第一章：软件工程学概述

软件危机：软件危机是指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。这些问题绝不仅仅是不能正常运行的软件才具有，实际上，几乎所有软件都不同程度地存在这些问题。软件危机包含下属两方面的问题；如何开发软件，以满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件。

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”软件系统不满意的现象经常发生。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所占的比例逐年上升。7、软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势。

产生软件危机的原因：1）软件本身特点造成；2）软件开发与维护的方法不正确。主要表现：（a）忽视软件需求分析；（b）认为软件开发就是写程序并使之运行；（c）轻视软件维护；

消除软件危机的途径：1）推广使用在实践中总结出来的开发软件的成功技术和方法，并研究探索更有效的技术和方法；2）开发和使用更好的软件工具；3）良好的组织管理措施。

软件工程：是指导计算机软件开发和维护的一门工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以经济地开发出高质量的软件并有效地维护它。

软件工程本质特性：1、软件工程关注于大型程序的构造。2、软件工程的中心课题是控制复杂性。3、软件经常变化。4、开发软件的效率非常重要。5、和谐地合作市开发软件的关键。6、软件必须有效地支持它的用户。7、在软件工程领域中通常由具有一种文化背景的人替具有另一种文化背景的人创造产品。

软件工程基本原理：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 3 要素：方法、工具和过程。其中，方法是完成软件工程的各项任务的技术方法，回答“怎样做”的问题；工具是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；过程是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。

软件生命周期：由软件定义（A.问题定义 B.可行性研究 C.需求分析）、软件开发（D.总体设计 E.详细设计 F.编码和单元测试 G.综合测试）和运行维护三个时期组成。指软件从提出到最终被淘汰的这个存在期。

软件生命周期每个阶段的基本任务：1、问题定义。2、可行性研究。3、需求分析。4、总体设计。5、详细设计。6、编码和单元测试。7、综合测试。8、软件维护。

软件过程：为了获得高质量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。

软件过程 3 要素：1、过程定义：将过程所包含的活动及程序文档化。2、过程学习：应将关于过程的知识传授给需要执行过程的每个人。3、过程执行

瀑布模型特点：1、阶段间具有顺序性和依赖性。2、推迟实现的观点。3、质量保证的观点。优点：采用规范的方法；严格规定每个阶段提交的文档；要求每个阶段交出的产品必须经过验证。

快速原型模型：是快速建立起来的可以在计算机上运行的程序，它所能完成的功能往往是最终产品能完成的功能的一个子集。优点：不带反馈环，基本上是线性顺序进行。

快速原型模型特点：优点：处理模糊需求 、用户参与、快速　　缺点：快速（弱功能）、资源规划和管理较为困难、对开发环境要求高　　适用范围：已有产品（原型）、简单而熟悉的领域、有快速原型开发工具、进行产品移植或升级

增量模型：也称为渐增模型，使用增量模型开发软件时，把软件产品作为一系列的增量构建来设计、编码、集成和测试。优点是能在较短时间内向用户提交可完成都分工作的产品，逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

增量模型特点：优点：任务或功能模块驱动，可以分阶段提交产品

缺点：条件比较苛刻

适用范围：1、在整个项目开发过程中，需求都可能发生变化，客户接受分阶段交付 2、分析设计人员对应用领域不熟悉，难以一步到位 3、中等或高风险项目 4、用户可参与到整个软件开发过程中 5、使用面向对象语言或第四代语言 6、软件公司自己有较好的类库、构件库

第二章：可行性研究

1.**可行性研究的目标**是用最小的代价在尽可能短的时间内确定问题是否可解，或者确定问题是否值得去解。

2.**可行性研究包括四个方面的内容：**

（1）经济可行性 ：进行成本／效益分析。从经济角度判断系统开发是否“合算”。

（2）技术可行性 ：进行技术风险评价。从开发者的技术实力、以往工作基础、问题的复杂性等出发，判断系统开发在时间、费用等限制条件下成功的可能性。

（3）操作可行性 ：判断系统的操作方式在该用户组织内部是否可行。

（4）法律可行性 ：确定系统开发可能导致的任何侵权、妨碍和责任。

3．**可行性研究的任务？**

(1) 进一步分析和澄清问题的定义，在澄清问题的基础上，导出系统的逻辑模型；

(2) 从系统逻辑模型中，选择问题的若干种主要解法，研究每一种解法的可行性，为以后的行动提出建议；

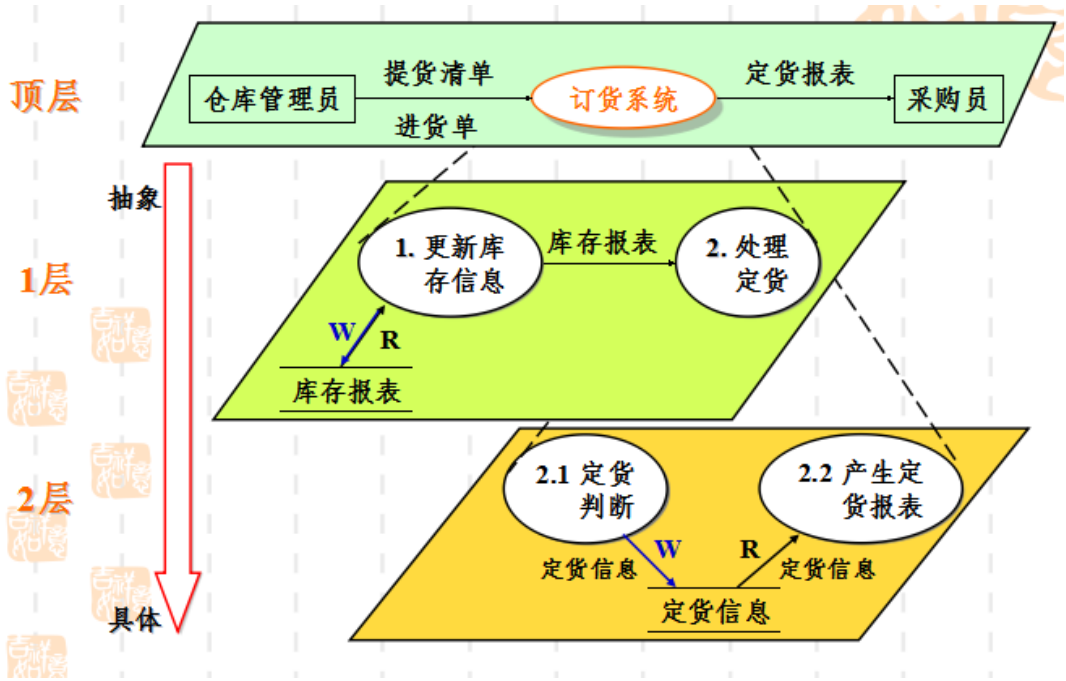
(3) 如果问题没有可行的解，建议停止系统开发；如果问题有可行的解，应该推荐一个较好的解决方案，并为工程制定一个初步的计划。

