第 1 章软件工程学概述

1.1软件危机

1.1.1软件危机的介绍

软件危机(软件萧条、软件困扰: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。

软件危机包含下述两方面的问题:

如何开发软件,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中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。

1.1.2产生软件危机的原因

(1与软件本身的特点有关

(2与软件开发与维护的方法不正确有关

1.1.3消除软件危机的途径

对计算机软件有正确的认识。

认识到软件开发是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。应该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,并继续研究探索。

应该开发和使用更好的软件工具。

总之,为了解决软件危机,既要有技术措施(方法和工具,又要有必要的组织管理措施。

1.2

1.2.1软件工程的介绍

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,这就是软件工程。(期中考

软件工程的本质特性:

软件工程关注于大型程序的构造

软件工程的中心课题是控制复杂性

软件经常变化

开发软件的效率非常重要

和谐地合作是开发软件的关键

软件必须有效地支持它的用户

在软件工程领域中是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

1.2.2软件工程的基本原理

用分阶段的生命周期计划严格管理

坚持进行阶段评审

实行严格的产品控制

采用现代程序设计技术

结果应能清楚地审查

开发小组的人员应该少而精

承认不断改进软件工程实践的必要性

1.2.3软件工程方法学

软件工程包括技术和管理两方面的内容。

软件工程方法学 3 要素:方法、工具、过程

1.传统方法学(生命周期方法学或结构化范型——强调自顶向下

2.面向对象方法学——强调主动地多次反复迭代

面向对象方法学 4 个要点:对象、类、继承、消息

1.3软件生命周期(必考

三个时期八个阶段:软件生命周期由软件定义、软件开发和运行维护(也称为软件维护三个时期组成,每个时期又进一步划分成若干个阶段。

三个时期:八个阶段:

软件生命周期软件定义

软件开发

软件维护

问题定义

可行性研究

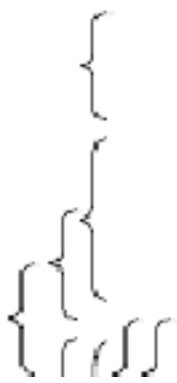
需求分析

概要设计

详细设计

编码和单元测试

综合测试



运行维护

系统设计

系统实现

1.4软件过程

1.4.1瀑布模型

1.4.2快速原型模型

1.4.3增量模型

1.4.4螺旋模型

1.4.5喷泉模型

第 2 章可行性研究

2.1可行性研究的任务

可行性研究的目的:

不是解决问题,而是确定问题是否值得去解决。

可行性研究的实质:

进行一次大大压缩简化了的系统分析和设计的过程,也就是在较高层次上以较抽象的方式进行的系统分析和设计的过程。

可行性研究的内容:

首先进一步分析和澄清问题定义,导出系统的逻辑模型;

然后从系统逻辑模型出发,探索若干种可供选择的主要解法(即系统实现方案;

对每种解法都研究它的可行性,至少应该从三方面研究每种解法的可行性。

主要方面:

技术可行性,经济可行性,操作可行性,

其他方面:

