面向企业管理信息系统的软件技术和开发方法

罗邵武　柴跃廷　张　宁　唐旭东　编著

CASE
Object—Oriented
4GT　4GL
Prototyping

兵器工业出版社

面向企业管理信息系统的软件技术和开发方法

罗邵武、柴跃廷、张宇、唐旭东　编著

兵器工业出版社

（京）新登字049号

面向企业管理信息系统的软件技术和开发方法

罗邵武　柴跃廷　张　宇　唐旭东

兵器工业出版社出版发行

（北京市海淀区车道沟10号）

各地新华书店经销

北方工业大学印刷厂印装

*

开本：787×1092　1/16　印张：18　字数：446千字

1992年12月第1版　1992年12月第1次印刷

印数：2500　定价：16.00元

ISBN　7-80038-491-8/TP·37

内 容 简 介

国内企业正在深化改革、强化企业内部的管理，计算机信息系统这种现代化的技术手段已被越来越多的工业界人士所认识，而企业管理信息系统正适合于信息/知识密集型计算机在企业管理中的应用。本书论述了有关人工智能、知识处理的基本原理和发展趋势，论述了软件的基本特征，大型信息系统开发的方法论——软件工程、信息工程、面向对象的系统分析与设计方法、原型法。同时也介绍了信息系统开发的较新技术——第四代语言的基本原理及软件发展的最新趋势——CASE 技术。国外应用 CASE 技术的经验表明：只有对软件工程、信息工程有关思想、方法有较深的了解，才能成功地应用 CASE 技术。本书通俗易懂，论述充分，结构严谨。既可作为高等院校热心于计算机应用开发的广大师生、工厂企业计算中心的有关技术人员以及服务于软件产业的广大工程技术人员的软件技术参考书，也可作为有关领导系统全面地了解有关信息系统开发的方法和软件技术的参考读物。

前　言

软件已成为许多产品、系统实现智能自动化的手段。随着计算机技术的研究和发展、软件技术越来越成为一种通用的问题求解策略。然而软件生产率不高这一“软件危机”问题使得软件开发速度赶不上待开发的软件系统的积累速度。怎样才能提高软件的生产率呢？其一是通过培养更多的软件工程师或使计算机系统更加容易使用，使普通用户也能参与软件开发进程，从而增加软件开发人员的数量。其二则是教会计算机建立软件系统，这就是软件开发自动化，也称为计算机辅助软件工程（CASE）。CASE 是一条充满希望之路，但是要想实现软件开发过程自动化，首先就得充分了解软件本身的特性以及软件生产的规律，力求使软件生产过程规范化、工程化，减少其开发过程中引入的开发者个人的“艺术创造特性”，才能使软件开发过程实现自动化成为可能。而软件工程学、信息工程学已为实现软件开发自动化奠定了理论基础。80 年代后期，国外特别是美国推出和发展了 CASE 技术，随着应用和研究的深入，不久便发现 CASE 工具既有应用成功的范例，又存在着应用失败的事例。通过调查研究发现：要想使 CASE 工具应用成功，对用户进行有关软件工程、信息工程等方法论的教育要比对工具本身使用的训练重要得多。CASE 技术给出了解决软件危机的一个技术方向，但还有严峻的人员和管理方面的问题有待解决。总之，要想提高软件生产率没有一种灵丹妙药。要取得成功，唯有使用最好的人才、最好的方法论和最好的工具。

作者近几年一直在从事计算机综合制造系统（CIMS）及有关方面的研究工作，发现软件开发同样需要计算机综合制造（CIM）技术的哲理，也就是软件生命周期的各个阶段同样需要，并且能够利用计算机给予辅助，为此我们组织起来编写这样一本技术参考书，它综合地论述了软件新技术的有关思想、方法和趋势，以期能使有关读者从中获得信息和收益。