2.4 数据流图

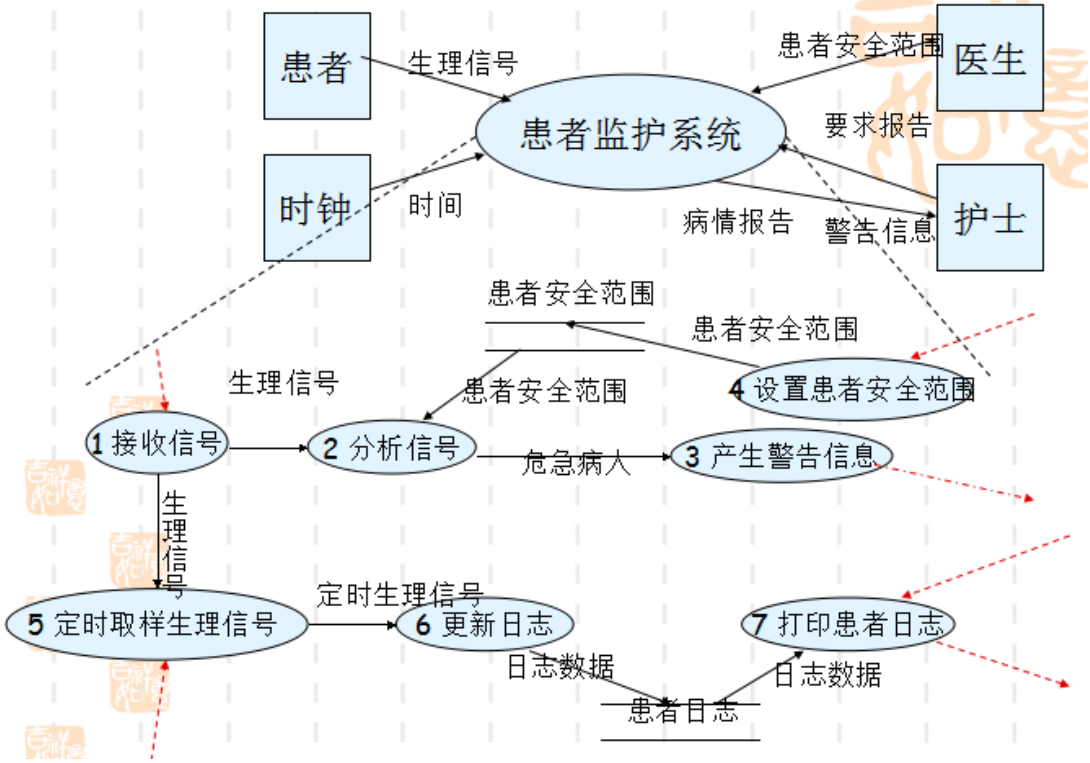
1.定义：数据流图(DFD)是一种图形化技术，它描绘数据在软件中流动和被处理的逻辑过程。非常容易理解，因此是分析员与用户之间极好的通信工具。

2.数据流图的**作用**：数据流图的用途是作为分析和设计的工具，也作为交流信息的工具。描绘系统所完成的功能而不是系统的物理实现方案。

例子：假设一家工厂的采购部每天需要一张定货报表，表中列出所有需要定货的零件。(对于每个需要定货的零件应该列出下述数据：零件编号，零件名称，定货数量，目前价格，主要供应者等)。通过放在仓库中的 CRT 终端把事务传送到定货系统。当某种零件的库存数量少于库存量临界值时就应该定货。



患者监护系统：



2.5 数据字典：数据字典是关于数据的信息的集合，也就是对数据流图中包含的所有元素的定义的集合。由数据流、数据流分量（即数据元素）、数据存储、处理四类元素组成。

1. 需求分析的任务：

1. 确定对系统的综合要求：

（1）功能需求:这方面的需求指定系统必须提供的服务。

(2)性能需求：如：相应时间（速度）、主存容量、磁盘容量、安全性、等。

（3）可靠性和可用性需求可靠性需求定量的指定系统的可靠性。

（4） 出错处理需求：系统发现错误时采取的行动，主要在系统关键部分设置。
（5） 接口需求：用户接口、硬件接口、软件接口、通信接口、等。
（6）约束：精度、工具和语言、设计约束、硬件约束、标准，等。
（7）逆向需求：说明软件系统不应该吃什么。
（8）将来可能提出的要求 ：应该明确地列出虽然不属于当前系统开发的范畴，但是根据分析将来可能会提出来的要求。

2. 分析系统的数据要求：通过建立数据模型来分析，如数据字典、层次方框图、Warnier图，并将数据结构规范化。
3. 导出系统的逻辑模型：包括完善的数据流图、实体－联系图、状态转换图、数据字典、主要的处理算法（IPO 图）等。
4. 修正系统开发计划：修订前期制定的开发进度计划、等。

3.3 需求分析的分析建模

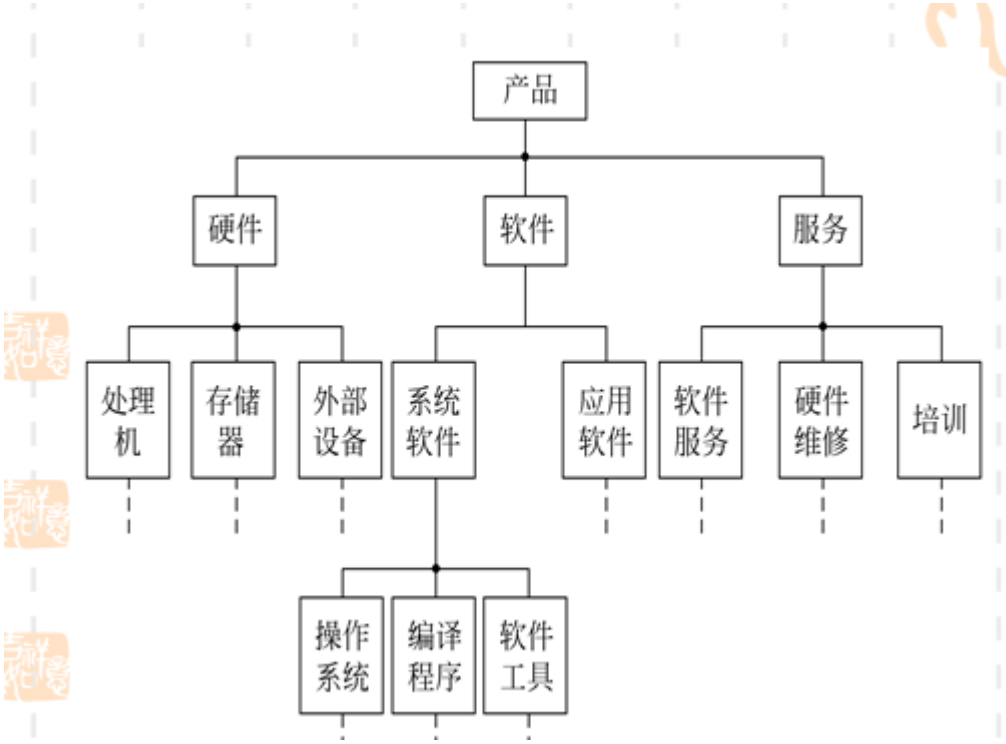
（1）数据模型：ERD(实体联系图)
（2）功能模型：DFD(数据流图)
（3）行为模型：STD(状态转换图)

