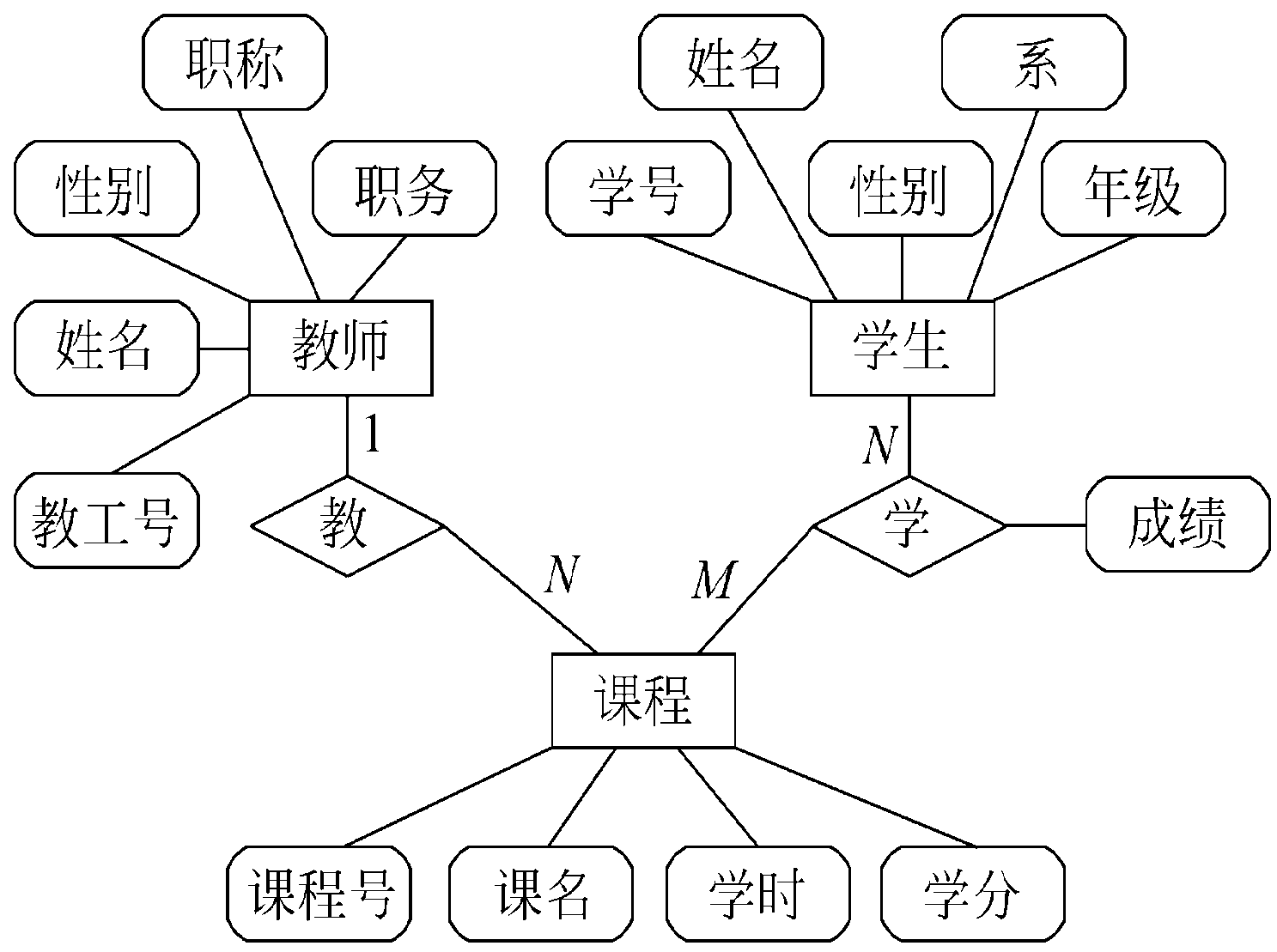
数据模型：E-R图(ERD)，描述数据对象，数据对象之间的关系；

功能模型：数据流图(DFD)，描述数据在系统中流动，变换的逻辑过程；

行为模型：状态转换图(STD)，描绘了系统的各种行为模式和在不同状态之间转换的方式；

模型中心：**数据字典 (DD)**。

ER图（实体-联系图）



数据规范化

范式：（1NF、2NF、3NF定义）

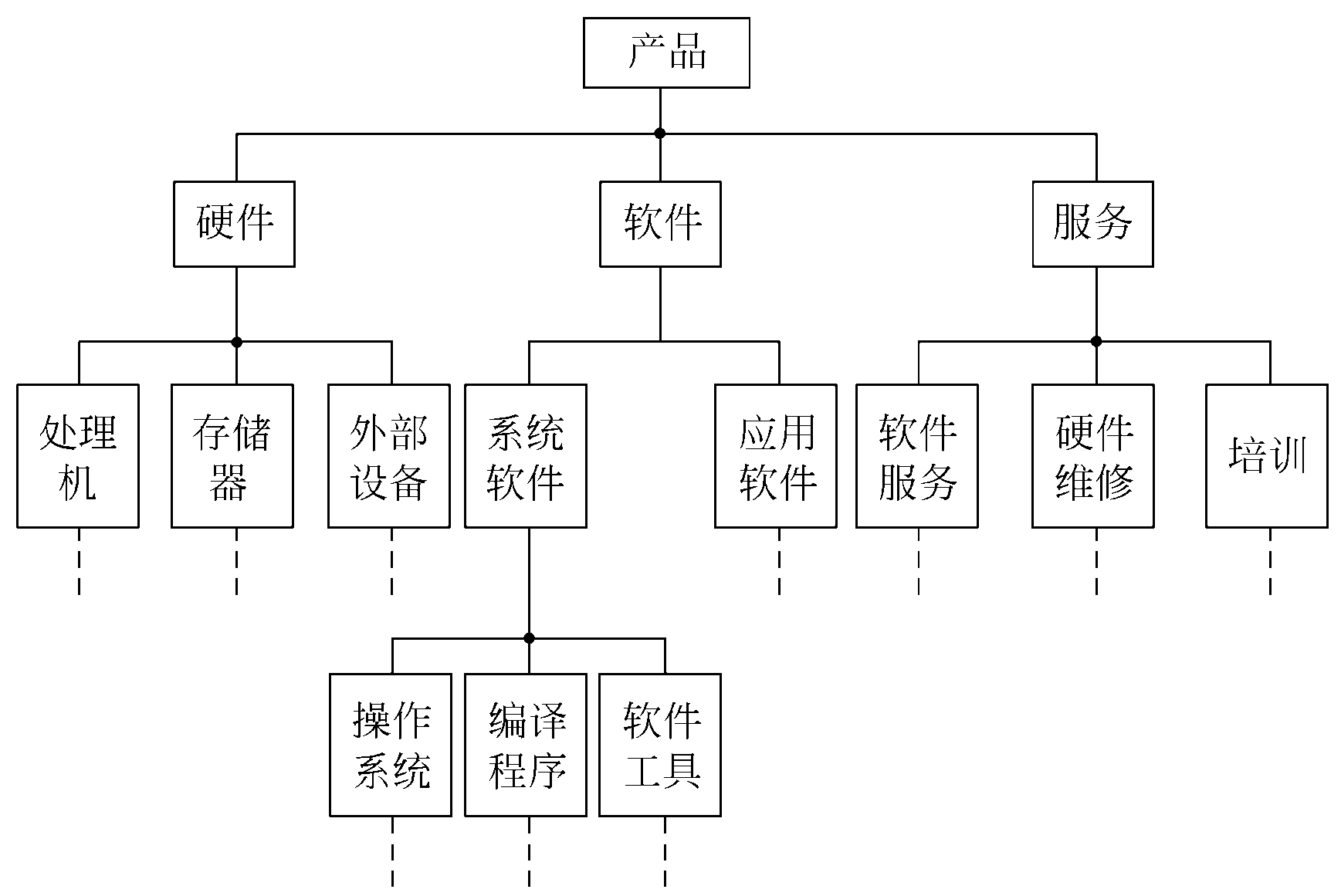
1.第一范式（1NF）：所有属性都是原子值，即不出现“表中有表”