全书内容由罗邵武组织和审阅校对。其中：

第一章　软件和软件工程	罗邵武编写
第二章　信息、知识处理与信息工程	罗邵武编写
第三章　面向对象的系统分析与设计方法	柴跃廷编写
第四章　原型和原型法	张　宇编写
第五章　第四代语言的基本概念和原理	张　宇编写
第六章　计算机辅助软件工程（CASE）技术	唐旭东编写

作者在编写本书的过程中得到清华大学自动化系李芳芸副教授的关心和鼓励，在此表示感谢！

作者

1992 年 1 月于清华园

目　录

第一章　软件和软件工程……（1）
1.1　软件技术背景 ……（1）
1.1.1　软件的重要性 ……（1）
1.1.2　软件发展历史 ……（2）
1.1.3　软件工程背景 ……（3）
1.2　软件基本概念 ……（4）
1.2.1　软件特征 ……（4）
1.2.2　软件构成 ……（6）
1.2.3　软件应用领域种类 ……（7）
1.3　软件危机 ……（8）
1.3.1　问题 ……（9）
1.3.2　原因 ……（9）
1.3.3　软件神话 ……（10）
1.4　软件工程模式和有关技术 ……（12）
1.4.1　软件工程概念 ……（12）
1.4.2　经典的生命周期模式 ……（13）
1.4.3　原型法 ……（14）
1.4.4　第四代技术 ……（15）
1.4.5　组合模式 ……（17）
1.5　软件工程广义模式 ……（17）
1.5.1　分析、定义阶段 ……（17）
1.5.2　设计、实现阶段 ……（24）
1.5.3　运行、维护阶段 ……（34）
1.6　综述 ……（35）
第二章　信息、知识处理、信息工程……（39）
2.1　企业管理信息系统概述 ……（39）
2.2　计算机文件存储媒介与文件组织方法 ……（42）
2.2.1　磁带机 ……（43）
2.2.2　磁盘机 ……（44）
2.2.3　文件组织法 ……（46）
2.3　形式语言语法分析 ……（51）
2.3.1　概述 ……（51）
2.3.2　语法分析原理 ……（52）
2.3.3　通用语法分析处理程序生成器 ……（58）

2.4 人工智能 …… (63)
2.4.1 理论基础 …… (63)
2.4.2 知识系统技术背景 …… (68)
2.4.3 思维模拟水平和方法 …… (71)
2.5 信息工程与有关的结构化分析设计技术 …… (73)
2.5.1 概述 …… (73)
2.5.2 IDEF 方法 …… (74)
2.5.3 DFD 方法 …… (86)
2.5.4 JACKSON 方法 …… (90)
2.5.5 LCP 方法（Warnier 方法） …… (91)
2.5.6 小结 …… (98)
2.6 综述 …… (99)
第三章 面向对象的系统分析与设计方法 …… (102)
3.1 概述 …… (102)
3.2 基本概念 …… (104)
3.2.1 对象、操作、消息和信息隐藏 …… (104)
3.2.2 类、实例和继承性 …… (105)
3.2.3 对象描述 …… (107)
3.3 面向对象的系统分析与设计方法 …… (108)
3.3.1 定义所研究的问题 …… (109)
3.3.2 非形式化的求解策略 …… (111)
3.3.3 将求解策略形式化 …… (111)
3.3.4 一种可供选择的方法 …… (120)
3.4 面向对象的系统分析设计方法的生命周期 …… (125)
3.4.1 各种系统分析设计方法的比较 …… (126)
3.4.2 各种系统开发方法的配合应用 …… (127)
3.4.3 面向对象的系统开发方法 …… (128)
3.4.4 面向对象的系统分析与设计方法生命周期的描述 …… (130)
3.5 综述 …… (132)
第四章 原型和原型法 …… (135)
4.1 原型法的基本概念 …… (135)
4.1.1 经典系统生命周期法 …… (135)
4.1.2 原型法 …… (137)
4.1.3 原型法的优缺点 …… (139)
4.1.4 小结 …… (142)
4.2 原型法的基本原理 …… (143)
4.2.1 信息系统开发实例分析 …… (143)
4.2.2 原型构造方法 …… (145)
4.2.3 各类原型的构造方法 …… (149)