3.6 状态转换图

1．**状态**：状态是任何可以被观察到的系统行为模式，一个状态代表系统的一种行为模式。
2．**事件**：事件是某个特定时刻发生的事情，它是引起系统做动作或状态转换的控制信息。
3．**符号**：在状态图中，初态用实心圆表示，终态用一对同心圆表示，中间状态用矩形。

3.7 层次方框图

层次方框图用树形结构的一系列多层次的矩形框描绘数据的层次结构。
例：

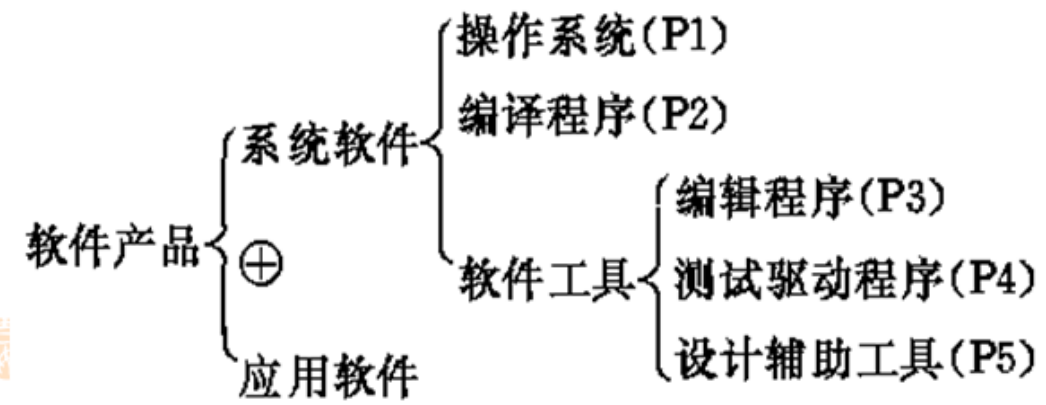


3.8 Warnier 图

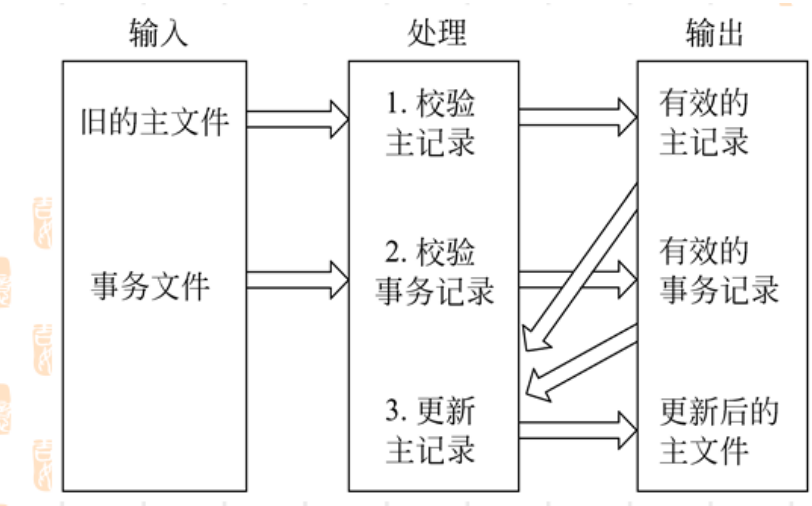
Warnier 图也用树形结构描绘信息，但是这种图形工具比层次方框图提供了更丰富的描绘手段。用 Warnier 图可以表明信息的逻辑组织，也就是说，它可以指出一类信息或一个信

息元素是重复出现的，也可以表示特定信息在某一类信息中是有条件地出现的。

例：

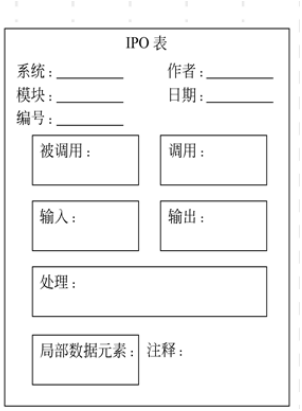


3.7 IPO 图



1. **IPO 图：**IPO 图是输入、处理、输出图的简称，它能够方便地描绘输入数据、对数据的处理和输出数据之间的关系。在需求分析阶段可以使用 IPO 图简略的描述系统的主要算法（即数据流图中各个处理的基本算法）。

2. **改进 IPO 图：**



3. IP0 图基本形式：在左边的框中列出有关的数据，中间列出主要的处理，右边列出结果。

第5章 总体设计

1、**总体设计任务**：

总体设计的基本目的就是回答“概括地说，系统应该如何实现?”总体设计又称为概要设计或初步设计。划分出组成系统的物理元素——程序、文件、数据库、人工过程和文档等等。每个物理元素仍然处于黑盒子级，这些黑盒子里的具体内容将在以后仔细设计。

2、**总体设计基本原理**：1、模块化 2、抽象 3、逐步求精 4、信息隐蔽和局部化 5、模块独立

2、**模块**是又称构件，是能够单独命名并独立地完成一定功能的程序语句的集合。**模块化**就是把程序划分成独立命名且可独立访问的模块，每个模块完成一个子功能，把这些模块集成起来构成一个整体，可以完成指定的功能满足用户的需求。

3、模块独立程度可以有两个定性标准度量：内聚和耦合。

4、耦合是对一个软件结构内不同模块之间互连程度的度量。耦合强弱取决于模块间接口的复杂程度，进入或访问一个模块的店，以及通过接口的数据。 **数据耦合：** 两个模块彼此间通过参数交换信息，而且交换的信息仅仅是数据，数据耦合是最低程度的耦合。**控制耦合：**传递的信息中有控制信息，控制耦合是中等程度的耦合。 **公用耦合：**两个或多个模块通过一个公共区相互作用时的耦合。公共区可以是：全程数据区、共享通信区、内存公共覆盖区、任何介质上的文件、物理设备等。

5、最高程度的耦合式**内容耦合**，如果出现下列情况就是内容耦合：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而转到另一个模块的内部、两个模块有一部分程序代码重叠、一个模块有多个入口。

6、当两个或多个模块通过一个公共数据环境相互作用时，他们之间的耦合称为**公共环境耦合。**

7、耦合是影响软件复杂程度的一个重要因素。应该采取下述**设计原则**：尽量使用数据耦合，少用控制耦合和特征耦合，限制公共环境耦合的范围，完全不用内容耦合。

8、**内聚：**一个模块内部各个元素彼此结合的紧密程度。 它是衡量一个模块内部组成部分间整体统一性的度量。

9、**低内聚**：如果一个模块完成一组任务，这些任务彼此间即使有关系，关系也是松散的，叫做**偶然内聚**。如果一个模块完成的任务在逻辑上属于相同或类似的一类，则叫做**逻辑内聚**。如果一个模块包含的任务必须在同一段时间内执行，就叫**时间内聚**。