2.第二范式（2NF）：在 1-NF 基础上，每个非主属性都由整个主关键字决定（而非依赖于主关键字的一部分）

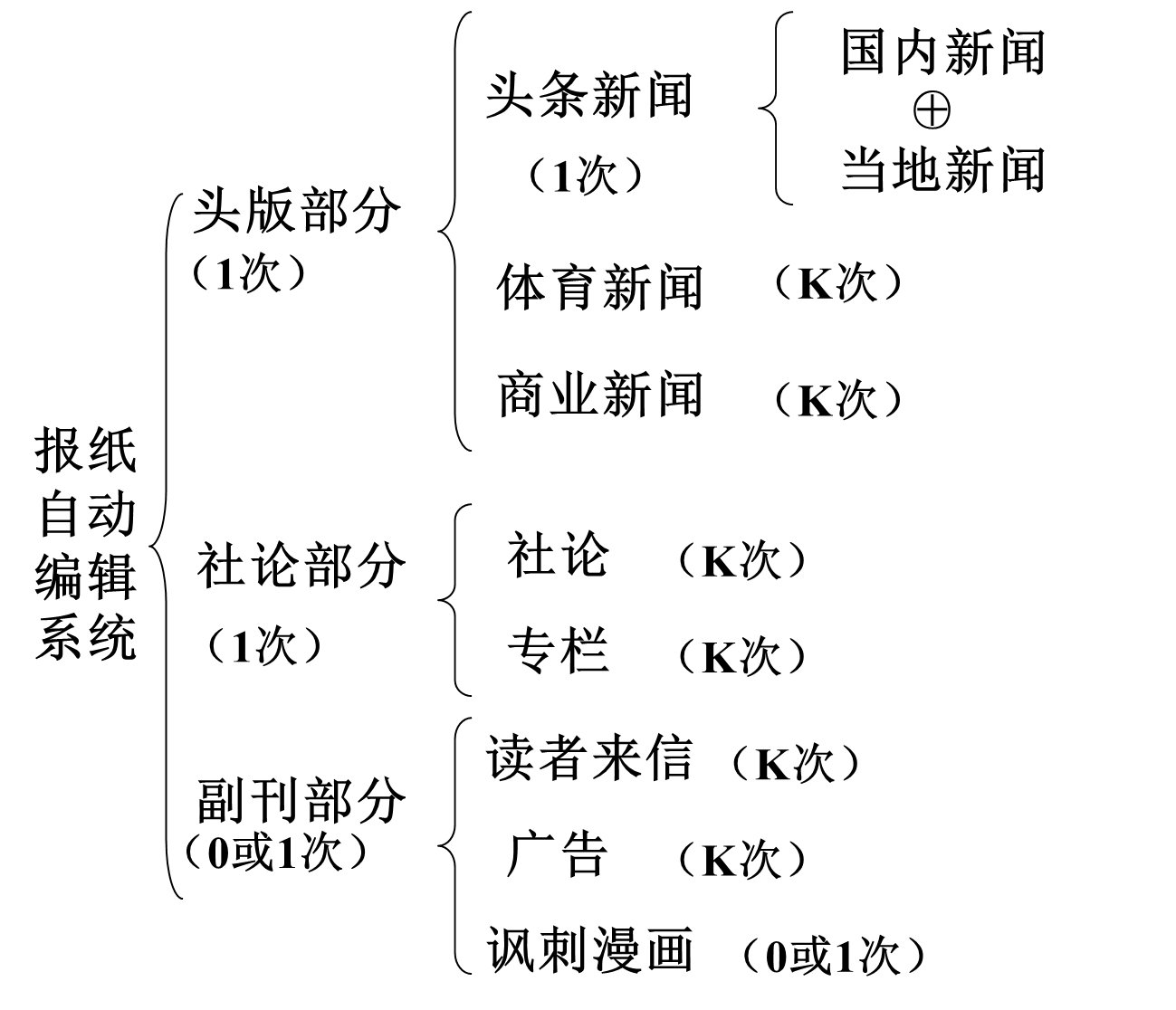
3.第三范式（3NF）：在 2-NF基础上，非主属性之间无依赖关系

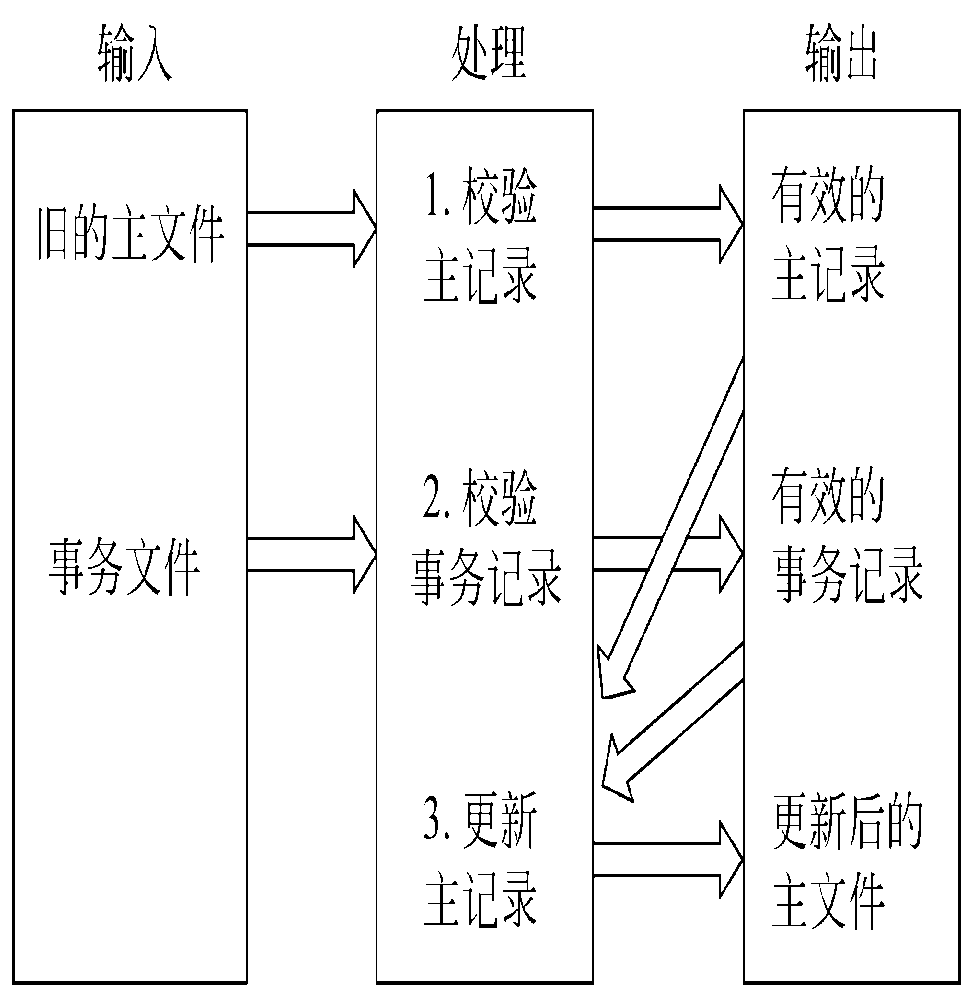
层次方框图（表示数据结构，注意与层次图的区分）

层次图

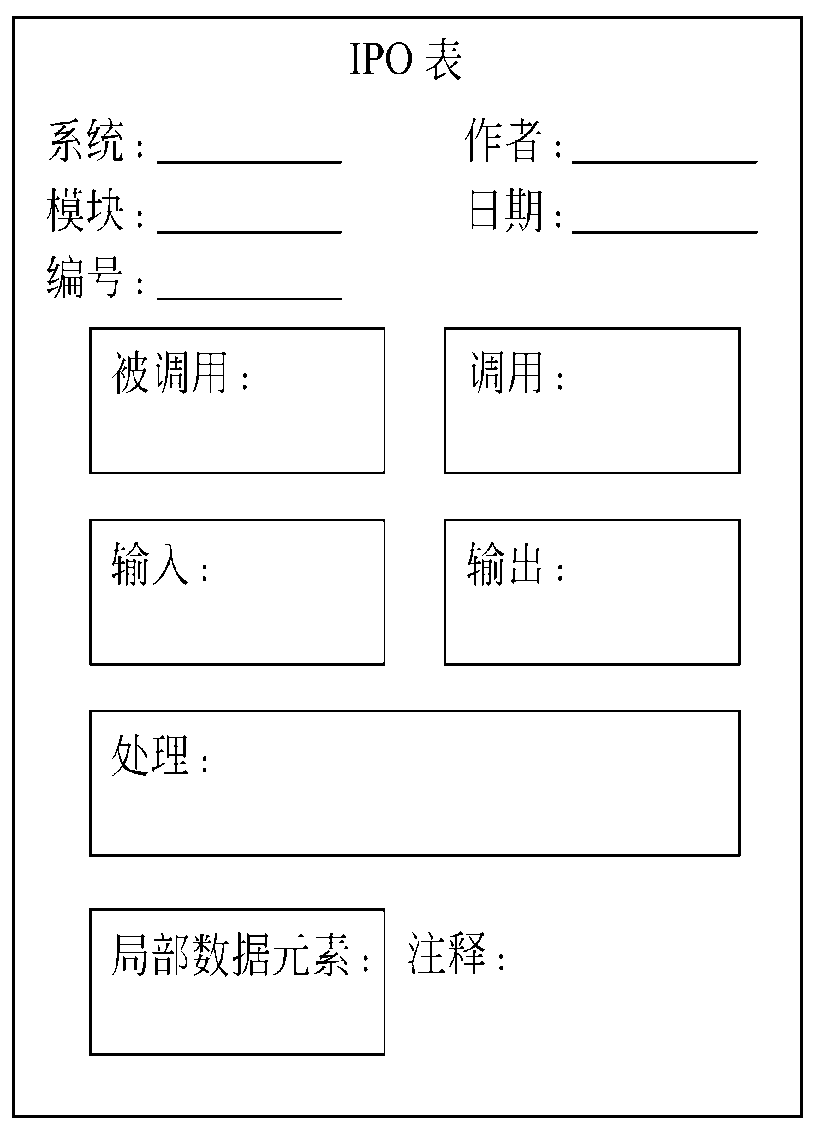


Warnier图

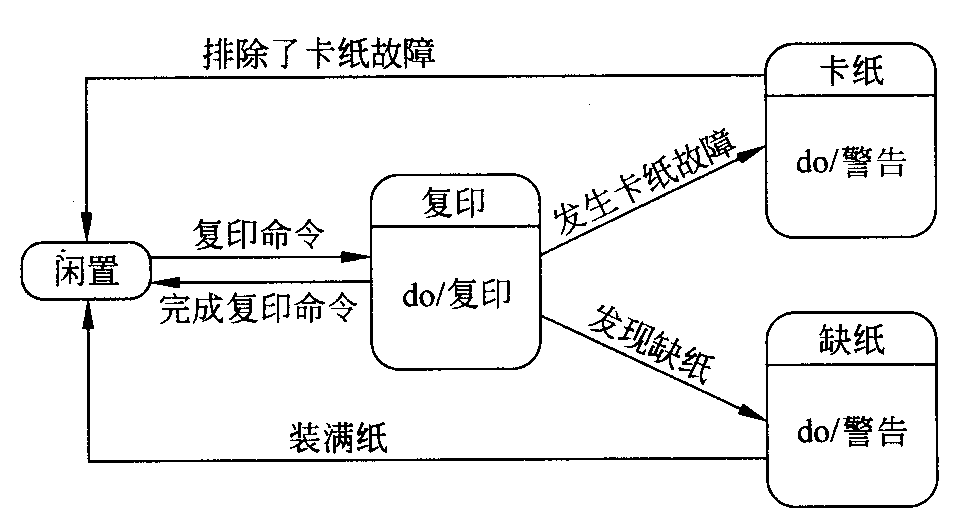


IPO图（输入处理输出图）

IPO表

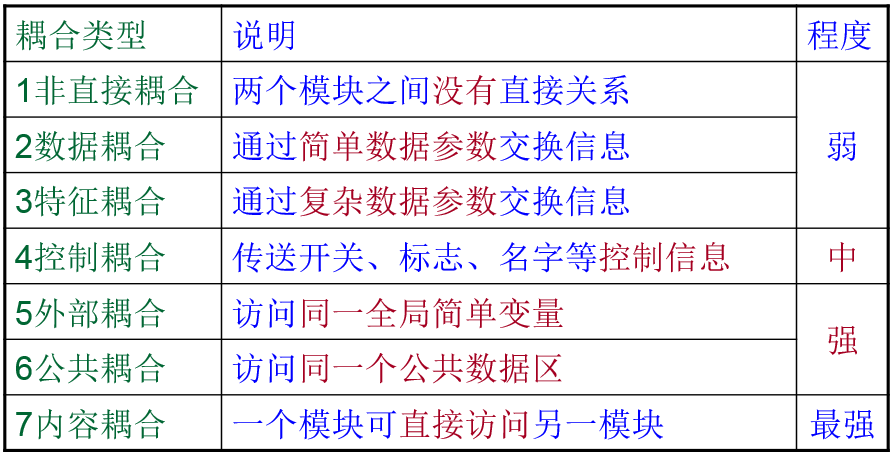


状态转换图=状态迁移图=状态图（注意事件的语法和图形符号）



第五章：

七个等级的内聚、耦合





七条启发规则

（1）改进软件结构提高模块独立性

（2）模块规模应该适中

（3）深度、宽度、扇出和扇入都应适当

（4）模块的作用域应该在控制域之内

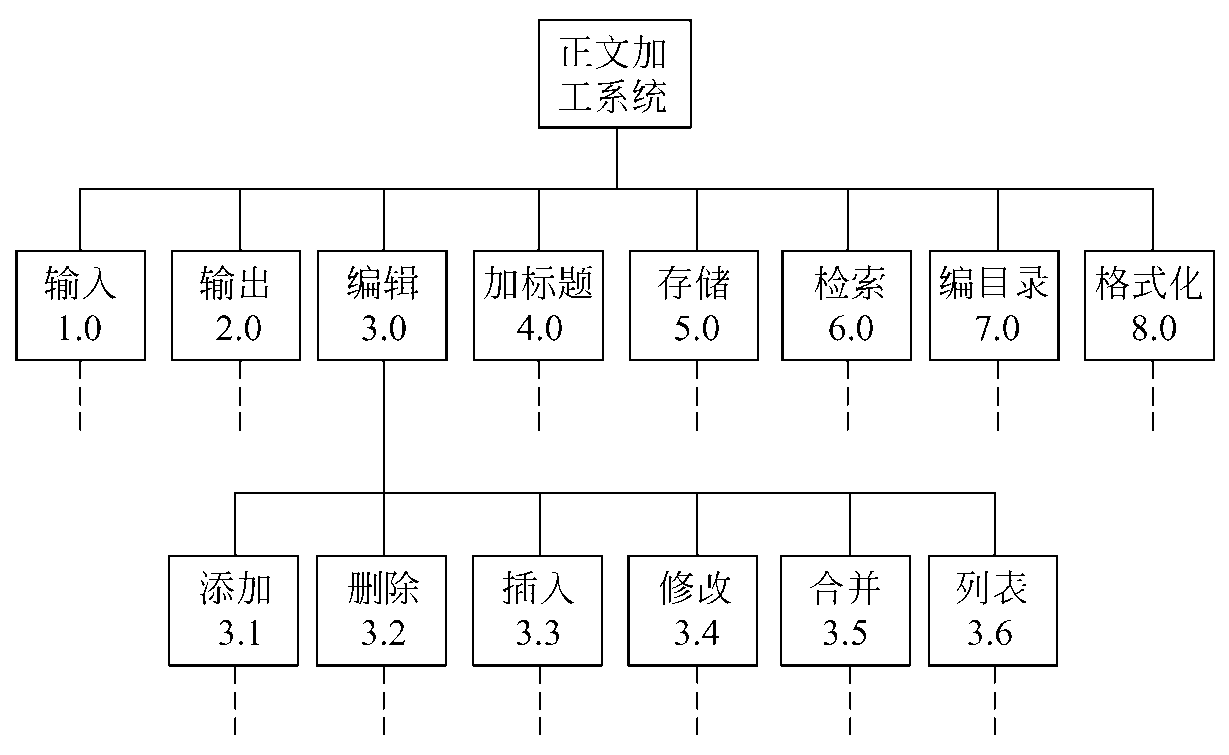
（5）力争降低模块接口的复杂程度

（6）设计单入口单出口的模块

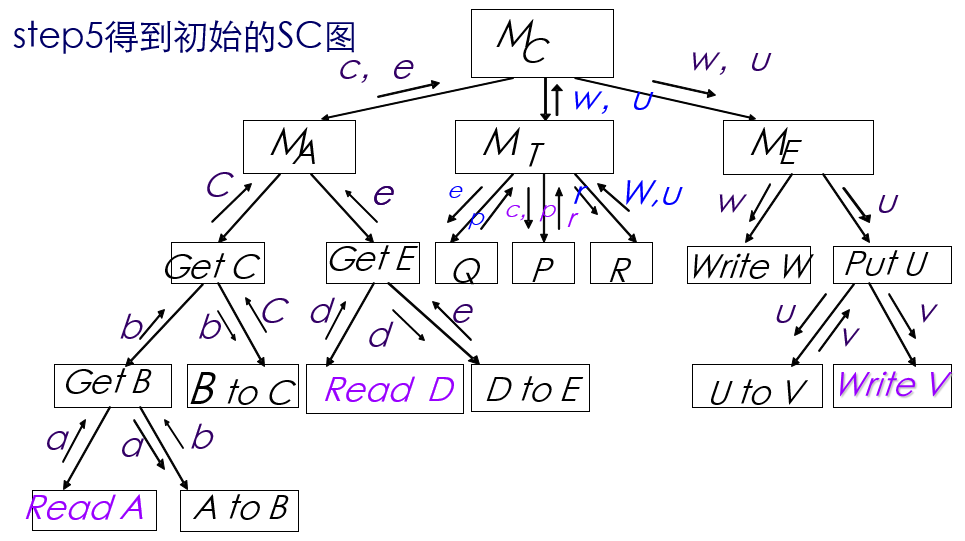
（7）模块功能应该可以预测