4.2.4 小结 …… (153)
4.3 原型开发生命周期 …… (154)
4.3.1 广义原型构造周期 …… (154)
4.3.2 广义原型构造周期的几种变化 …… (158)
4.3.3 小结 …… (159)
4.4 原型构造方法的应用分析 …… (160)
4.4.1 原型构造方法的目标和规划 …… (160)
4.4.2 原型构造方法应用项目的选取策略 …… (161)
4.4.3 原型构造方法模式 …… (162)
4.4.4 原型构造工具 …… (165)
4.4.5 原型法应用的管理问题 …… (166)
4.4.6 原型法应用的成功范例 …… (170)
4.4.7 小结 …… (171)
第五章 第四代语言的基本概念和原理 …… (172)
5.1 第四代语言的起源 …… (172)
5.1.1 计算机程序设计语言综述 …… (172)
5.1.2 第四代语言技术背景 …… (174)
5.1.3 第四代语言应用成功范例 …… (180)
5.1.4 小结 …… (181)
5.2 第四代语言的环境 …… (182)
5.2.1 引言 …… (182)
5.2.2 第四代语言的分类 …… (185)
5.2.3 第四代语言的标准和发展趋势 …… (187)
5.2.4 各种部门应用第四代语言情况综述 …… (190)
5.2.5 小结 …… (194)
5.3 第四代语言的基本原理 …… (195)
5.3.1 概述 …… (195)
5.3.2 第四代语言的基本特性 …… (195)
5.3.3 第四代语言工具 …… (202)
5.3.4 第四代语言的选择标准 …… (208)
5.3.5 小结 …… (213)
5.4 综述 …… (213)
第六章 计算机辅助软件工程（CASE） …… (217)
6.1 概述 …… (217)
6.1.1 CASE 基本概念 …… (217)
6.1.2 CASE 卓越之处 …… (218)
6.1.3 典型 CASE 技术 …… (219)
6.1.4 CASE 发展历史 …… (220)
6.1.5 CASE 与其它软件技术的联系 …… (222)

6.2 CASE 系统 …… (225)
6.2.1 CASE 支持下的软件开发环境概述 …… (225)
6.2.2 图形功能 …… (226)
6.2.3 检错问题 …… (229)
6.2.4 CASE 信息库 …… (231)
6.2.5 软件工具的集成 …… (234)
6.2.6 CASE 硬件平台的几种形式 …… (235)
6.2.7 CASE 系统硬件体系的基本形式 …… (238)
6.3 CASE 对软件开发过程的支持 …… (239)
6.3.1 重视软件生命周期中的前期 …… (240)
6.3.2 原型法工具 …… (240)
6.3.3 模拟功能 …… (240)
6.3.4 代码生成 …… (241)
6.3.5 对结构化方法论的支持 …… (241)
6.3.6 CASE 技术下的软件生命周期 …… (245)
6.4 CASE 工具的种类 …… (249)
6.4.1 CASE 工具的分类 …… (249)
6.4.2 CASE 工具箱 …… (250)
6.4.3 CASE 工作台 …… (257)
6.4.4 CASE 方法论指南工具 …… (258)
6.5 CASE 应用情况及考虑因素 …… (259)
6.5.1 CASE 应用分析 …… (259)
6.5.2 软件问题的多面性及解决办法 …… (261)
6.5.3 是否应用 CASE 技术的决策方法 …… (262)
6.5.4 CASE 实施计划的制定 …… (263)
6.6 90 年代软件技术发展趋势展望 …… (265)
6.6.1 软件技术现状 …… (266)
6.6.2 软件技术发展动态 …… (266)
6.6.3 软件工具的发展趋势 …… (267)
6.6.4 比 CASE 开发环境更先进的环境 …… (268)
6.6.5 方法论驱动器 …… (274)