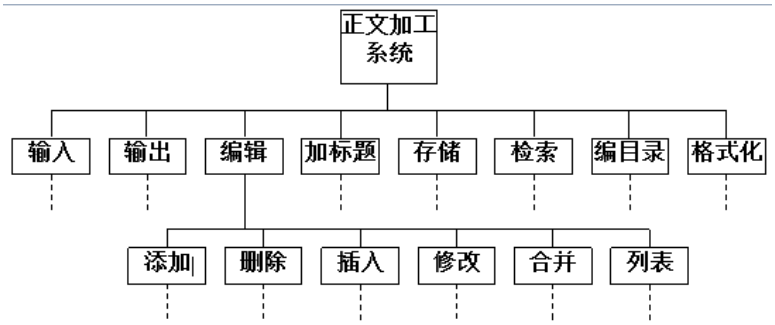
10、**中内聚**：如果一个模块内的处理元素师相关的，而且必须以特定的次序执行，叫做**过程内聚**。如果模块中所有元素都使用同一个输入数据或产生同一个输出数据，则叫做**通信内聚**。

11、**高内聚**：如果一个模块内的处理元素和同一个功能密切相关，而且这些处理必须顺序执行，则称为**顺序内聚**。如果模块内所有处理元素属于一个整体，完成一个单一的功能，则称为**功能内聚**。

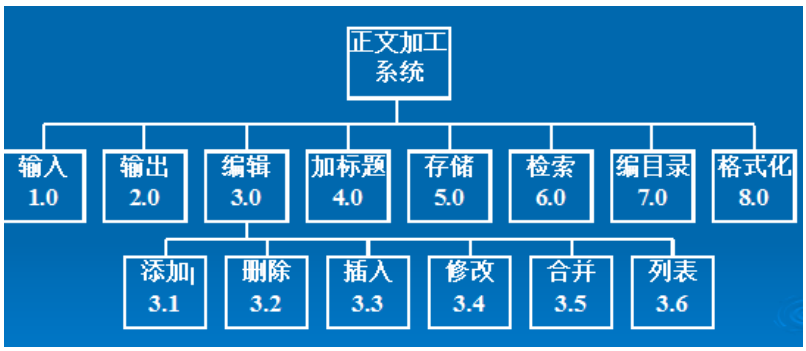
12、软件设计中应该：力求做到高内聚，尽量少用中内聚，不用低内聚。

13、描绘软件结构的图形工具：层次图、HIPO 图、结构图。

14、**层次图**用来描绘软件的层次结构，很适于在自顶向下设计软件的过程中使用。



15、**HIPO 图**是：“层次图＋输入/处理/输出图”



16、**面向数据流设计（Data Flow-Oriented Design，DFOD）**是与数据流分析（DFA）对应的结构化软件设计技术。面向数据流的设计将得到以数据流图为基础的软件模块结构图。

17、**信息流**有两种类型：**a 变换流**：具有较明确的输入、变换（或称主加工）和输出界面的数据流图称为变换型数据流图。 **b 事务流**： 事务型数据流图中存在一个事务中心（也就是数据处理、加工中心），它将输入分离成若干个发散的数据流，形成许多活动路径，并根据输入值选择其中一条路径。

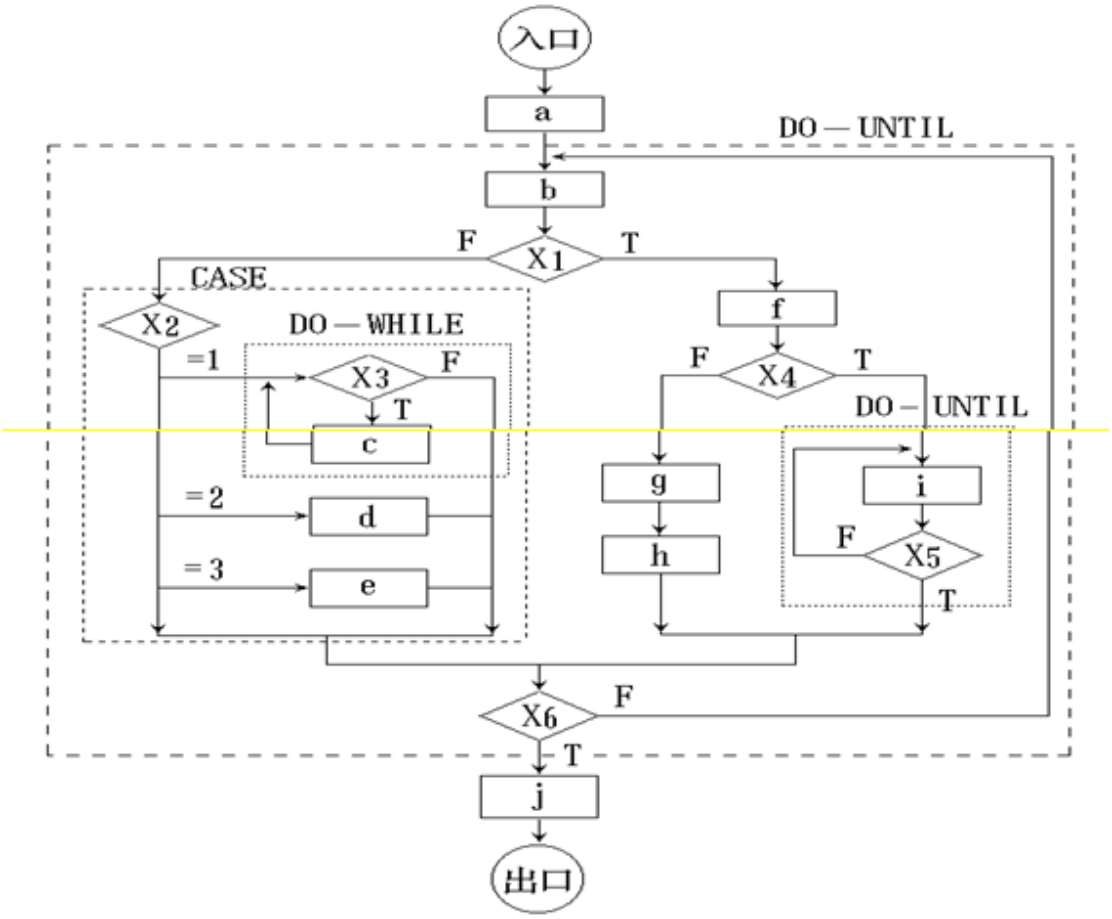
18、**变换分析**是一系列设计步骤的总称，步骤有：a、复查基本系统模型 b、复查并精化数据流图 c、确定数据流图具有变换特性还是事务特性 d、确定输入流和输出流的边界，从而孤立出变换中心 e、完成“第一级分解” f、完成“第二级分解” g、使用设计度量和启发式规则对第一次分割得到的软件结构进一步精化