描绘软件结构：层次图、HIPO图（层次图+输入处理输出图）、结构图（SC）

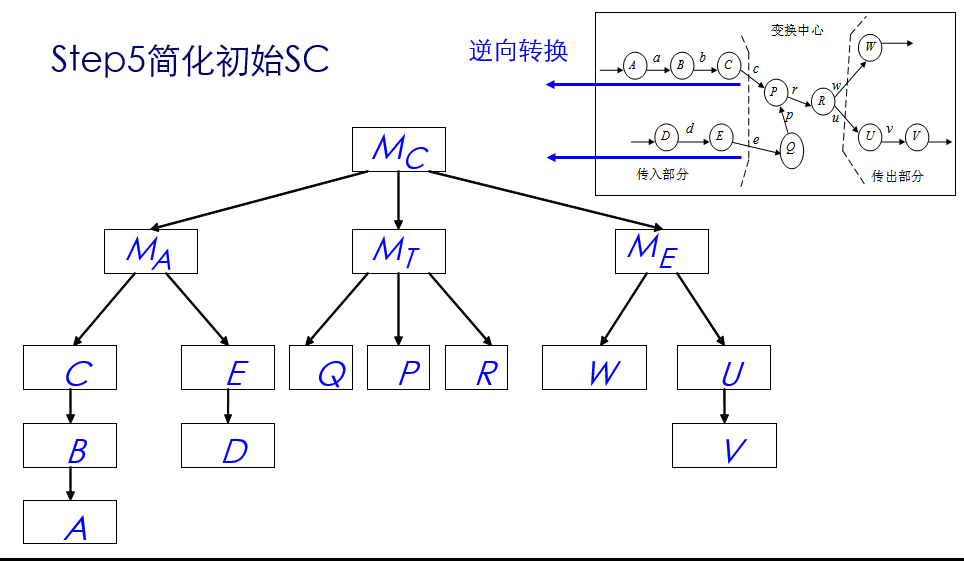
HIPO图：



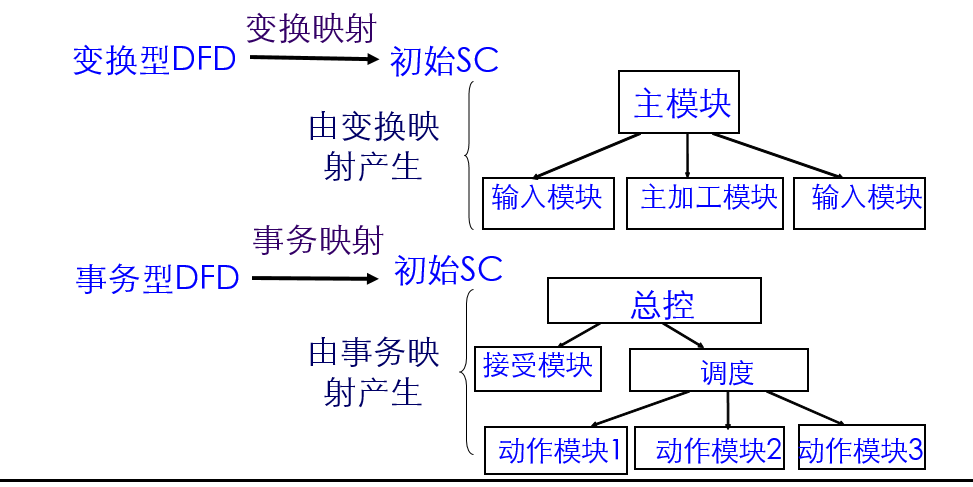
结构图：



结构图简化版：



结构化设计：变换流、事务流，DFD转SC（变换、事务）



第六章：

程序流程图（基本结构）

1.如果只允许使用顺序、IF-THEN-ELSE型分支和DO-WHILE型循环这3种基本控制结构，则称为经典的结构程序设计；

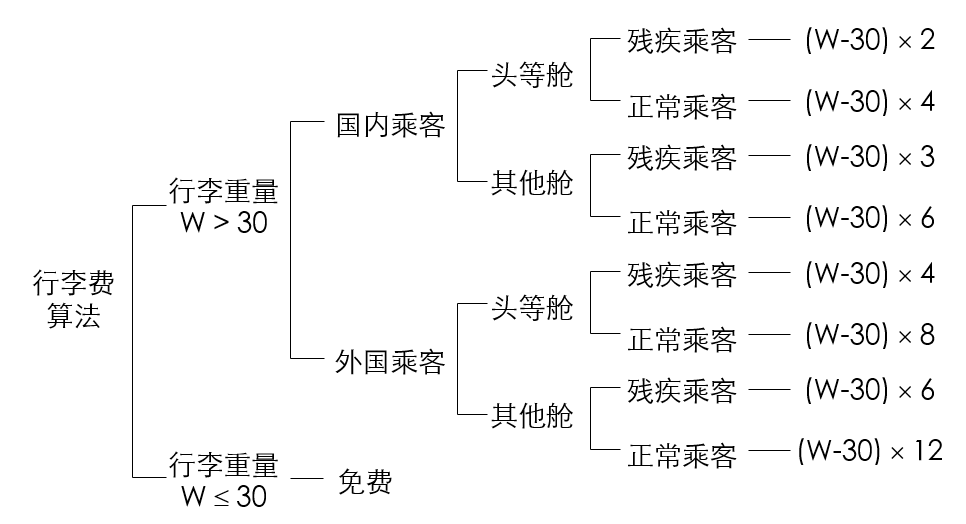
2.如果除了上述3种基本控制结构之外，还允许使用DO-CASE型多分支结构和DO-UNTIL型循环结构，则称为扩展的结构程序设计；

3.如果再允许使用LEAVE(或BREAK)结构，则称为修正的结构程序设计。

判定表



判定树



Jackson系统开发方法（JSD）（与SD方法对比）与Jackson图了解即可

面向数据结构的设计是根据问题的数据结构定义一组映射，把问题的数据结构转换为问题解的程序，最终目标是得出对程序处理过程的描述

与结构化设计SD的由DFD导出结构设计不同，Jackson系统开发方法JSD（Jackson System Development）是以数据结构(data structure)为基础设计每个模块的处理过程。