第一章　软件和软件工程

1.1　软件技术背景

1.1.1　软件的重要性

许多工厂和工业部门都将信息技术视为竞争武器，因为市场竞争从未象现在这样变得如此残酷——对产品的价格、质量、性能和交货信誉异常敏感。而软件是信息技术的一个重要组成部分。实际上，现在许多事物之间的区别是用软件来衡量的。软件（以及相关数据库）提供的信息的完整性和即时性是相互竞争公司之间的主要区别，软件的设计和用户友好性是产品性软件与一般计算程序的主要区别，两种类似的工业或家用产品的区分常常取决于各自产品中的软件提供的智能和性能。总而言之，正是软件产生了许多事物之间的差异。

在计算机系统发展的前30年，主要的挑战是如何开发计算机硬件以减少数据处理和存储的成本。而到了20年世纪80年代，微电子技术的迅猛发展，导致了计算机硬件的性能价格比的日益提高——即计算机成本的日益降低，而功能却不断提高。

因而，现在面临的问题与计算机系统发展前30年所面临的问题不同了。现在主要的挑战是如何减少基于计算机的求解问题方案的成本，也就是如何减少用软件实现的求解问题方案的成本，以及如何改善其质量。

现在一块集成电路就能实现过去大型计算机所提供的功能。使人敬畏的现代硬件的处理和存储数据的能力代表着计算机系统的潜力，而软件却是使人类能够利用和开发这种潜力的机制。

1.1.2　软件发展历史

软件发展过程与计算机系统发展的40多年密切相关。硬件性能不断提高、体积日益缩小，以及成本日益下降，加速了基于计算机系统的日益复杂化，人类已经从应用真空管处理器时期进入到广泛应用微电子器件的新时期。托夫勒在《第三次浪潮》一书中，将微电子技术的出现作为人类历史上第三次变革浪潮的一部分。人们知道工业化社会向信息化社会的过渡将对人类生活产生巨大的影响。

在计算机系统发展的早期，硬件的变革连续不断地进行，而软件却被人们忽视了。计算机程序设计缺少系统化的方法，程序设计曾是一种缺少根基的随意创作艺术。实际上，过去软件的开发是无管理的，从而曾经常常遇到下述情况，要么推迟进度计划，要么增加成本，在这个时期，许多系统都是采用批处理方式工作的。当然也有例外情况，如美国联合航空公司使用的订票系，以及象SAGE这种用于国防的实时系统等就是早期的交互式系统。尽管如此，其它大部分系统却是仅用于执行某个程序，也就是说仅用于某种单一的专门应用。

在这个时期里，已有不少通用化硬件。而软件却是各自为专项应用而设计的，同时分布性也很少。产品化软件（即：用于销售给一个或多个用户而开发的软件）正处于发展的初级阶段。大部分软件是由使用者自己开发。程序的开发者，同时又是程序的使用者，在使用过