第6章 详细设计

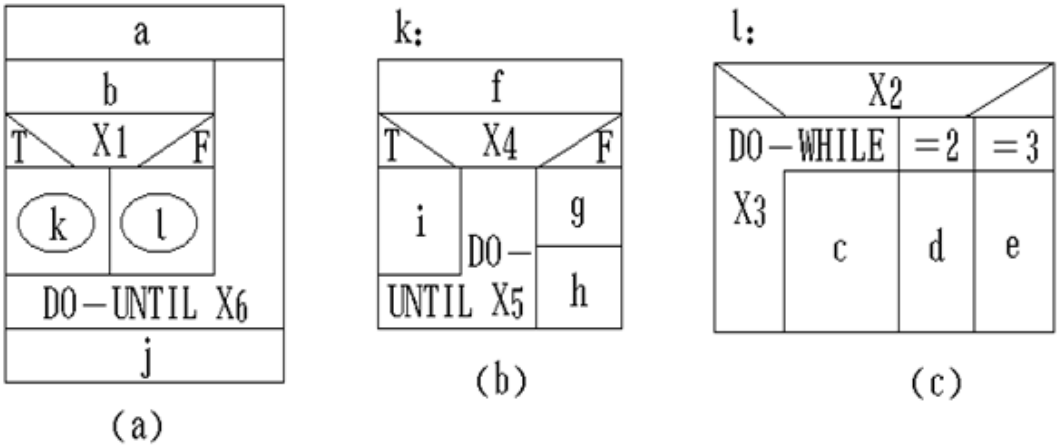
1、用 3 种基本的控制结构就能实现任何单入口单出口的程序：顺序，选择，循环

2、**结构程序设计：**是一种尽可能少用 GOTO 语句的程序设计技术，它采用自顶向下、逐步细化的设计方法和单入口单出口的控制技术。

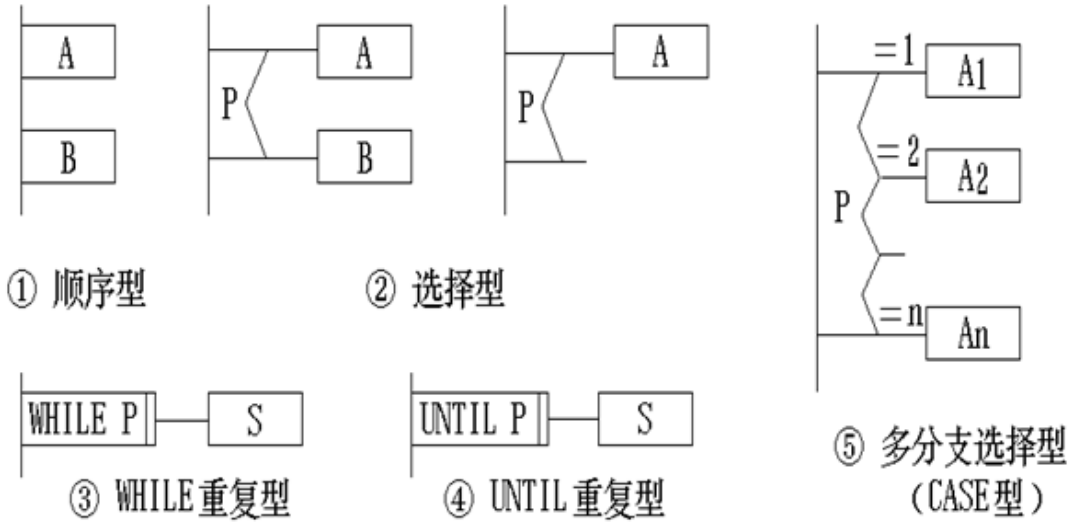
3、**程序流程图**又称程序框图，是一种描述程序的控制结构流程和指令执行情况的有向图。**优点**是对控制流程的描绘很直观，**缺点**如下：a、程序流程图本质上不是逐步求精的好工具，它有事程序员过早地考虑程序的控制流程，而不去考虑程序的全局结构。b、程序流程图中用箭头代表控制流，因此程序员不受任何约束，可以完全不顾结构程序设计的精神，随意转移控制。c、程序流程图不易表示数据结构



4、**盒图**：特点： 控制结构作用域明确、不能任意转移控制、 SP 方式思考和解决问题的习惯、易表示嵌套关系、模块的层次结构



5、**PAD 图**：即是问题分析图。它用二维树形结构的图来表示程序的控制流，将这种图翻译成程序代码比较容易。**优点**：a、使用表示结构化控制结构的 PAD 符号所涉及出来的程序必然是结构化程序 b、PAD 图所描绘的程序结构十分清晰 c、用 PAD 图表现程序逻辑，易读、易懂、易记。 d、容易将 PAD 图转换成高级语言源程序，可用软件工具完成，利于提高软件可靠性和软件生产率 e、既可用于表示程序逻辑，也可用于描绘数据结构。f、PAD 图的符号支持自顶向下、逐步求精的方法的使用。



6、**过程设计语言**（PDL）也称为伪码。它是用正文形式表示数据和处理过程的设计工具。有严格的关键字外部语法，用于定义控制结构和数据结构。

7、**PDL 特点**：a、关键字的固定语法，它提供了结构化控制结构、数据说明和模块化的特点。 b、自然语言的自由语法，它描述处理特点。 c、数据说明的手段。应该既包括简单的数据结构，又包括复杂的数据结构。 d、模块定义和调用的技术，应该提供各种借口描述模式。 **优点**：a、可以作为注释直接插在源程序中间。 b、可以使用普通的正文编辑程序或文字处理系统。很方便地完成 PDL 的书写和编辑工作。 c、已经有自动处理 PDL 的程序存在，而且可以自动由 PDL 生产程序代码。 **缺点**：不如图形工具形象直观，描述复杂的条件组合与动作间的对应关系时，不如判定表清晰简单。

8、**Jackson 方法的目标**是：得出对程序处理过程的详细描述。

9、**Jackson 结构程序设计方法由五个步骤**组成：1）分析并确定输入数据和输出数据的逻辑结构，并用 Jackson 图描绘这些数据结构；2）找出输入数据结构和输出数据结构中有对应关系的数据单元；3）用三条规则从描绘数据结构的 Jackson 图导出描绘程序结构的 Jackson 图：A．为每对有对应关系的数据单元，按照它们在数据结构图中的层次在程序结构图的相应层次画一个处理框；B．根据输入数据结构中剩余的每个数据单元所处的层次，在程序结构图的相应层次分别为它们画上对应的处理框；C．根据输出数据结构中剩余的每个数据单元所处的层次，在程序结构图的相应层次分别为它们画上对应的处理框； 4）列出所有操作和条件（包括分支条件和循环结束条件），并且把它们分配到程序结构图的适当位置；5）用伪码表示程序。f

10、**MaCabe 计算环形复杂度的方法**

(1) 流图中的区域数等于环形复杂度。

(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　实现

1、软件测试：为发现程序中的错误而执行程序的过程。

测试的目的：暴露程序中的错误

软件测试的准则（尽早和不断的测试、彻底测试的不可能、软件测试是有风险的行为、并非所有的软件错误都能恢复、反向思维逻辑、由小到大的测试范围、避免检查自己的代码、追溯至用户需求）

测试方法（黑盒测试和白盒测试）

测试步骤（模块测试、子系统测试、系统测试、验收测试、平行运行）

3 单元测试也称模块测试（module testing ）测试的方法一般采用白盒法。单元测试需要从程序的内部结构出发设计测试用例，以路径覆盖为最佳准则，且系统内多个模块可以并行地进行测试。模块并不是一个独立的程序，在考虑测试模块时，同时要考虑它和外界的联系，用一些辅助模块去模拟与被测模块相联系的其它模块。

单元测试的内容：模块接口、局部数据结构、重要的执行路径、错误处理、边界测试

4 集成测试，也叫组装测试或联合测试。在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。程序在某些局部反映不出来的问题，在全局上很可能暴露出来，影响功能的实现。