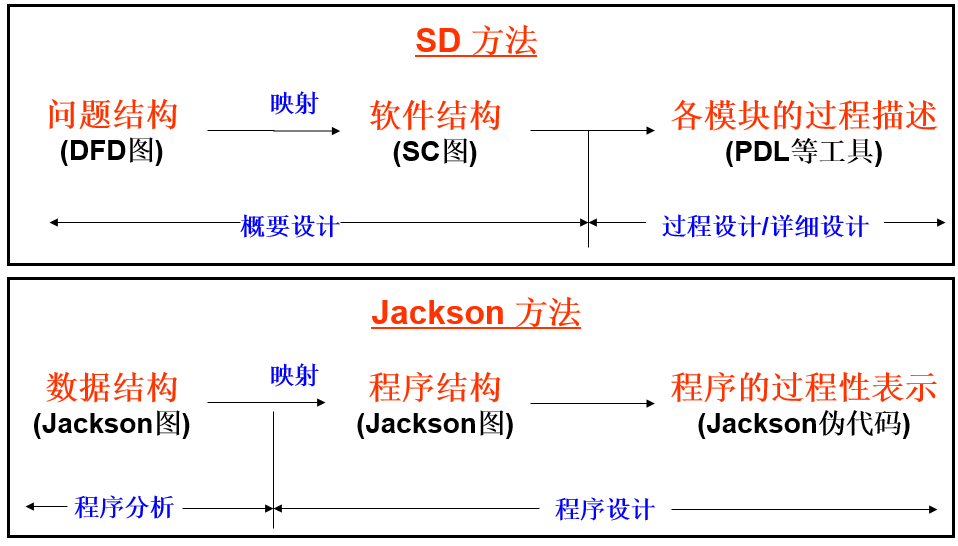
Jackson方法的基本步骤：

1.建立数据结构；

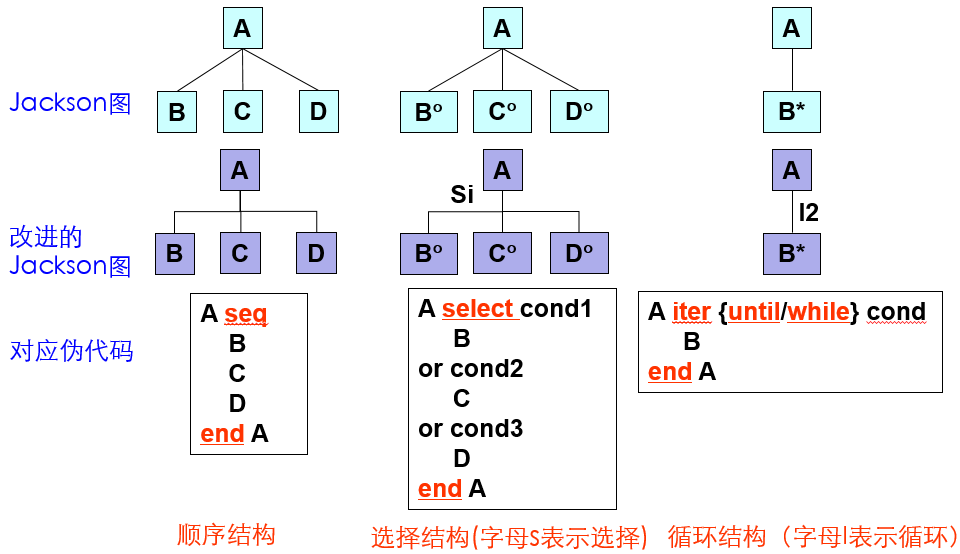
2.以数据结构为基础，对应地建立程序结构；

3.列出程序中要用到的各种基本操作，再将这些操作分配到程序结构中适当的模块。

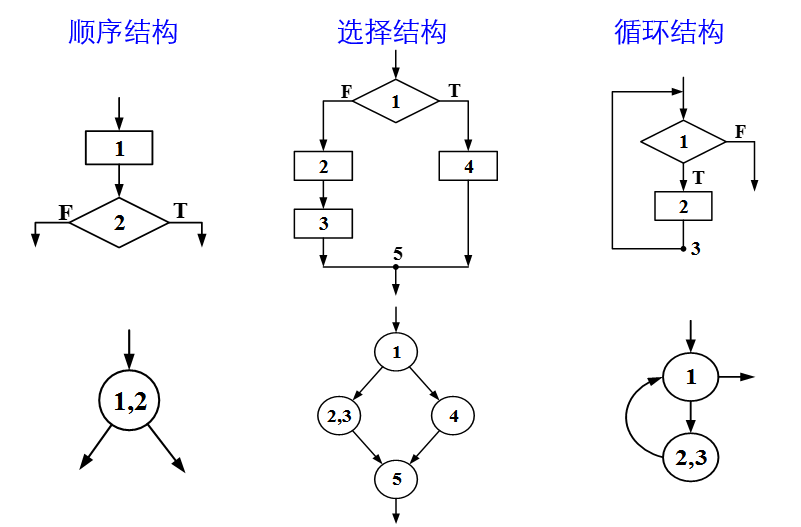
这三步分别对应结构化方法的需求分析、概要设计和详细设计。

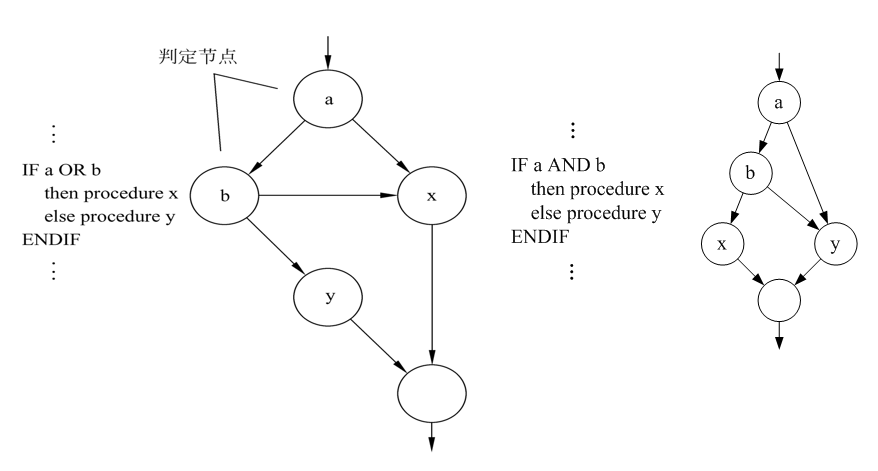


Jackson图



McCabe度量法必考（流程图转流图）





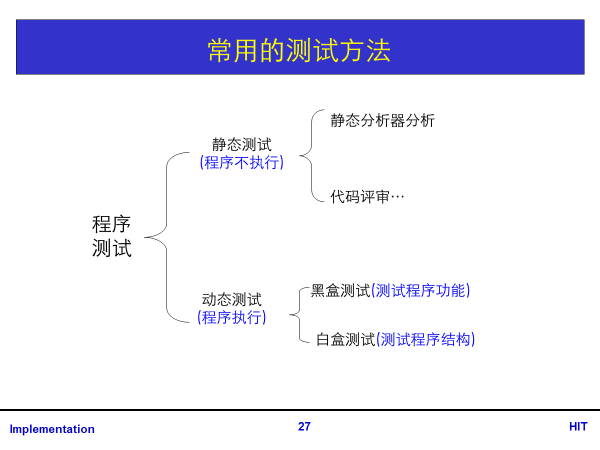
计算环形复杂度

(1)流图G的环形复杂度V(G)=平面被流图划分成的区域数（包含外部区域）。

(2)流图G的环形复杂度V(G)=E-N+2,其中E是流图中边的条数，N是结点数。

(3)流图G的环形复杂度V(G)=P+1，其中，P是流图中判定结点的数目。

第七章：



黑盒测试与白盒测试的定义

1.黑盒测试（又称功能测试）把程序看作一个黑盒子，完全不考虑程序的内部结构和处理过程。在程序**接口**进行的测试，测功能。

2.白盒测试（又称结构测试）是把程序看成装在一个透明的白盒子里，测试者完全知道程序的结构和处理算法。在程序内部的测试，测**逻辑**。

（1）模块测试=单元测试

单元测试集中检测软件设计的最小单元——模块，通过对模块的静态分析与动态测试，使其代码达到模块说明的需求。

单元测试和编码属于软件过程的同一个阶段，主要使用白盒测试技术，而且对多个模块的测试可以并行地进行。

单元测试的目的是：保证每个模块作为一个单元能正确运行，所以单元测试通常又称为模块测试。

在这个测试步骤中所发现的往往是编码和详细设计的错误。

在多模块程序中，每一模块都可能调用其它模块或者被其它模块所调用。所以在单元测试时，需要为被测试模块编制若干测试软件，给它的上级模块或下级模块作替身。代替上级模块的称为驱动模块，代替下级模块的称为桩/存根模块。