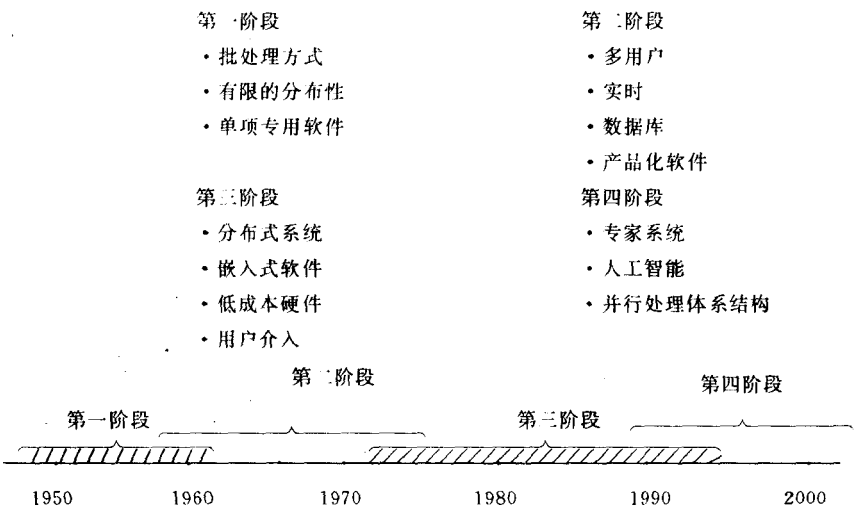

图1.1.1　软件技术发展过程

程中，如果发现程序有问题，则又自行加以修改。当时工作的迁移性比较低，当软件出现问题时，程序的开发者往往还在本部门，主管们对此是比较有把握的。在这种个人自行开发软件模式的情况下，软件的设计过程是在开发者个人头脑中默默地完成的，自然常常是没有设计文档。

通过这个时期，人们获得了许多开发基于计算机系统的知识，但对计算机系统工程的认识却很少。公正地说，无论如何，人们应该感激这个时期开发出来的许多优秀的系统。而其中的部分系统至今仍在工作着，并且取得了里程碑式的成就，对于现在来说，仍然是值得称颂的。

计算机系统发展的第二个阶段始于60年代中期，一直延续到70年代后期。在这个阶段，出现了多进程、多用户计算机系统，并引入了有关人机交互的新概念，并且人机交互技术开创了应用新天地。同时硬件和软件的复杂程度也提高了。实时系统可以从多处收集数据，并对其进行分析和处理。因而控制决策的制定是以毫秒为单位的，而不是用分来计量的。联机数据存储设备的发展导致了第一代数据库管理系统的产生。

计算机系统发展的第二个阶段的另一主要特征便是已经开始使用产品化软件，以及相关的软件开发公司的出现，这个时期，软件开发公司开发出的软件在变化莫测的软件市场上进行大范围销售。用于大型和小型机的某一程序，可以有数以百计或数以千计的用户。来自工业部门和研究机构的一些穷困潦倒的技术人员，转而去从事产品化软件的开发，不久这些创业者便令人吃惊地挣了数目可观的钱。

随着计算机应用系统数量的增加，计算机应用软件库也开始膨胀。软件商开发的各种软件产品往往都有数以万计的程序编程语句，而且用户在购买来的产品化软件上往往还要增加成千上万条新的程序编程语句，于是计算机世界的上空便出现了一团乌云，每当发现软件有问题时，就不得不对组成软件的数以万计的程序编程语句进行审阅并修改；同样当用户的需求出现变化时，也不得不对这令人头痛的大量编程语句进行审阅修改；有时为了应用新的性能价格比高的硬件，也要阅读数以万计的程序以完成调整软件的任务。所有上述这些修改软

件的工作合称为软件维护。人们发现软件维护工作以令人吃惊的速度消耗各种资源。更糟糕的是，许多程序上的个人化特征使得这些程序实际上是不可维护的，这样便出现了软件危机。

计算机应用系统发展的第三个阶段起始于70年代中期，一直持续到现在。这个时期出现了众多的分布式系统，即多计算机系统，分布式系统中的各计算机同时分别地完成各种功能，并且相互之间进行通信协调处理。分布式系统的出现使计算机应用系统的复杂性大大地增加了。全局和区域性网络，对大通道容量的数字通信，以及数据存取速度的要求日益强烈，从而给软件开发者带来了巨大的压力。