集成测试常用方法：非渐增式测试和渐增式测试。

自顶向下集成优缺点：优点：1、尽早测试主要的控制、关键的抉择（上层）2、尽早实现软件的完整功能并验证（深度优先）3、无需 driver

缺点：1、需编写 stub　2、底层关键模块测试晚 3、软件结构中没有重要的数据自下往上流 4、并行测试困难

自底向上集成优缺点：优点：1、无需 stub　2、可尽早并行测试 3、可尽早发现底层关键模块的错误 4、易建立测试条件，易判定测试结果

缺点：1、需编写 driver　2、接口出错发现较迟 3、系统轮廓形成较晚

回归测试：是指重新执行已经做过的测试的某个子集，以保证上述这些变化没有带来非预期的副作用。

5、**确认测试又称验收测试，它的目标是验证软件的有效性。确认测试阶段有两项工作**：有效性测试测试、软件配置审查。

6、**Alpha 测试**由用户在开发者的场所进行，并且在开发者对用户的“指导”下进行测试。Alpha 测试是在受控的环境中进行的。

7、**Beta 测试**是软件在开发者不能控制的环境中的“真实”应用。

6、 白盒法：又称为逻辑覆盖法，其测试用例选择，是按照不同覆盖标准确定的。

1. 白盒法常用的覆盖标准：

① 语句覆盖：选择足够的测试用例，使得程序中每个语句至少都能被执行一次。

② 判定覆盖：执行足够的测试用例，使得程序中每个判定至少都获得一次“真”值和“假”值。

③ 条件覆盖：执行足够的测试用例，使得判定中的每个条件获得各种可能的结果。

④ 判定/条件覆盖：执行足够的测试用例，使得判定中每个条件取到各种可能的值，并使每个判定取到各种可能的结果。

⑤ 条件组合覆盖：执行足够的例子，使得每个判定中条件的各种可能组合都至少出现一次。

6.点覆盖：选取足够多测试数据，使程序的每条可能路径至少经流过流图的每个结点一次。

7 边覆盖：选取足够多测试数据，使程序的每条可能路径至少经过流图中每条边一次。

8 路径覆盖：选取足够多测试数据，使程序的每条可能路径都至少执行一次。

常用的控制结构测试技术：基本路径测试、条件测试、循环测试。

基本路径测试：

第一步，根据过程设计结果画出相应的流图。

第二步，计算流图的环形复杂度。

第三步，确定线性独立路径的基本集合。

第四步，设计可强制执行基本集合中每条路径的测试用例

7、黑盒法：不考虑程序的内部结构与特性，只根据程序功能或程序的外部特性设计测试用

例。

等价分类法基本思想：根据程序的 I/O 特性，将程序的定义域划分为有限个等价区段 —“等价类”，从等价类中选择出的用例，具有“代表性”

等价类分为：有效等价类—对于程序的规格说明是合理的、有意义的输入数据构成的集合。

无效等价类 —对于程序的规格说明，是不合理的，是没有意义的输入数据构成的集合。

边值分析法基本思想： 选择等价类的边缘值作为测试用例，让每个等价类的边界都得到测试，选择测试用例既考虑输入亦考虑输出。

分析步骤：

A、先划分等价类。

B、选择测试用例，测试等价类边界。

边界选择原则：

A、按照输入值范围的边界。

B、按照输入/输出值个数的边界。

C、输出值域的边界。

D、输入/输出有序集的边界。

错误推测法的概念：凭经验或直觉推测可能的错误，列出程序中可能有的错误和容易发生错误的特殊情况，选择测试用例。

错误推测方法的基本思想：列举出程序中所有可能有的错误和容易发生错误的特殊情况，根据它们选择测试用例。例如：在单元测试时曾列出的许多在模块中常见的错误、以前产品测试中曾经发现的错误等，这些就是经验的总结。还有，输入数据和输出数据为 0 的情况、输入表格为空格或输入表格只有一行等。这些都是容易发生错误的情况，可选择这些情况下的例子作为测试用例

调试（也称为纠错）作为成功测试的后果出现，也就是说，调试是在测试发现错误之后排除错误的过程。

无论采用什么方法，调试的目标都是寻找软件错误的原因并改正错误。通常需要把系统地分析、直觉和运气组合起来，才能实现上述目标。一般说来，有下列 **3 种调试途径可以采用：蛮干法，回溯法，原因排除法。**

软件可靠性的定义：对于软件可靠性有许多不同的定义，其中多数人承认的一个定义是：软件可靠性是程序在给定的时间间隔内，按照规格说明书的规定成功地运行的概率

软件可用性的一个定义： 软件可用性是程序在给定的时间点，按照规格说明书的规定，成功地运行的概率。

第八章　维护

1. **软件维护**是指软件系统交付使用以后，为了改正错误或满足新的需求而修改软件的过程。按照不同的维护目的，**维护工作可分成 4 类**。

完善性维护：扩充原有系统的功能，提高原有系统的性能，满足用户的实际需要。

纠错性维护：对在测试阶段未能发现的，在软件投入使用后才逐渐暴露出来的错误的测试、诊断、定位、纠错以及验证、修改的回归测试过程。

适应性维护：要使运行的软件能适应运行环境的变动而修改软件的过程。

预防性维护：为了进一步改善软件的可靠性和易维护性，或者为将来的维护奠定更好的基础而对软件进行修改。

2.**软件的可维护性定性地定义为：** 维护人员理解、改正、改动或改进这个软件的难易程度。

重用：同一事物不做修改或稍加改动就在不同环境中多次重复使用。

提高可维护性是支配软件工程方法学所有步骤的关键目标。

第九章 面向对象方法引论

面向对象方法学的基本原则：尽可能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。

对象：在应用领域中有意义的、与所要解决的问题有关系的任何事物都可以作为对象，它既可以是具体的物理实体的抽象，也可以是人为的概念，或者是任何有明确边界和意义的东西。

对象的定义：（1）定义 1：对象时具有相同状态的一组操作的集合。（2）定义 2：对象是对问题域中某个东西的抽象，这种抽象反映了系统保存有关这个东西的信息或与它交互的能力。（3）定义 3：对象：：=<ID,MS,DS,MI>。其中，ID 是对象的标识或名字，MS 是对象中的操作集合，DS 是对象的数据结构，MI 是对象受理的消息名集合。

对象基本特点：1、以数据为中心；2、对象是主动的；3、实现了数据封装；4、本质上具有并行性；5、模块独立性好。

类：就是对具有相同数据和相同操作的一组相似对象的定义，也就是说，类是对具有相同属性和行为的一个或多个对象的描述，通常在这种描述中也包括对怎样创建该类的新对象的说明。类是支持继承的抽象数据类型，而对象就是类的实例。

类有私有变量（Private）和公有变量（Public）。缺省时，都属于私有的，只能由类内部其他成员来访问，不能由程序的其他部分来访问。这是一种实现封装的方法。

实例：就是由某个特定的类所描述的一个具体的对象。

消息：就是要求某个对象执行在定义它的那个类中所定义的某个操作的规格说明。消息既可以是数据流，也可以是控制流。一条消息可以发送给不同的对象，对消息的解释完全由接收信息的对象来完成，不同的对象对相同形式的消息可以有不同的解释。一个消息由 3 部分组成：1、接受消息的对象；2、消息选择符；3、零个或多个变元。