（2）集成测试=子系统测试（内部还有回归测试）

通过单元测试的模块要按照一定的策略组装为完整的程序，在该组装过程中进行的测试称为集成测试或组装测试。

集成测试是：把经过单元测试的模块放在一起形成一个子系统来测试。

模块相互间的协调和通信是这个测试过程中的主要问题，因此，这个步骤着重测试模块的接口。

集成测试的策略：

1.自顶向下集成：从主控制模块开始，沿着程序的控制层次向下移动，逐渐把各个模块结合起来。

2.自底向上测试：从软件结构最低层的模块开始组装和测试。

3.混合测试方式：对软件结构中较上层使用的自顶向下方法与对软件结构中较下层使用的自底向上方法相结合。

集成测试中，每并入一个模块，除进行新的测试项目外，还须重复进行先前已经进行过的测试，后者称为回归测试。

回归测试集包括3类测试用例：

- 检测软件全部功能的代表性测试用例。

- 针对可能受修改影响的软件功能的附加测试。

- 针对被修改过的软件成分的测试。

（3）系统测试

系统测试/系统集成测试是：把经过测试的子系统装配成一个完整的系统来测试。

这个过程中不仅应该发现设计和编码的错误，还应该验证系统确实能提供需求说明书中指定的功能，而且系统的动态特性也符合预定要求

此测试中发现的往往是**软件设计**中的错误，也可能发现**需求说明**中的错误。

（4）验收测试=确认测试

确认测试也称为验收测试，它的目标是验证软件的有效性。

软件需求规格说明书是进行确认测试的基础。

验收测试：测试内容与系统测试一致，但是它是在用户积极参与下进行的，而且可能主要使用实际数据进行测试。

验收测试的目的是验证系统确实能够满足用户的需要，在这个测试步骤中发现的往往是系统需求说明书中的错误。

1.测试所有功能要求是否满足，是否能达到每个性能要求，文档资料是否准确完整。

2.测试其他预定的要求是否满足，例如安全性、可移植性、兼容性和可维护性等。

平行运行测试

平行运行：同时运行新开发出来的系统和将被它取代的旧系统，以便比较新旧两个系统的处理结果。

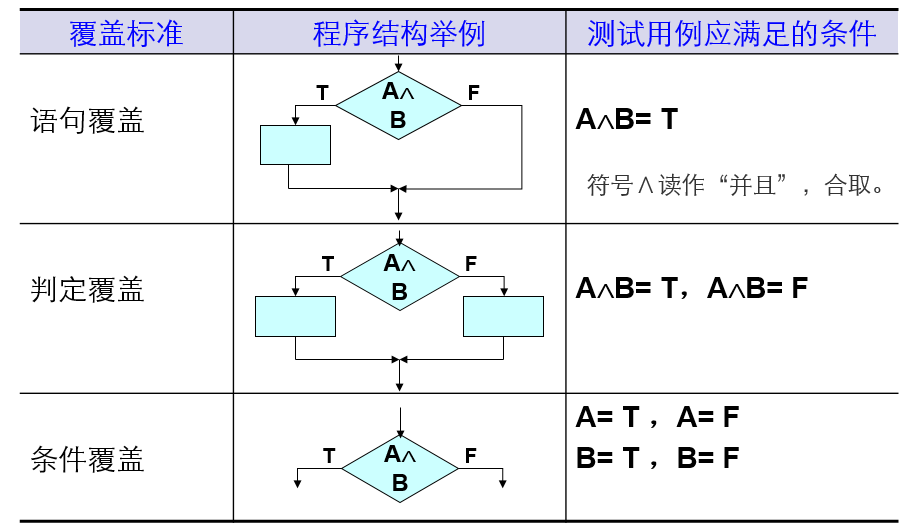
Alpha测试与Beta测试

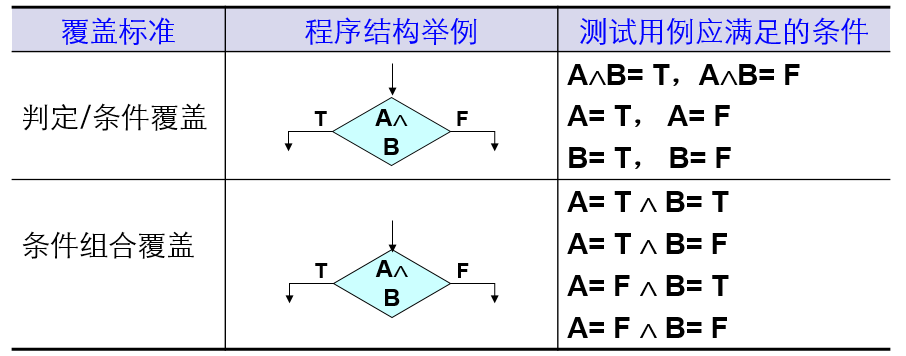
Alpha测试：是在一个受控的环境下，由用户在开发者的指导下进行测试，由开发者负责记录错误和使用中出现的问题。

Beta测试：由最终用户在自己的场所进行，开发者通常不在场，也不能控制应用的环境。由用户记录错误和使用中出现的问题，并定期地交给开发者来解决。

白盒测试的两个方法（逻辑覆盖测试和路径测试）

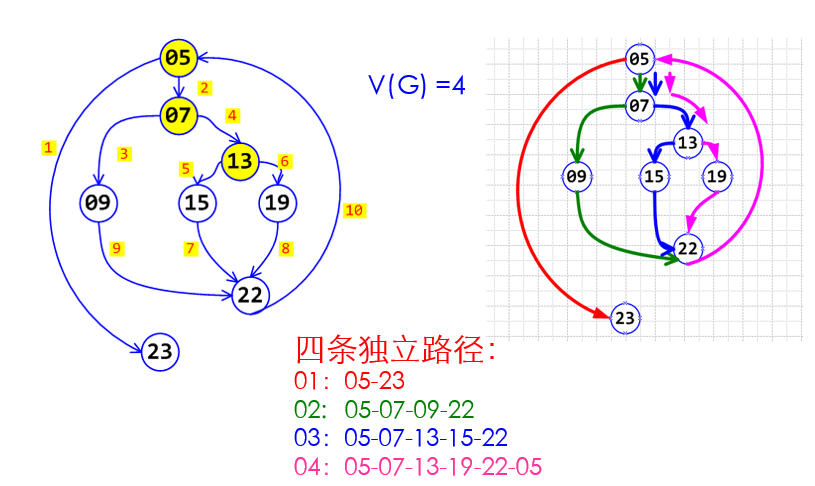
逻辑覆盖测试：





路径测试：对程序图中每一条可能的程序执行路径至少测试一次。如果程序中含有循环(在程序中表现为环)，则每个循环至少执行一次。

通过流图的环形复杂度确定独立路径数量：



黑盒测试的两个方法（等价测试和边界测试）

等价测试：把输入数据的可能值划分为若干个等价类，使每类中的任何一个测试用例，都能代表同一等价类中的其它测试用例。划分等价类不仅要考虑代表“有效”输入值的有效等价类，还要考虑代表“无效”输入值得无效等价类；每一个无效等价类至少要用一个测试用例，不然可能漏掉某一类错误，但允许若干个有效等价类合用一个测试用例，以便进一步减少测试的次数。



边界测试：实践表明，程序员在处理边界情况时，很容易发生编码错误。例如，数组容量、循环次数以及输入数据与输出数据在边界值附近程序出错概率往往较大。使用边界值分析方法设计测试方案首先应该确定边界情况，通常输入等价类和输出等价类的边界。选取的测试数据应该刚好等于、刚刚小于和刚刚大于边界值。

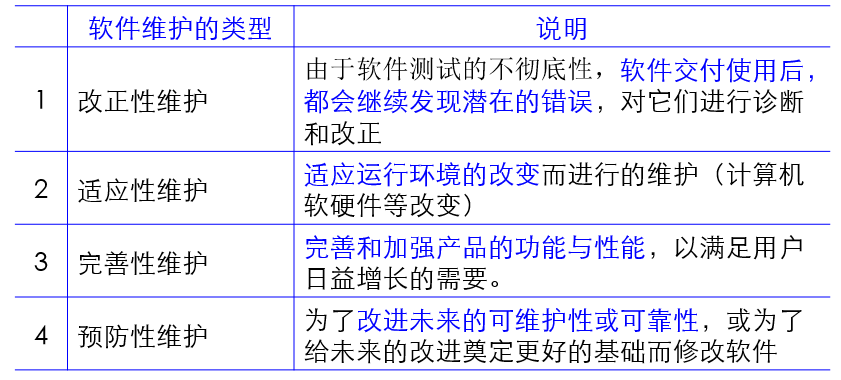


第八章

软件维护的定义

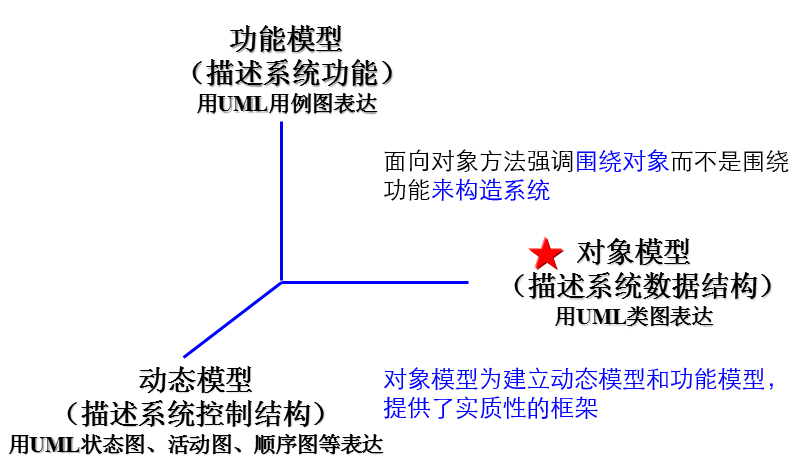
所谓软件维护，就是在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程。