这个时期的另一个主要特征是，微处理器和个人计算机的出现和迅速推广。微处理器是各种智能产品的关键元素。这类智能产品有汽车、微波炉、工业机器人以及血液诊断设备等。在许多情况下，是由那些不太了解软件开发技术、仅熟悉硬件设计技术的技术人员，将软件集成到产品中去的。个人计算机是许多公司发展的催化剂。在计算机应用系统发展的第二个阶段，软件公司最多能将自己的产品软件销售几百份或几千份，而到了计算机应用系统发展的第三阶段，软件公司却可能将自己的产品软件销售几万份，甚至几十万份。个人计算机出现不久，便很快成为畅销产品，而适用于个人计算机的软件使其具有各种不同的特征。事实上，尽管在80年代中期，个人计算机销售增长速率趋于平稳，但用于个人计算机的软件产品的销售却一直在增长。个人和企业花在购买有关软件上的钱比购买计算机硬件系统的钱还要多。

计算机应用系统发展的第四个阶段起始于80年代的中后期。费根鲍姆和麦克柯达克等一些作者指出："第五代"计算机以及与之相关的软件将对世界范围内的政治和工业实力的平衡产生重大影响。目前，软件的第四代技术（4GT——the fourth Generation Techniques）已经对软件工作人员的编程方式产生了较大影响。而专家系统和人工智能软件最终将走出实验室进入实际应用中去，从而解决现实世界中的大量问题。当计算机应用系统的发展进入第四个时期时，软件危机进一步恶化，其主要特征是：

- 硬件的复杂性，已经超出人们开发相应软件的能力，从而使硬件的应用受到限制。
- 人们开发新程序的速度达不到要求建造新程序的期望速度。
- 人们维护现有程序的能力受到设计质量差，以及受到不合适的资源限制。

为了应付软件危机，软件工程的应用正被越来越多的软件工作人员接受。

1.1.3 软件工程背景

在计算机应用系统发展的早期，开发计算机应用系统，主要使用面向硬件的管理技术，开发项目的主管们主要考虑有关硬件开发的问题，因为在开发计算机应用系统时，硬件是投资最大的部分。为了控制硬件开发成本，项目主管们制定了许多规范条例和技术标准，要求在研制系统之前，必须进行详细的分析和设计，并对研制过程进行监控，以确定可加以改进的地方。简而言之，人们在研制硬件系统时，使用了各种有效的规章条例、方法和工具。这些行之有效的规章条例、方法和工具，便构成了所谓的硬件工程，而对大多数软件开发来说，只留有事后思考的余地。在这个时期，人们更多地将设计程序看成是一种随意创作的艺术形式。程序设计者们往往是通过尝试性的实践来训练自己的设计技能。实际上，软件开发没有规章条例可循，并且当时的许多软件开发者确实是乐于这样。

然而，现在开发计算机应用系统的费用分配发生了巨大变化。耗费开发费用最多的常常是软件，而不是硬件，在过去的10多年里，项目主管及许多技术人员经常提出下述问题：

• 为什么要花如此长时间才能完成软件开发？
• 为什么软件成本会如此高？
• 为什么在将软件交给用户之前不能发现所有错误？
• 为什么软件开发过程的进度难以估测？

这些，以及其它许多有关软件的问题，都表达了人们对软件及其开发方式的关心与思考。正是这种关心和思考产生了软件工程的实践并使其得到广泛接受。

1.2 软件基本概念

1.2.1 软件特征

在 20 多年以前，能够正确描述“计算机软件”这一概念的文献数量，不到有关文献总数的 1%。而现在许多人认为，大多数专业文献以及众多的书刊都能正确描述“计算机软件”这个概念。情况真是如此吗？