方法：就是对象所能执行的操作，也就是类中所定义的服务。描述了对象执行操作的算法，响应消息的方法。在 C++语言中称为成员函数。

属性：就是类中所定义的数据，它是对客观世界实体所具有的性质的抽象，在 C++语言中称为数据成员。

对象具有封装性的条件：1、有一个清晰的边界；2、有确定的接口；3、受保护的内部实现。

封装也就是信息隐藏，通过封装对外界隐藏了对象的实现细节。

继承：是子类自动地共享基类中定义的数据和方法的机制。继承，是指能够直接获得已有的性质和特征，而不必重复定义它们。

多态性：是指子类对象可以像父类对象那样使用，同样的消息既可以发送给父类对象也可以发送给子类对象。在 C++中，多态性是通过虚函数来实现的。

函数重载是指在同一作用域内的若干参数特征不同的函数可以使用相同的函数名字;**运算符重载**是指同一个运算符可以施加于不同类型的操作数上面。在 C++中函数重载是通过静态联编实现的。

面向对象的 3 种模型：1、对象模型 对象、类、层次和关系 2、动态模型 对象和系统的行为 3、功能模型 类似于高层的 DFD，描述穿越系统的信息流

对象模型表示静态的、结构化的系统的“数据”性质。

类图描述类及类与类之间的静态关系。类图是一种静态模型，它是创建其他 UML 图的基础。一个系统可以由多张类图来描述，一个类也可以出现在几张类图中。

功能模型：表示变化的系统的“功能”性质，它指明了系统应该“做什么”，因此更直接地反映了用户对目标系统的需求。

通常，功能模型由一组数据流图组成。UML 提供的用例图也是进行需求分析和建立功能模型的有力工具。以用例图建立起来的系统模型称为用例模型，它描述的是外部行为者所理解的系统功能。

用例图：1、系统；2、用例(use case)行为者感受到的一个完整的功能。用例是类，代表一类功能，用例的实例称为脚本。特征：（1）用例代表某些用户可见的功能，实现一个具体的用户目标；（2）用例总是被行为者启用的，并向行为者提供可识别的值；（3）用例必须是完整的。3、行为者(actor)与系统交互的人或其他系统。它代表一种角色。

用例之间的关系：1）扩展关系：向一个用例中添加一些动作后构成另一个用例，它们之间构成扩展关系。2）使用关系：一个用例使用另一个用例，它们之间构成使用关系。

扩展与使用的异同：描述一般行为的变化时采用扩展关系；两个或多个用例中出现重复描述时可采用使用关系。

用例建模：一个用例模型由若干幅用例图组成。创建用例模型的工作包括：定义系统、寻找行为者、寻找用例、描述用例、定义用例之间的关系、确认模型。

寻找行为者：下述问题有助于发现行为者： 1）谁将使用系统的主要功能？ 2）谁需要借助系统的支持来完成日常工作？ 3）谁来维护和管理系统？4）系统控制哪些硬件设备？5）系统需要与哪些其他系统交互？6）哪些人或系统对本系统产生的结果感兴趣？

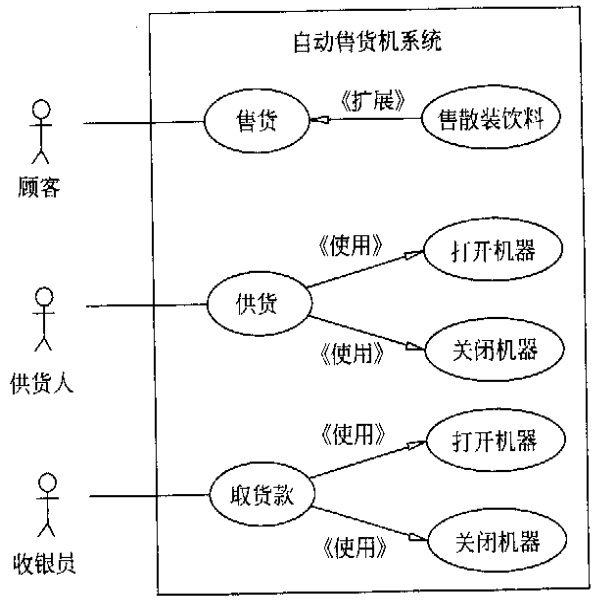


图 9.18　含扩展和使用关系的用例图

三种模型之间的关系：1）针对每个类建立的动态模型，描述了实例的生命周期或运行周期。2）状态转换驱使行为发生，这些行为在数据流图中被映射成处理，在用例图中被映射成用例，它们同时与类图中的服务相对应。3）功能模型中的处理（或用例）对应于对象模型中的类所提供的服务。4）数据流图中的数据存储，以及数据的源点/终点，通常是对象模型中的对象。5）数据流图中的数据流，往往是对象模型中对象的属性值，也可能是整个对象。6）用例图中的行为者，可能是对象模型中的对象。7）功能模型中的处理（或用例）可能产生动态模型中的事件。8）对象模型描述了数据流图中的数据流、数据存储以及数据源点/终点的结构。

第十章：面向对象分析

10.1　面向对象分析的基本过程

1.　3 个子模型与 5 个层次

（1）3 个子模型：对象模型（静态结构）；　动态模型（交互次序）；

功能模型（数据变换）。

（2）5 个层次：主题层；类与对象层；结构层；属性层；服务层。

2. **建立对象模型的步骤：**1、确定类与对象 2、确定关联 => 结构层 3、划分主题 4、确立属性 5、识别继承关系 6、反复修改

3. 建立动态模型的步骤：1、编写脚本 2、设想用户界面 3、画事件跟踪图 4、画状态图 5、审查动态模型 6、反复修改

3.