运行可行性,法律可行性,

2.2可行性研究过程

1.复查系统规模和目标

2.研究目前正在使用的系统

3.导出新系统的高层逻辑模型

4.进一步定义问题

5.导出和评价供选择的解法

6.推荐行动方针

7.草拟开发计划

8.书写文档提交审查

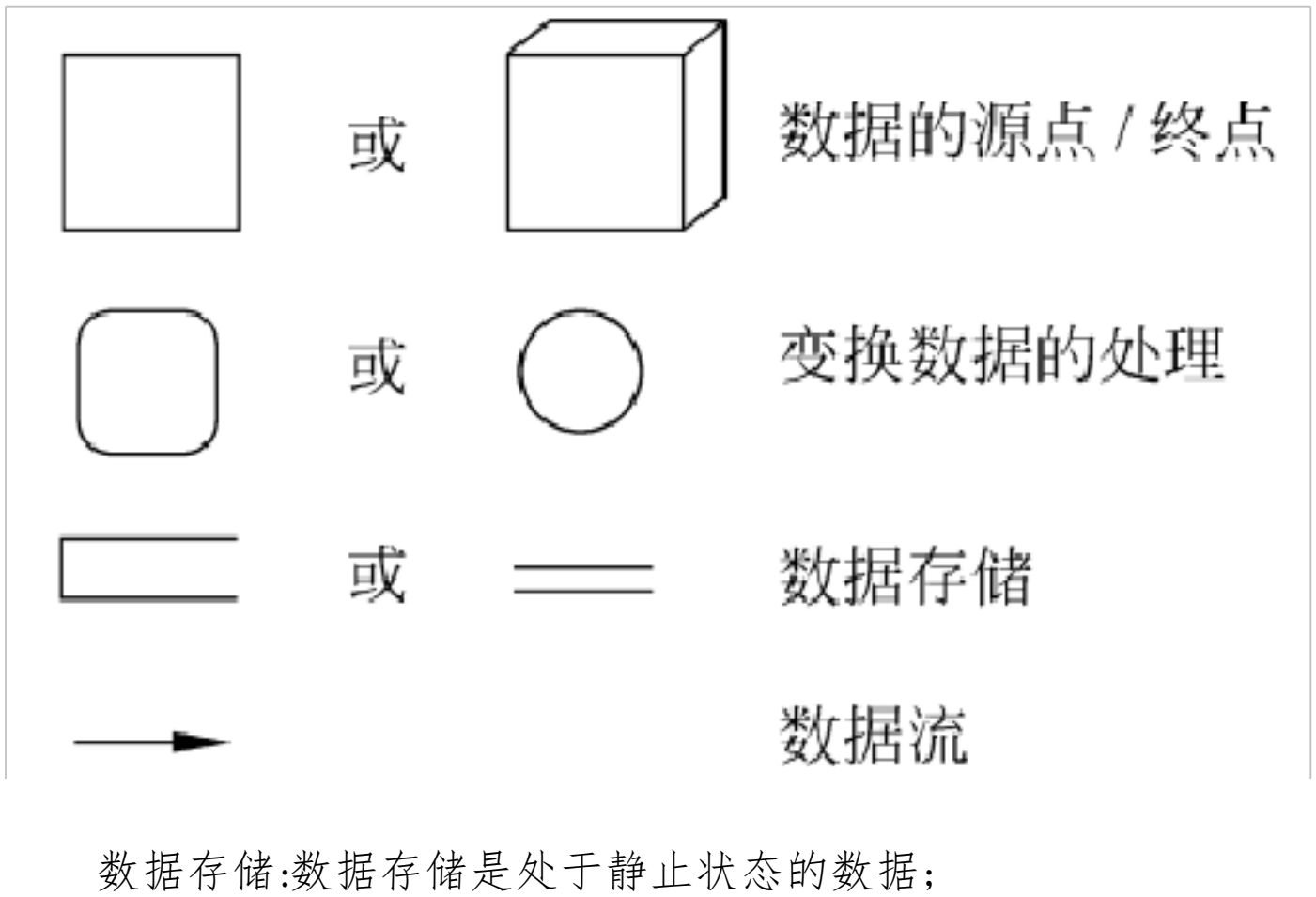
2.3系统流程图

系统流程图:是概括地描绘物理系统的传统工具。表达的是数据在系统各部件之间流动的情况,而不是对数据进行加工处理的控制过程。

2.4数据流图

2.4.1符号

基本符号:



数据存储:数据存储是处于静止状态的数据;

数据流:数据流是处于运动中的数据。

附加符号:

星号(*:表示 “与”关系

加号(+:表示 “或”关系

异或(⊕:表示互斥关系

2.5数据字典

数据流图和数据字典共同构成系统的逻辑模型。

2.5.1数据字典的内容

数据字典的组成:数据流数据流分量(即数据元素 数据存储处理

2.5.2定义数据的方法

方法:对数据自顶向下分解。

数据组成方式(三种基本类型:顺序选择重复附加类型:可选

符号:

=意思是等价于(或定义为;

+意思是和(即,连接两个分量;

[]意思是或(即,从方括弧内列出的若干个分量中选择一个,通常用“|”隔开供选择的分量;

{ }意思是重复(即,重复花括弧内的分量;常常使用上限和下限进一步注释表示重复的花括弧。

(意思是可选(即,圆括弧里的分量可有可无。

2.5.3数据字典的实现

计算机实现人工实现

2.6成本/效益分析

2.6.1成本估计:1.代码行技术 2.任务分解技术

3.自动估计成本技术

2.6.2成本/效益分析的方法

成本/效益分析涉及的 4 个概念:

1.货币的时间价值

2.投资回收期

3.纯收入

4. 投资回收率:P = F1/(1 + j + F2/(1 + j 2 + …+ Fn(1 + j n

第 3 章需求分析

需求分析的任务:

需求分析是软件定义时期的最后一个阶段,它的基本任务是准确地回答“系统必须做什么?”这个问题。

确定系统必须完成哪些工作,也就是对目标系统提出完整、准确、清晰、具体的要求。

系统分析员应该写出软件需求规格说明书,以书面形式准确地描述软件需求

3.1需求分析的任务

确定对系统的综合要求

分析系统的数据要求

导出系统的逻辑模型

修正系统开发计划

3.1.1确定对系统的综合要求

1.功能需求

2.性能需求

3.可靠性和可用性需求

4.出错处理需求

5.接口需求

7.逆向需求

8.将来可能提出的要求

3.1.2分析系统的数据要求

建立数据模型　　ER 图

描绘数据结构——层次方框图和 Warnier 图

数据结构规范化

3.2与用户沟通获取需求的方法

访谈:1.正式访谈 2.非正式访谈 3.调查表 4.情景分析技术

面向数据流自顶向下求精

简易的应用规格说明技术

快速建立软件原型:(1第四代技术(4GL(2 可重用的软件构件 (3形式化规格说明和原型环境

3.3分析建模与规格说明

3.3.1分析建模

需求分析过程应该建立 3 种模型:数据模型功能模型行为模型

数据字典是分析模型的核心

实体-联系图用于建立数据模型的图形

数据流图是建立功能模型的基础

3.4实体-联系图

数据模型中包含3种相互关联的信息:数据对象、数据对象的属性、数据对象彼此间相互连接的关系

3.4状态转换图

3.6.1状态

状态图分类:

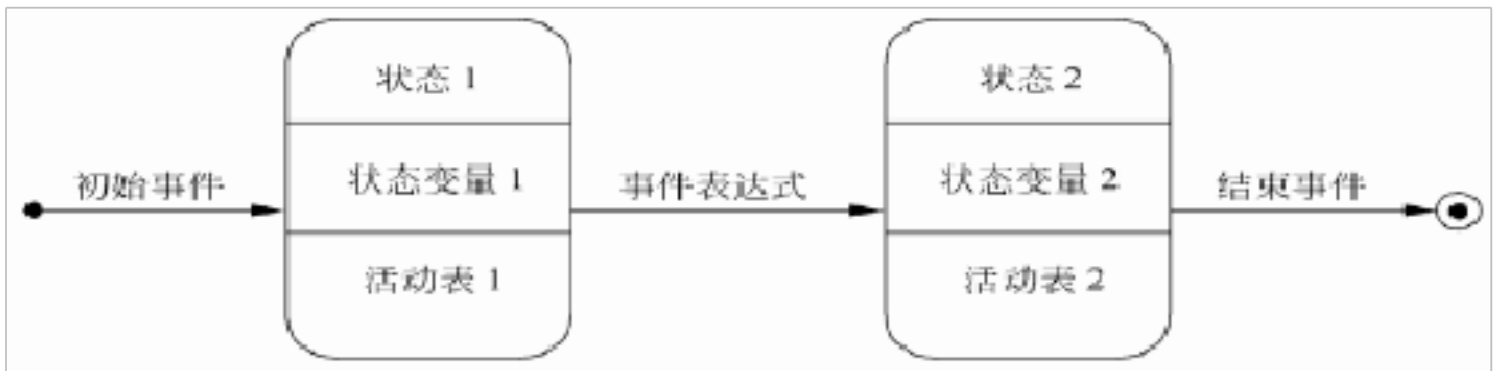
表示系统循环运行过程,通常不关心循环是怎样启动的。

表示系统单程生命期,需要标明初始状态和最终状态。

3.6.2事件

事件就是引起系统做动作或(和转换状态的控制信息。

3.6.3符号



3.7其他图形工具

3.7.1层次方框图

3.7.2Warnier 图

3.7.3IPO 图

(重点

3.8.1从哪些方面验证软件需求的正确性

一致性完整性现实性有效性

第五章总体设计

5.1设计过程

由两个主要阶段组成:

系统设计阶段,确定系统的具体实现方案:设想供选择的方案选取合理的方案推荐最佳方案

结构设计阶段,确定软件结构:功能分解设计软件结构设计数据库制定测试文档书写文档审查和复查

5.2设计原理

5.2.1模块化

模块化的作用:

采用模块化原理可以使软件结构清晰,不仅容易设计也容易阅读和理解。

模块化使软件容易测试和调试,因而有助于提高软件的可靠性。

模块化能够提高软件的可修改性。

模块化也有助于软件开发工程的组织管理。

5.2.2抽象

5.2.3逐步求精

5.2.5模块独立

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。

七种内聚的优劣评分结果：

高内聚:功能内聚

顺序内聚

中内聚:通信内聚

过程内聚

低内聚:时间内聚

逻辑内聚

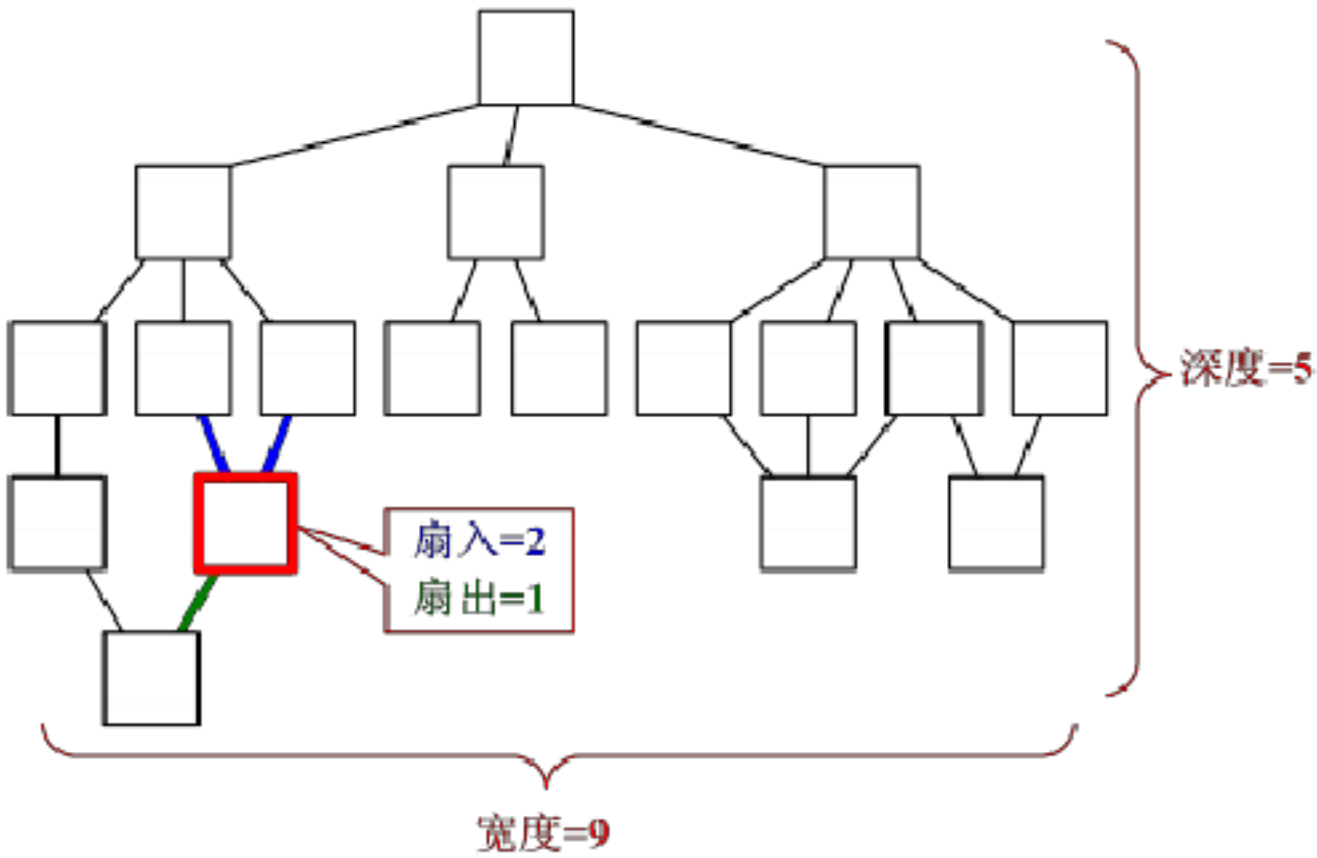
偶然内聚

5.3启发规则

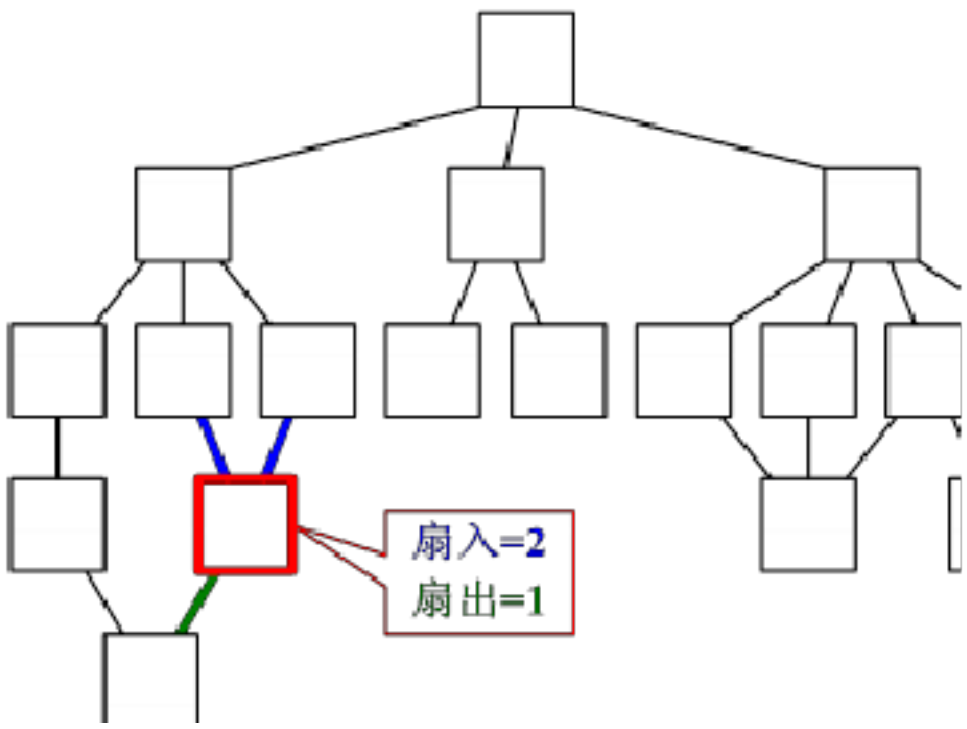
1.改进软件结构提高模块独立性

2.模块规模应该适中

3.深度、宽度、扇出和扇入都应适当



4.