一些教科书中就可能如此描述软件：

(1) 是指令集（计算机程序），当它们运行时，就能提供所期望的功能和性能；

(2) 是使程序能够正确而又充分地处理有关信息的数据结构；

(3) 是描述程序使用和操作的有关文档。

毫无疑问，确实还可以给出一种更全面完整的软件定义，但是人们需要的难道仅仅是正式的软件定义吗？

为了能真正理解软件的这个概念，以及最终认识软件工程，分析一下软件的特征是很有用的，软件的这些特征是人们可以用来区分软件与其它人造物的主要因素。在开发硬件时，人们的创造性工作（分析、设计、构造、测试）最终被转变成有形的物理产品。如开发一种新型计算机硬件系统，人们所作的初始草图以及随后的正式设计图和原型系统，最后将转化为使用超大规模集成电路的各种印刷电路板和有关的电源、机壳等组成的有形物理产品。而软件则是一种逻辑性系统元素，不是物理性系统元素，因此，软件的特征与硬件的特征有很大的差别。

软件只能设计出来或实现，而不能用传统意义上的制造进行生产。尽管软件开发与硬件开发有某些类似之处，但两者之间有着根本性区别，虽然在这两种不同的过程中都存在产品的设计阶段，并且这一阶段对各自的最终产品质量都具有至关重要的作用，但是硬件开发过程中的硬件制造阶段也能产生有关质量问题。而软件开发过程却不存在这种问题（至少可以说软件实现过程中，因疏忽引起的错误是很容易予以纠正的）。尽管这两种活动都要依靠人才能完成，但是参加人员与完成的工作之间的关系，对这两种活动而言，却存在很大的差别。尽管这两种活动都要给出产品，但两者采取的生产方法是不同的。

软件开发成本主要集中于软件开发的工程管理上，这就是说软件项目的管理不能象管理硬件开发项目那样进行，在过去 5 年里，已有一些文献论述过“软件工厂”的概念，但这个概念并不是说软件开发和硬件开发是完全一样的，注意到这一点很重要。然而“软件工厂”的概念建议使用自动化工具用于软件开发，这对于提高软件开发生产率是很有必要的。

软件不会“磨损”，而硬件却会磨损。图 1.2.1 给出了硬件故障率随时间变化的曲线，这种曲线常被称为浴盆截面曲线，它表明，硬件在其生命周期的初始阶段存在相当高的故障率，

这时出现的故障，常常是由设计或者制造过程中出现的失误造成的，当这些研制过程中出现的错误被纠正后，硬件产品趋于定型，因而故障率便下降到一个较低的水平，并在一定时间里相对稳定不变，随着时间的推移，使用中的硬件的故障率开始上升，这往往是由于遭受灰尘的污损，以及震动、使用不当、极限温度等许多不良环境的累积效果引起的，简而言之，就是硬件出现了“磨损”现象。

软件对造成硬件磨损的不良环境并不敏感，因而，理论上的软件故障率曲线应象图 1.2.2 给出的那样。

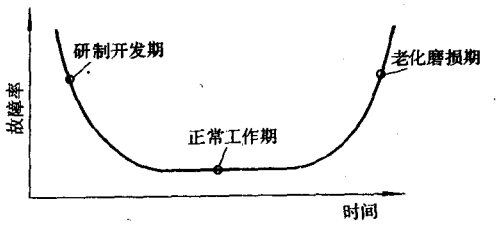

图 1.2.1　硬件故障率曲线

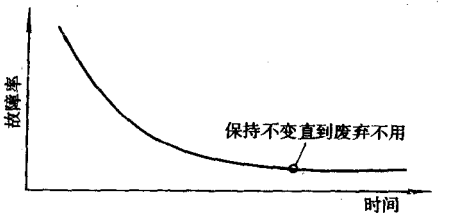

图 1.2.2　理想的软件故障率曲线

图 1.2.2 是软件实际故障率模型的大为简化的结果，但它的含义是很清楚的——软件不会磨损。尽管如此，软件却会退化。借助于图 1.2.3 给出的说明，便不难理解这看似矛盾的说法的真实性。在软件的生命周期中，其本身将会遇到修改（维护），此时便很可能引入新的缺陷，导致故障率突然上升，正如图 1.2.3 描绘的那样。随后，软件故障率应回落，并趋于原来的平稳故障率值，可是在回落到这个平稳故障率值之前，新的修改要求又出现了，于是软件故障率又突然上长，如此这般循环往复地波动着，从而导致软件故障率的最小值随着时间的增长而缓慢地增加，可以说，由于修改，软件正在退化。