软件维护的分类



第九章

面向对象方法的三种模型



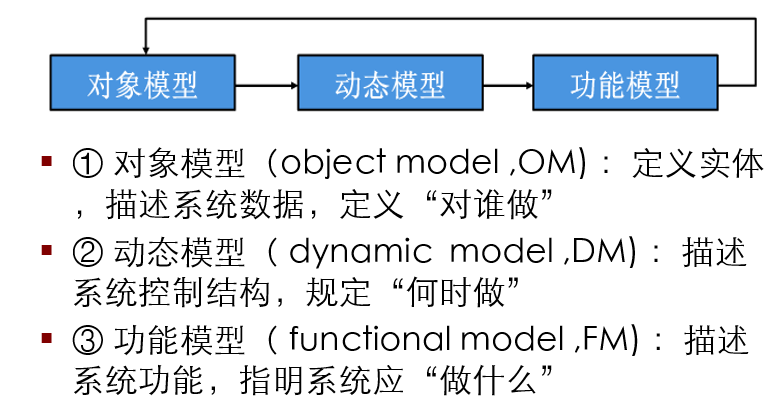
侧重点

对象模型：定义了做事情的实体,描述了动态模型、功能模型所操作的数据结构。对象模型中的操作对应于动态模型中事件和功能模型中的函数。

动态模型：描述了对象的控制结构，它明确规定了什么时候(即在何种状态下接受了什么事件触发)做;。

功能模型：指明了系统应该“做什么”，由数据流图和用例图组成，描述了对象模型中操作的含义、动态模型中动作的意义以及对象模型中约束的意义。

建模顺序

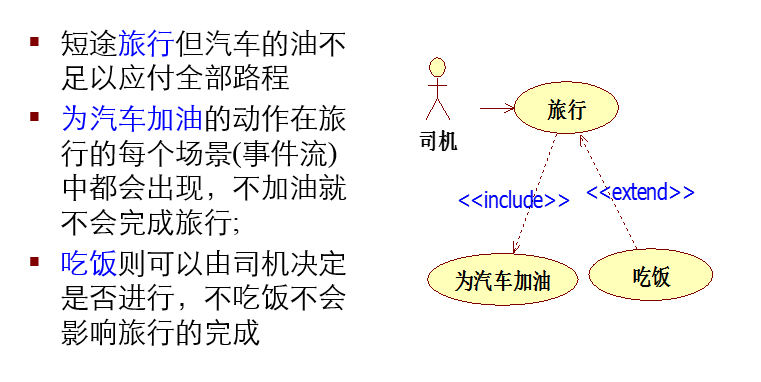


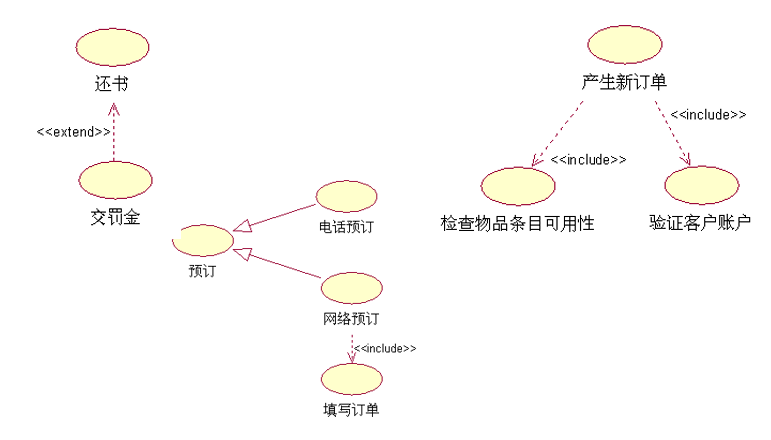
对象模型：UML类图（首选）

动态模型：UML状态图（首选）、活动图、顺序图

功能模型：UML用例图（首选）、DFD图

重点：用例图的关联关系





第十章：

对象模型五个层次：

1.主题层

2.类与对象层

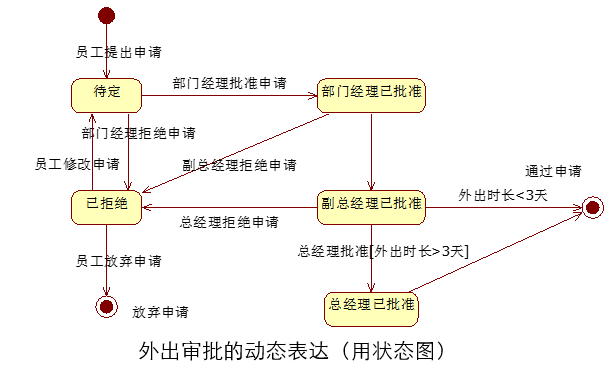
3.结构层

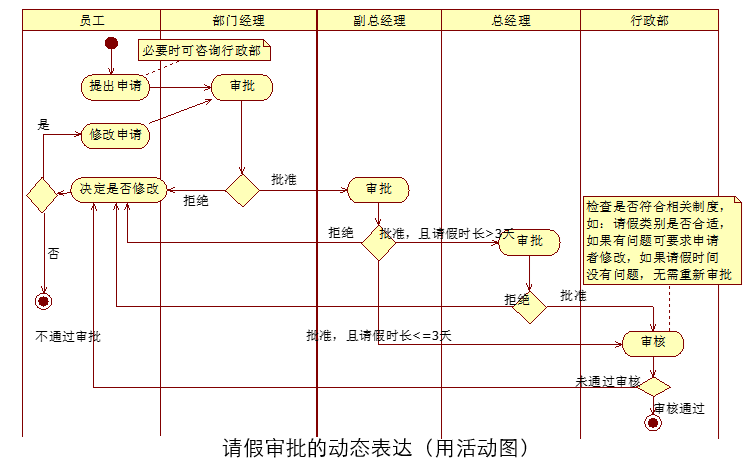
4.属性层

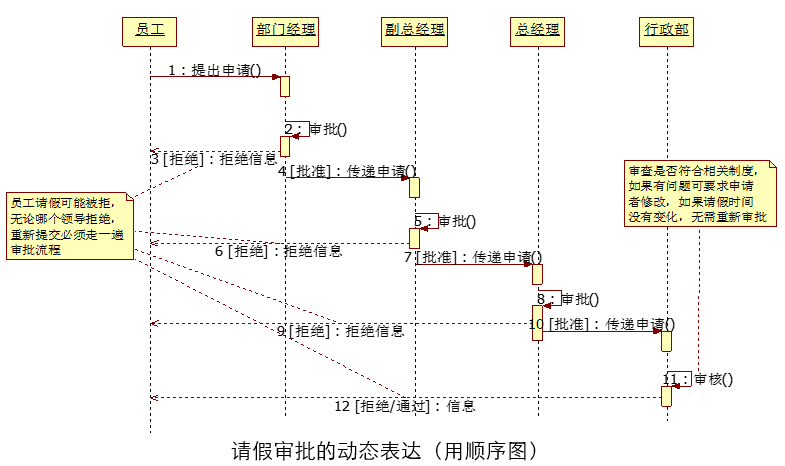
5.服务层

如果系统并非很复杂，也可以没有主题

动态模型







第十一章

九条职责分配的一般规则

信息专家模式：

某一个设计模型可能定义几百个类，几千个责任，如何将责任分配给对象？

将职责分配给具有完成该职责所需信息的那个类

创建者模式：

谁有创建某个类的对象的责任？

如果以下条件之一成立，则可以将创建类A实例的职责分配给类B。

1.B对象聚合、或者包含了A对象；

2.B对象将A对象存入内存；

3.B对象紧密地调用A对象；

4.B对象包含了创建A对象的数据；

低耦合模式：

分配职责以使(不必要的)耦合保持在较低的水平。

高内聚：

高内聚度是对一个类中的各个职责之间相关程度和集中程度的度量，包括两个意思：

1.不要给一个类分派太多的职责，在履行职责时尽量将部分职责分派给有能力完成的其它类去完成。

2.不相关的职责不要分派给同一个类。

控制器模式:

系统消息由外部用户产生，系统为了响应这些消息需要一些操作。谁负责处理输入系统的消息和相应事件？

将接受（处理）系统事件消息的责任分配给一个代表整体系统（或子系统、设备）或者代表一个用例场景的类。

多态模式:

如果一个程序使用了条件语句，则当有新的条件被加入的话，则需要修改原来的条件语句组的逻辑，程序不容易扩展。

当相关的行为基于类型而变化，使用多态操作，将行为责任分配给相关的类型。

纯虚构模式：

将一组高内聚的责任分配给一个虚构的类，该类不代表领域类概念。

间接模式：

引入一个间接类，利用该类取消所有子类的高耦合。

受保护变化模式：

如何设计对象、子系统和系统，以使这些元素中的变化或不稳定对其他元素不会产生不良影响？

确定能预测到的（类型）变化或不稳定点；分配责任以建立一个稳定的接口。

第十二章

面向对象测试和结构化测试的区别

测试的定义必须扩大包括用于OOA和OOD模型的错误发现技术

单元和集成测试策略必须有很大的改变

测试用例的设计必须考虑OO软件的独特特征

模块测试：（白盒测试）

考虑面向对象的软件时，单元的概念改变了，最小的可测试单元是封装起来的类和对象。测试面向对象软件时，不能孤立地测试单个操作，而应该把操作作为类的一部分测试。

传统的单元测试往往关注模块的算法细节和模块接口间的数据流动，而OO软件的单元测试是由封装在类中的操作和类的状态行为所驱动。

着重测试单个类和类中封装的方法主要有：随机测试、划分测试和基于故障的测试等3种。

随机测试：随机选取测试类操作序列以测试该类对象不同的生命历史

划分测试：把输入输出分类，然后设计测试用例测试每个类别；3种划分：状态，属性，功能

基于故障的测试：与传统的错误推测法类似，首先推测软件中可能有的错误，然后设计出最可能发现这些错误的测试用例。

集成测试：（黑盒测试）

在面向对象的软件中不存在层次的控制结构，传统的自顶向下或自底向上的集成策略就没有意义了。传统的增量集成方法（一次集成一个操作到类中），经常是不可能的。

面向对象软件的集成测试主要有下述两种不同的策略：

1.基于线程的测试：集成对应系统的一个输入或者事件所需的一组类，每个线程被集成并分别测试，应用回归测试以保证没有产生副作用。

2.基于使用的测试：通过测试那些几乎不使用服务器类的类（称为独立类）开始构造系统，在独立类测试完成后，再测试下一层的使用独立类的类（称为依赖类）。这个依赖层次的测试序列一直持续到构造完整个系统为止。

集成测试用例主要针对类间协作进行测试。

测试类间协作可运用基于使用的测试策略, 找出相互依赖的类，然后设计用例触发类间的各种交互序列。

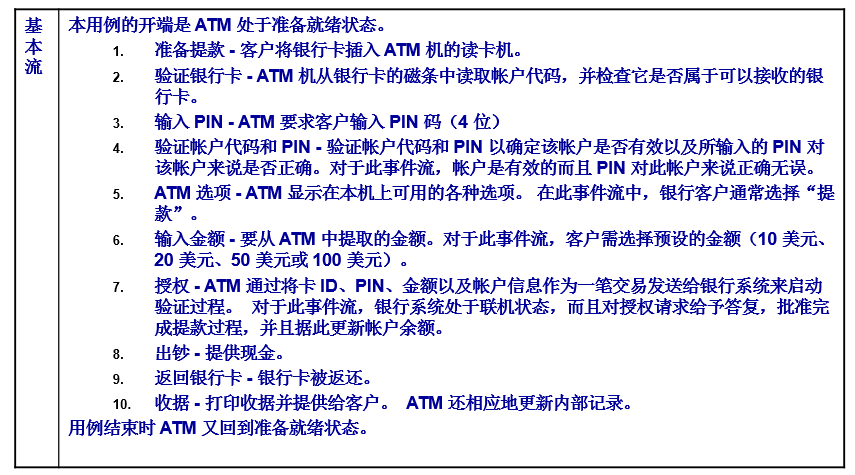
测试类协作可以使用随机测试方法和划分测试方法、基于情景/脚本的测试和行为测试来完成。

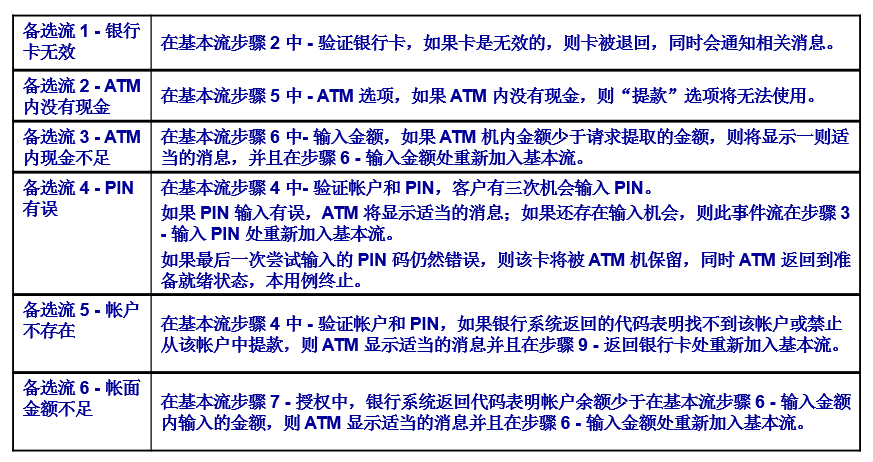
系统测试：

**基于情景（场景）**的测试是OO系统测试的主流方法。

用于功能性测试的测试用例来源于测试目标的用例，应该为每个用例场景编制测试用例。

用例场景要通过描述流经用例的路径来确定，这个流经过程要从用例开始到结束遍历其中所有基本流和备选流。











确认测试：（黑盒测试）

和传统的确认测试一样，面向对象软件的确认测试也集中检查用户可见的动作和用户可识别的输出。从动态模型和描述系统行为的脚本（即用例描述）可导出确认测试用例，以发现用户交互需求可能错误的情景。

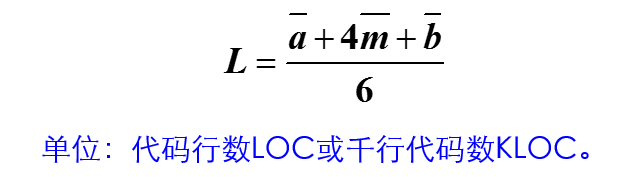
黑盒测试方法也可用于设计上述确认测试用例

第十三章

代码行技术

代码行技术：依据经验和历史数据，估计实现一个功能所需源程序行数。

估算方法：由多名有经验的软件工程师分别估计程序的最小规模(a)、最大规模(b)和最可能的规模(m)，分别算出这3种规模的平均值、和之后，再用下式计算程序规模的估计值：