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

5.4描绘软件结构的图形工具

5.4.1层次图和 HIPO 图

1.层次图(H 图

2.HIPO 图

5.4.2结构图

5.5面向数据流的设计方法

结构化设计方法(简称 SD 方法,也就是基于数据流的设计方法。

5.5.1概念

面向数据流的设计方法把信息流映射成软件结构,信息流的类型决定了映射的方法。信息流有两种类型:变换流事务流

第 6 章详细设计

6.1结构程序设计

经典的结构程序设计:

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;

扩展的结构程序设计:

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;

修正的结构程序设计:

再加上允许使用 LEAVE(或 BREAK 结构。

6.2人机界面设计

6.2.1设计问题

设计人机界面过程中会遇到的4个问题:

系统响应时间:长度易变性

用户帮助设施:集成的帮助设施附加的帮助设施

出错信息处理

命令交互

6.2.3人机界面设计指南

一般交互指南

信息显示指南

数据输入指南

6.3过程设计的工具

6.3.1程序流程图(程序框图

程序流程图的主要缺点:

程序流程图本质上不是逐步求精的好工具,它诱使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。

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。

程序流程图不易表示数据结构。

6.3.2盒图(N-S图

盒图具有下述特点:

功能域明确。

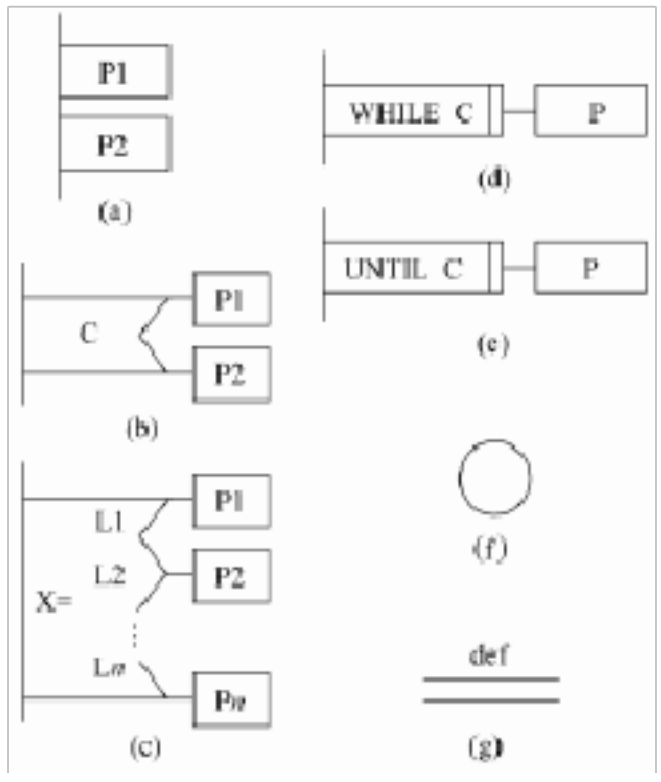
不可能任意转移控制。

很容易确定局部和全程数据的作用域。

很容易表现嵌套关系,也可以表示模块的层次结构。

6.3.3 PAD 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。



PAD 图的主要优点如下:

使用表示结构化控制结构的 PAD 符号设计出来的程序必然是结构化程序。

PAD 图所描绘的程序结构十分清晰。

PAD 图表现程序逻辑易读、易懂、易记。

容易将 PAD 图转换成高级语言源程序,这种转换可用软件工具自动完成。

即可表示程序逻辑,也可描绘数据结构。

PAD 图的符号支持自顶向下、逐步求精方法的使用。

6.3.4判定表

判定表却能够清晰地表示复杂的条件组合与应做的动作之间的对应关系。

所有条件	条件组合矩阵
所有动作	条件组合 对应的动作

判定表的缺点:

判定表的含义不是一眼就能看出来的,初次接触这种工具的人理解它需要有一个简短的学习过程。

当数据元素的值多于两个时,判定表的简洁程度也将下降。

6.3.5判定树

判定树的优点:

它的形式简单,一眼就可以看出其含义,因此易于掌握和使用。

判定树的缺点:

简洁性不如判定表,数据元素的同一个值往往要重复写多遍,而且越接近树的叶端重复次数越多。

画判定树时分枝的次序可能对最终画出的判定树的简洁程度有较大影响。

6.3.6过程设计语言(伪码

伪代码的基本控制结构:

简单陈述句结构:避免复合语句。

判定结构:IF_THEN_ELSE 或 CASE_OF 结构。

选择结构:WHILE_DO 或 REPEAT_UNTIL 结构。

PDL 的优点:

可以作为注释直接插在源程序中间。有助于保持文档和程序的一致性,提高了文档的质量。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。