第11章 面向对象设计

1、**重用**也叫再用或复用，指同一事物不作修改或稍加改动就多次重复使用。 重用的三个层次：1）知识重用；2）方法和标准的重用；3）软件成分的重用。

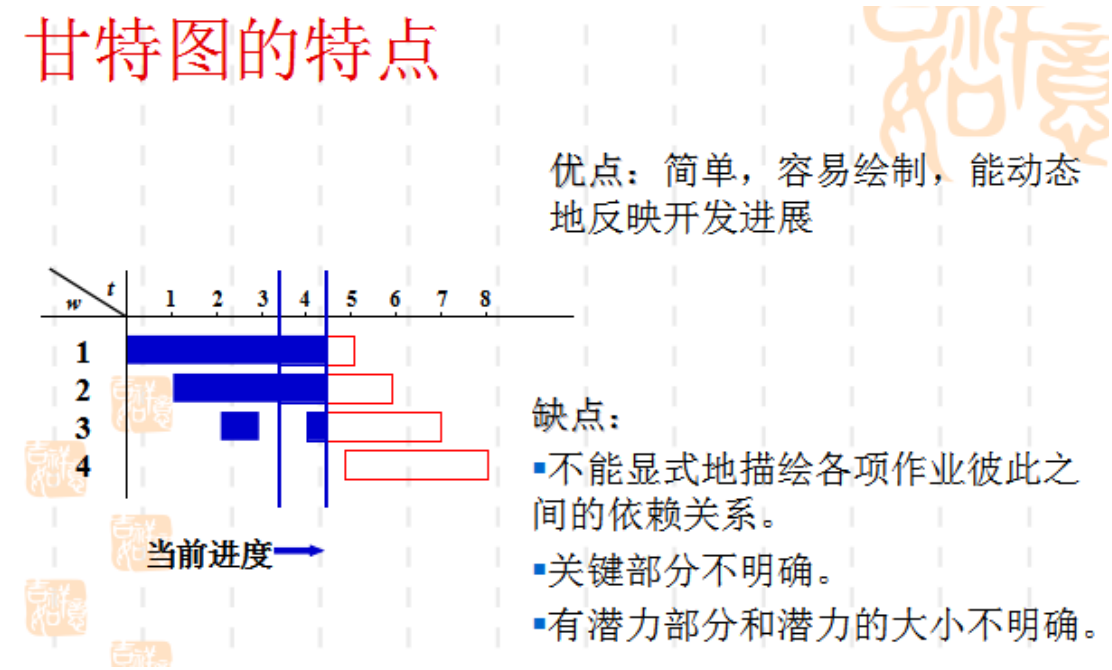
2、**软件成分的重用级别**：1）代码重用 a.　源代码剪贴 b.　源代码包含 c.　继承；2）设计结果重用；3）分析结果重用

3、类构件的重用方式：1）实例重用 2）继承重用 3）多态重用

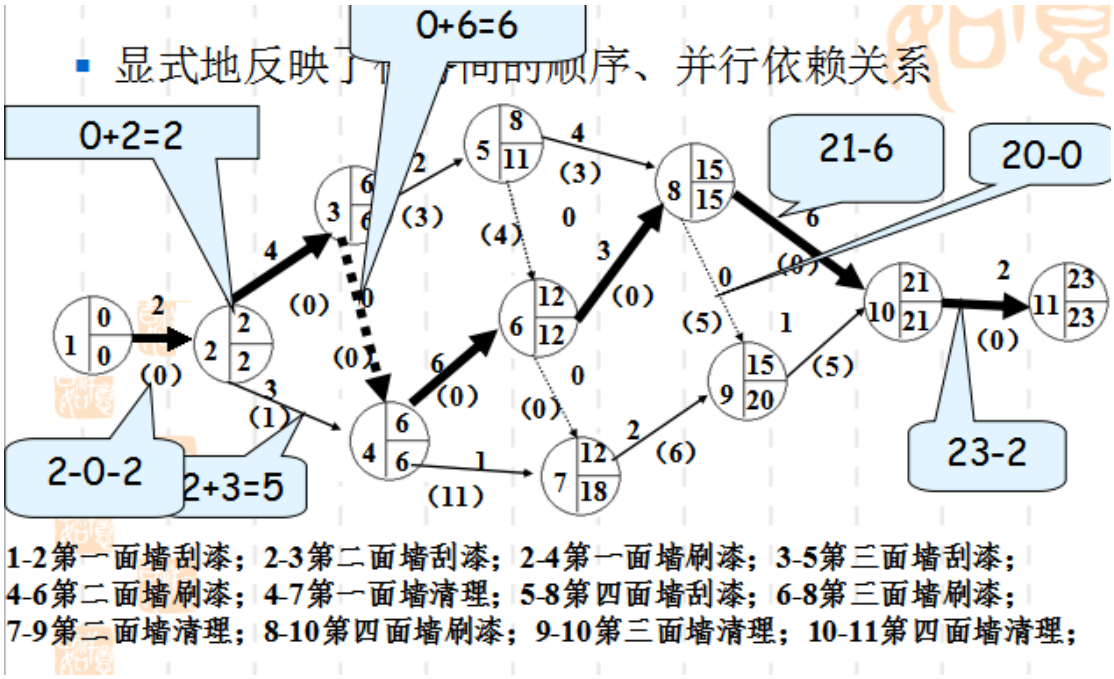
4、子系统之间的两种交互方式：1）客户-供应商关系 2）平等伙伴关系

第13章 软件项目管理

1、Gantt 图



2、工程网络图



3、**关键路径**上的事件必须准时发生，组成关键路径的作业的实际持续时间不能超过估计的持续时间，最早时刻和最迟时刻相同。

4、机动时间

一个作业可以有的全部机动时间等于它的结束事件的最迟时刻减去它的开始事件的最早时刻，再减去这个作业的持续时间：

机动时间=(LET)结束-(EET)开始-持续时间

5、**软件质量**是软件与明确地叙述的功能和性能需求、文档中明确描述的开发标准以及任何专业开发的软件产品都应该具有的隐含特征相一致的程度。

6、**软件质量保证的措施**主要有：基于非执行的测试、基于执行的测试、程序正确性证明

7、软件配置项：1、计算机程序（源代码和可执行程序）2、描述计算机程序的文档（供技术人员或用户使用）3、数据（程序内包含的或在程序外的）

8、 **基线**是已经通过了正式复审的规格说明或中间产品，它可以作为进一步开发的基础，并且只有通过正式的变化控制过程才能改变它。基线就是通过了正式复审的软件配置项。

9、**软件配置管理过程步骤：**1、标识软件配置中的对象 2、版本控制 3、变化控制 4、配置审计 5、状态报告

10、**能力成熟度（CMM）**的 5 个等级从低到高依次是：初始级、可重复级、已定义级、已管理级、优化级

特点：1、初始级：软件过程的特征是无序的，混乱的。

2、可重复级：（1）进行较为现实的承诺。（2）主要是逐个项目地建立基本过程管理条例来加强过程能力。（3）管理工作主要跟踪软件经费支出、进度及功能。（4）采用基线（baseline）来标志进展、控制完整性。（5）定义了软件项目的标准，并相信它、遵循它。（6）通过子合同建立有效的供求关系。

3、已定义级：（1）无论管理方面或工程方面的软件过程都已文件化、标准化，并综合成软件开发组织的标准软件过程。（2）软件过程标准被应用到所有的工程中，用于编制和维护软件。（3）在从事一项工程时，产品的生产过程、花费、计划以及功能都是可以完全控制的，从而软件质量也可以控制。（4）软件工程过程组（SEPG）负责软件过程活动。（5）在全组织范围内安排培训计划。

4、已管理级：（1）制定了软件过程和产品质量的详细而具体的度量标准。软件过程和产品的质量都可以被理解和控制。（2）软件组织的能力是可预见的。（3）组织的度量工程保证所有项目对生产率和质量进行度量，并作为重要的软件过程活动。（4）具有良好定义及一致的度量标准来指导软件过程，并作为评价软件过程及产品的定量基础。（5）在开发组织内已建立软件过程数据库，保存收集到的数据，可用于各项目的软件过程

5、优先级：（1）整个组织特别关注软件过程改进的持续性、预见及增强自身。（2）加强定量分析，通过来自过程的质量反馈和吸收新观念、新科技，使软件过程能不断地得到改进。（3）根据软件过程的效果，进行成本/效益分析，从成功的软件过程实践中吸取经验，加以总结。（4）组织能找出过程的不足并预先改进。（5）对软件过程的评价和对标准软件过程的改进，都在全组织内推广。