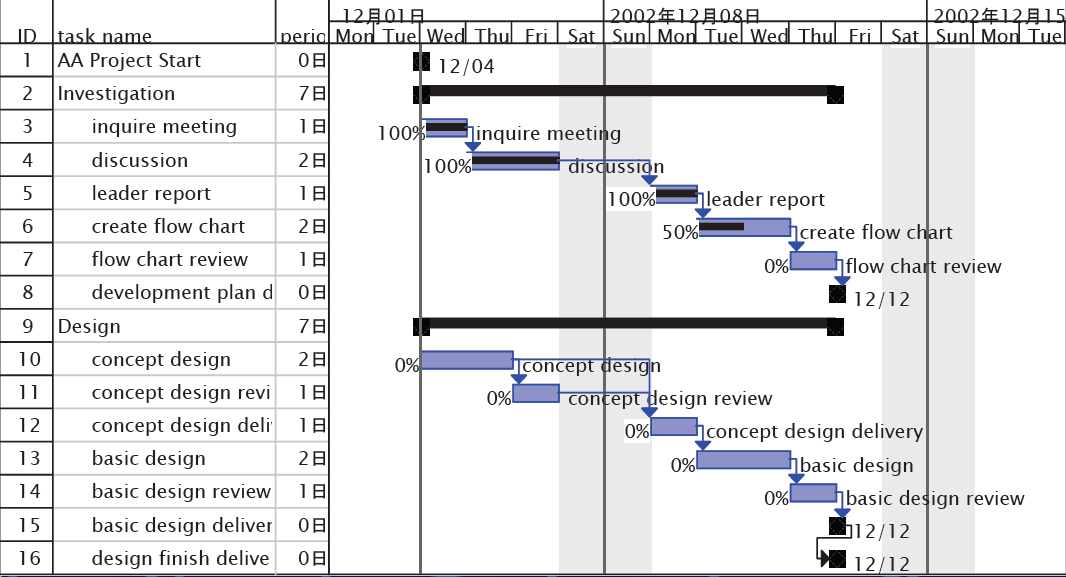


优点：所有软件开发项目都有代码，而且很容易计算代码行数；有大量参考文献和数据 。

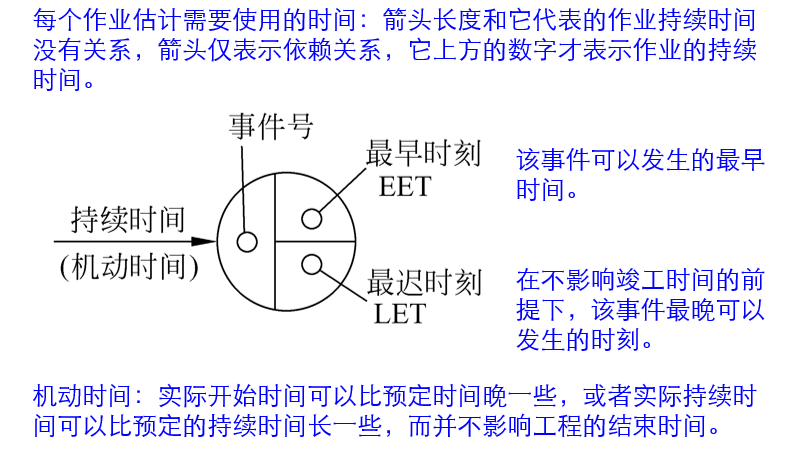
缺点：源程序仅是软件配置的一个成分，由源程序度量软件规模不太合理；用不同语言实现同一个软件所需要的代码行数并不相同；不适用于非过程性语言。

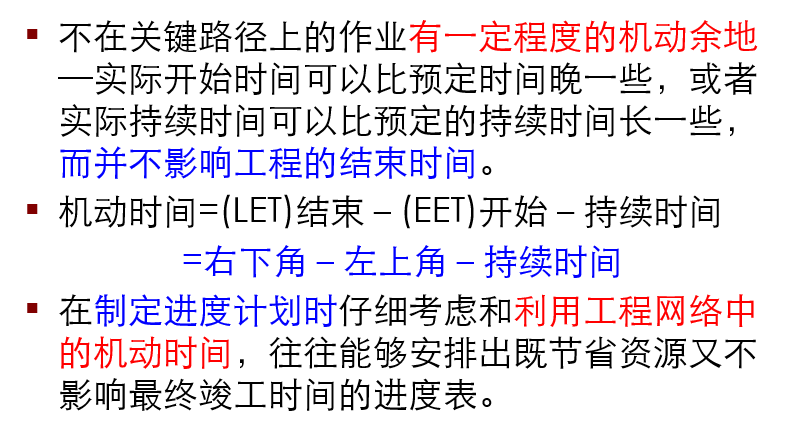
甘特图

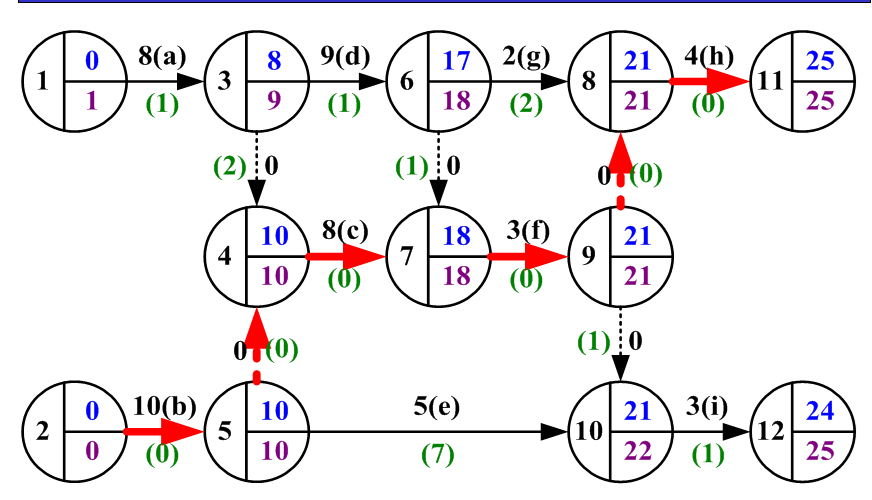
Gantt(甘特)通过日历形式列出项目活动及其相应的开始和结束日期，是反映项目进度信息的一种标准形式。



工程网络







能力成熟度模型（CMM）

初始级、可重复级、已定义级、已管理级、优化级

能力成熟度模型（capability maturity model，CMM)，是用于评价软件机构的软件过程能力成熟度的模型。

CMM的定义：CMM是对于软件组织在定义、实施、度量、控制和改善其软件过程的实践中各个发展阶段的描述。

1.初始级：处于这个最低成熟度等级的软件机构，基本上没有健全的软件工程管理制度

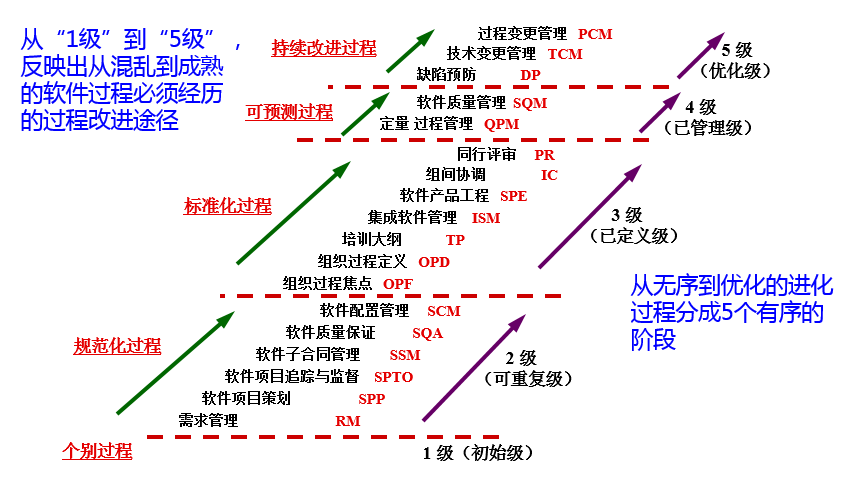
其软件过程完全取决于项目组的人员配备，所以具有不可预测性，人员变了过程也随之改变

2.可重复级：软件机构建立了基本的项目管理过程 （过程模型），可跟踪成本、进度、功能和质量。

3.已定义级：软件机构已经定义了完整的软件过程（过程模型），软件过程已经文档化和标准化。

4.已管理级：处于4级成熟度的软件机构的过程能力可以概括为，软件过程是可度量的，软件过程在可度量的范围内运行。

5.优化级：处于5级成熟度的软件机构的过程能力可以概括为，软件过程是可优化的。



押题：简答题  
简述面向对象模型和传统模型相比有哪些不同？

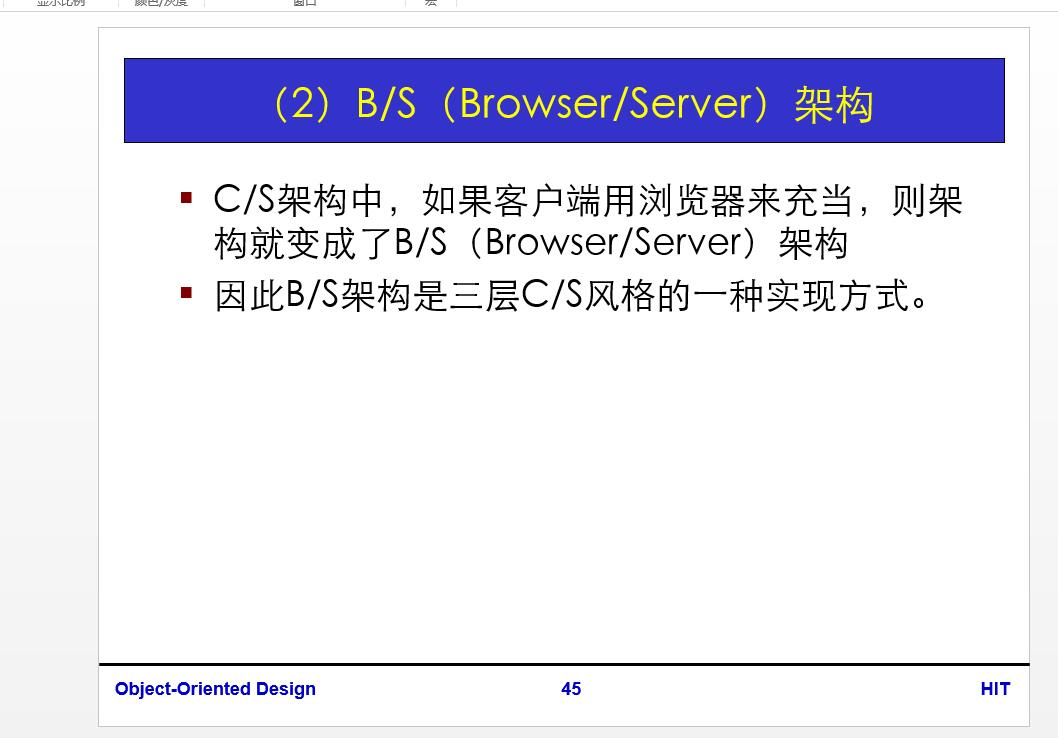
传统方法学：也称为生命周期方法学或结构化范型。它采用结构化技术来完成软件开发的各项任务，并使用适当的软件工具或软件工程环境来支持结构化技术的运用。其特点：传统方法学把软件生命周期的全过程依次划分为若干个阶段，然后顺序地完成每个阶段的任务。每个阶段的开始和结束都有严格标准，对于任何两个相邻的阶段而言，前一阶段的结束标准就是后一阶段的开始标准。