已经有自动处理程序存在,而且可以自动由 PDL 生成程序代码。

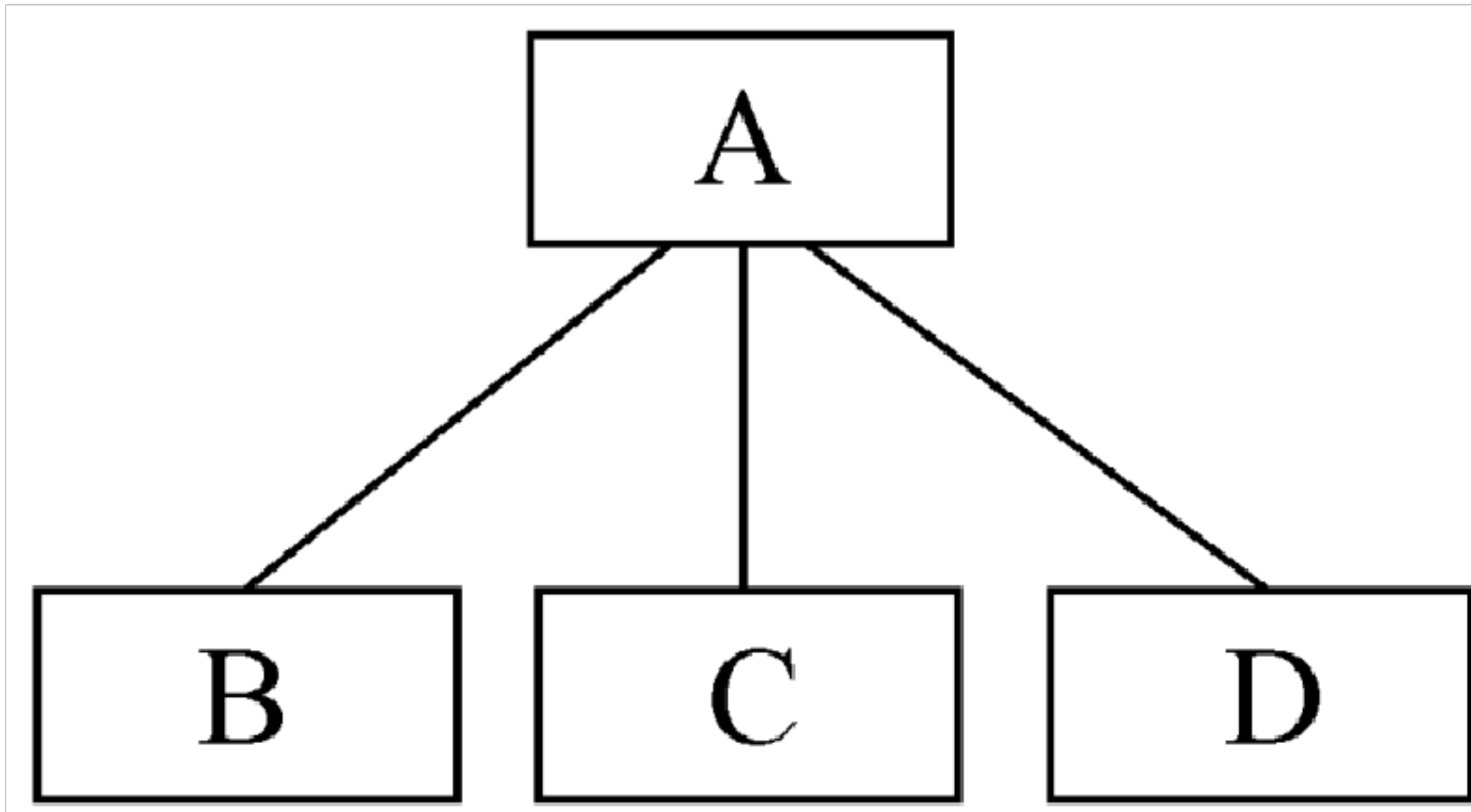
PDL 的缺点:

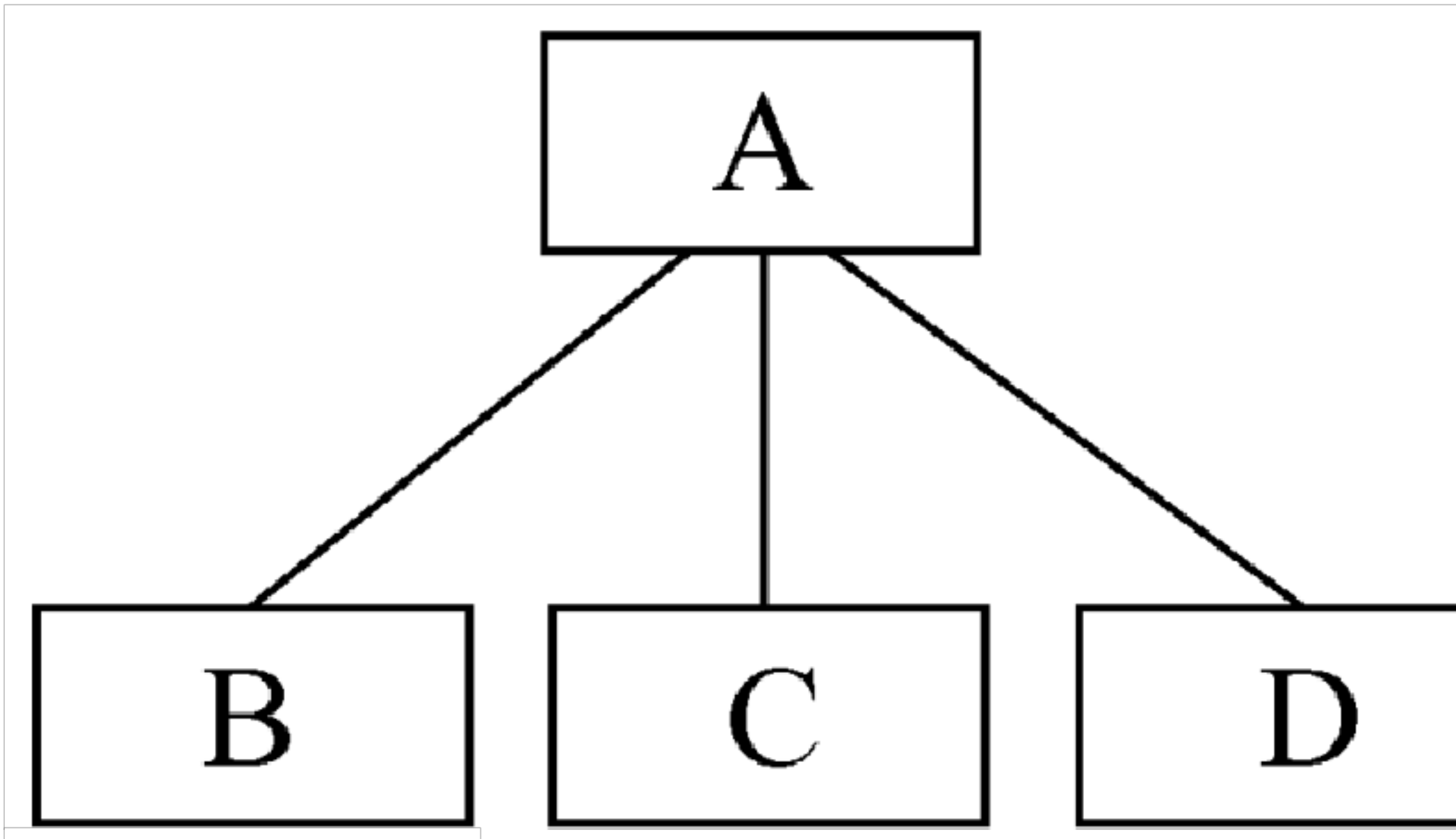
不如图形工具形象直观,描述复杂的条件组合与动作间的对应关系时,不如判定表清晰简单。