面向对象方法学：面向对象方法把数据和行为看成是同等重要的，面向对象方法学以数据为主线，把数据和处理结合构成统一体，即对象。它是一种以数据为主线，把数据和对数据的操作紧密地结合起来的方法。四个要点：把对象(object)作为融合了数据及在数据上的操作行为的统一的软件构件。把所有对象都划分成类(class)。按照父类与子类的关系，把若干个相关类组成一个层次结构的系统。对象彼此间仅能通过发送消息互相联系。基本原则：尽量模拟人类习惯的思维方式，使开发软件的方法与过程尽可能接近人类认识世界、解决问题的方法与过程，从而使描述问题的问题空间(也称为问题域)与实现解法的解空间(也称为求解域)在结构上尽可能一致。优点：降低了软件产品的复杂性，提高了软件的可理解性，简化了软件的开发和维护工作。面向对象方法特有的继承性和多态性，进一步提高了面向对象软件的可重用性。

押题：简答题

简述C/S架构与B/S架构的异同点。

以下是老师在群里突然说的：

B/S架构中，Browser里面啥都没有，也就是没有代码或者业务是在Browser里面的，我强调一下哈，大家别弄错了。  
C/S因为客户端是自己做的，当然是有逻辑和代码在里面的。  
怕大家理解错了，这一页ppt只留这些内容，诸位关注一下：  


软件架构

软件架构（Software Architecture , SA）的关注点：

1.如何将复杂的软件系统划分为模块；

2.如何规范模块的构成和性能；

3.以及如何将这些模块组织为完整的系统；

软件架构可分为逻辑架构和物理架构，逻辑架构是软件的宏观组织结构。之所以称其为逻辑架构，是因为并未决定如何在不同的操作系统进程或网络中物理的计算机上对这些元素进行部署。物理架构一般是指软件在物理系统上的部署。通常采用拓扑结构进行表示。

C/S架构

客户机/服务器：一个应用系统被分为两个逻辑上分离的部分，每一部分充当不同的角色、完成不同的功能，共同完成统一的任务。

客户机(前端，front-end)：业务逻辑、与服务器通讯的接口；

服务器(后端：back-end)：与客户机通讯的接口、业务逻辑、数据管理。

客户机为完成特定的工作向服务器发出请求,服务器处理客户机的请求并返回结果。

B/S架构

C/S架构中，如果客户端用浏览器来充当，则架构就变成了B/S（Browser/Server）架构，因此B/S架构是三层C/S风格的一种实现方式。

C/S+B/S混合

为了克服C/S与B/S各自的缺点，发挥各自的优点，在实际应用中，通常将二者结合起来.

企业内部用户通过局域网直接访问数据库服务器

C/S结构；

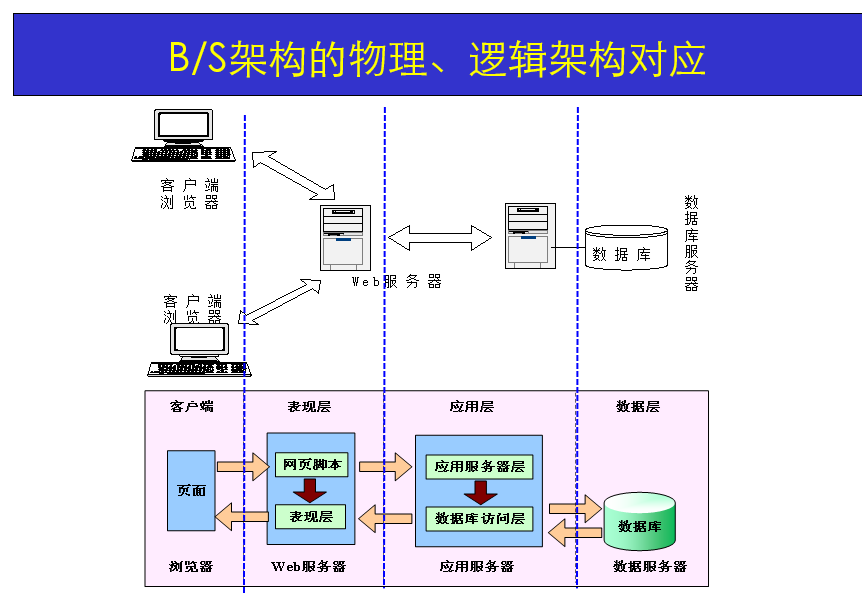
交互性增强；

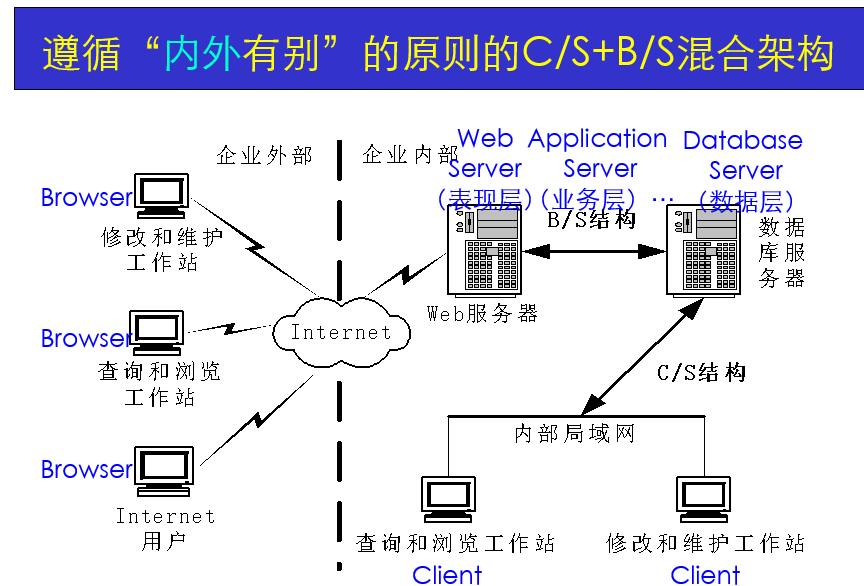
数据查询与修改的响应速度高；

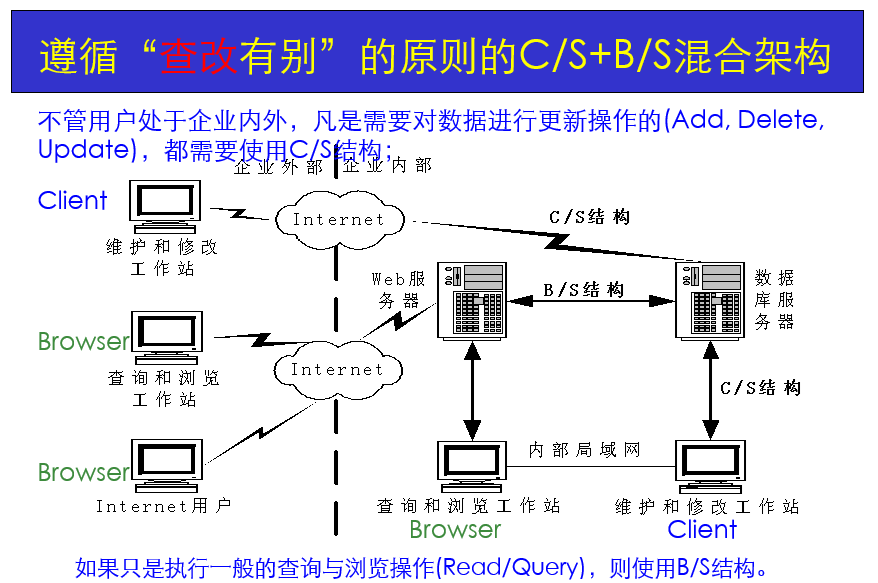
企业外部用户通过Internet访问Web服务器/应用服务器

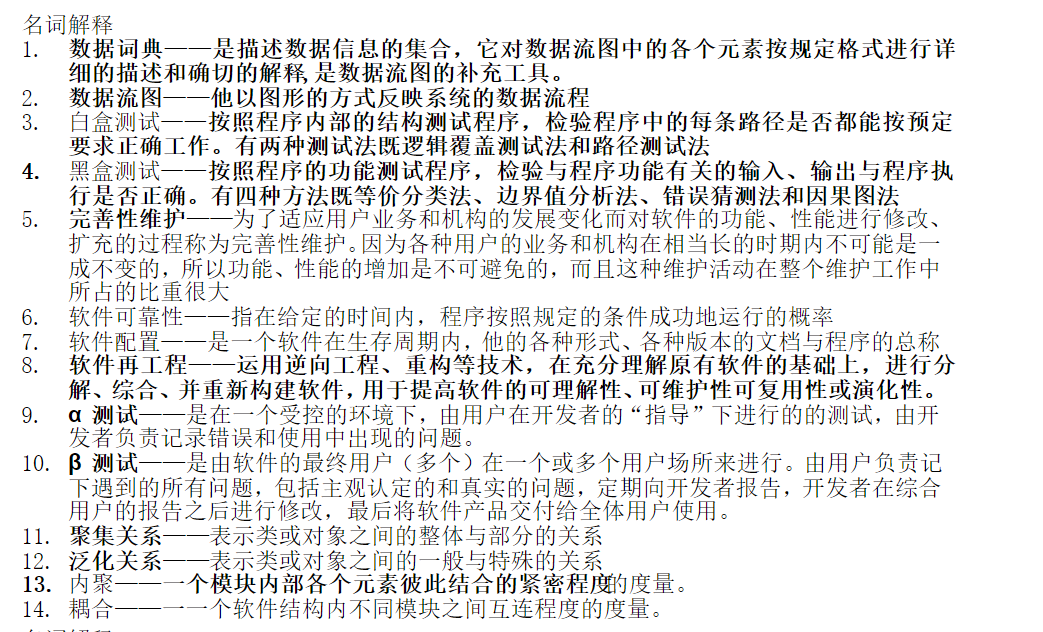
B/S结构；

用户不直接访问数据，数据安全；









图：<https://wenku.baidu.com/view/d58de81b011ca300a7c39036.html?tdsourcetag=s_pctim_aiomsg&qq-pf-to=pcqq.c2c>