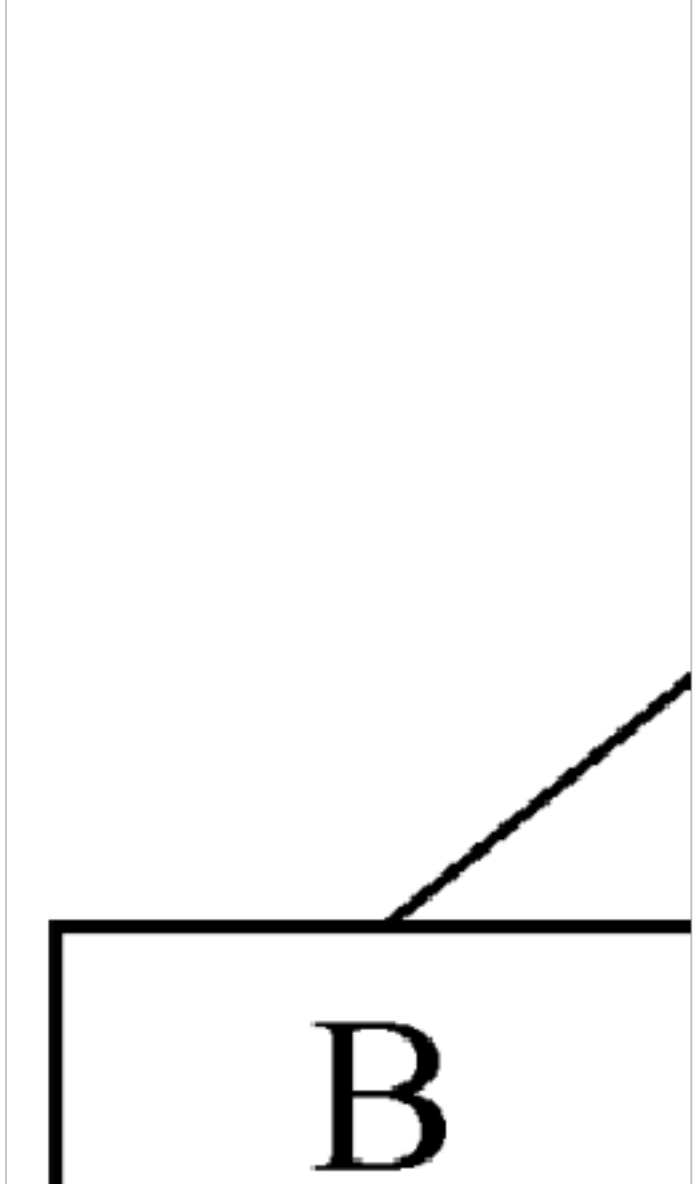
6.4面向数据结构的设计方法

面向数据结构的设计方法的最终目标是得出对程序处理过程的描述。

6.4.1Jackson


A
B
C
D


A
B
C
D


B

A 由 B、C、D 3 个元素顺序组成根据条件 A 是 B 或 C 或 D 中的某一个 A 由 B 出现 N 次(N ≥0 组成

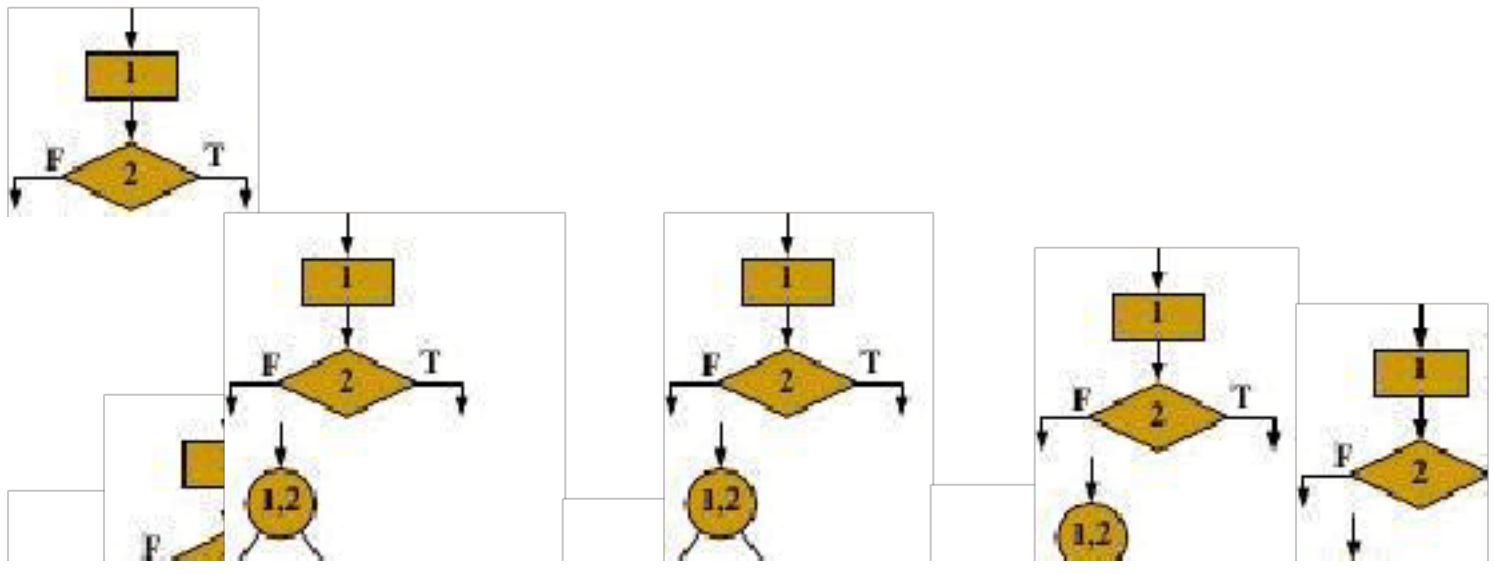
6. 4. 2改进的 Jackson 图

6.4.3 Jackson 方法

6. 5程序复杂程度的定量度量

6.5.1 McCabe 方法

1. 流图(程序图



2. 计算环形复杂度的方法

V(G=流图中的区域数

V(G=E-N+2

其中 E 是流图中的边数,N 是结点数

V(G=P+1

其中 P 是流图中判定结点的数目

6.5.2 Halstead 方法

令 N1 为程序中运算符出现的总次数,N2 为操作数出现的总次数,程序长度 N 定义为:

N=N1+N2

程序中使用的不同运算符(包括关键字的个数 n1,以及不同操作数(变量和常数的个数 n2。预测程序长度的公式如下:

H = n1 log2n1 + n2 log2n2

预测程序中包含错误的个数的公式如下:

E = N log2 (n1+n2/3000

第 7 章实现

编码和测试统称为实现。

7. 1编码

7. 1. 1选择程序设计语言

主要的实用标准:

系统用户的要求

可以使用的编译程序

可以得到的软件工具

工程规模

程序员的知识

软件可移植性要求

软件的应用领域

7.1.2编码风格

1.程序内部的文档:恰当的标识符适当的注解程序的视觉组织

2.数据说明

3.语句构造

4.输入输出

5.效率:程序运行时间存储器效率输入输出的效率

7.2软件测试基础

7.2.1软件测试的目标

测试是为了发现程序中的错误而执行程序的过程;

好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案;

成功的测试是发现了至今为止尚未发现的错误的测试。

7.2.3测试方法

黑盒测试(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;

是在程序接口进行的测试。

白盒测试(结构测试:

把程序看成装在一个透明的盒子里;