硬件和软件还存在下述区别：当某一硬件元素磨损后，可以用相同的备用元素更换，从而使硬件系统恢复正常工作。而软件却没有相同的备用元素可言，因为软件出现的每一个故障，要么是由于设计考虑不周造成的，要么是将设计转变为机器可执行命令集的过程中存在失误。由于软件无备用元素可供更换，因而软件维护比硬件维护要复杂得多。

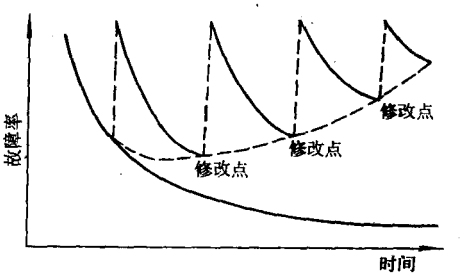

图 1.2.3　实际软件故障率曲线

大多数软件是专项独立开发的，不是由现有的什么组件装配而成的。让我们来看看基于微处理器的产品中使用的控制线路板是怎样设计和制造出来的，首先硬件设计工程师画出简单的数字电路图，接着对其进行基本原理性分析以保证能得到正确的功能，然后便查阅数字电路元器件目录，在这上面，列有每种集成电路块的产品号、通过检验的功能说明，以及有关的接口定义和集成标准，当挑选好所需元器件的类型之后，便可以从有关的供应商那里将它们按所需数量采购回来。可悲的是，软件设

计者不能获得象集成电路块那样的元器件用于实现软件，没有软件组件目录这种说法，目前还不存在例外情况，尽管可以买到产品化软件，但它往往是作为一个整体进行销售，也就是说，普通的产品化软件不能当作组件，用来装配生产出一个新的程序。虽然已有许多论述“软件可重用性”的文章，但至今取得的实质性成果仍很少。

1.2.2 软件构成

计算机软件是一种信息，它以两种基本形式存在：计算机不可运行形式和计算机可执行形式。图 1.2.4 描述了软件是如何由计算机不可运行形式转变成计算机可执行形式的过程。

软件的设计将以一种计算机可以理解的语言来实现，用这种语言描述软件的数据结构、各种处理过程以及有关的要求。继而这种语言形式通过翻译程序处理变换成机器可执行的指令序列集。

最理想的是，人们可以用自然语言（如英语、西班牙语、汉语等）与计算机进行交流。不幸的是，自然语言有着大量丰富的词汇，而且它的词法结构复杂，用于理解语言意义的文法结构也很复杂，正是这些特点使得用自然语言进行人机交互变得十分困难。虽然有关语义信息处理和自然语言识别的一些研究已经为用自然语言进行人机交互奠定了基础，但是至少在最近几年，程序编程语言仍将局限于人工语言，或形式语言。

目前使用的所有编程语言都是人工语言，它们都有有限的词汇，并且它们的语法清楚简单，它们的语义和句法规则是结构化的。这些特征是机器翻译的基础。软件所拥有的语言形式通常划分为：机器语言、高级语言和非过程语言等。

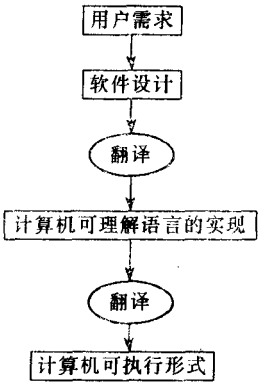

图 1.2.4 计算机软件变换过程

当一个经验丰富的程序设计师将某一程序设计得具有很好的可维护性，而且文档清楚、完全时，那么使用机器语言来实现这样一个程序，就可以最有效地利用存储空间，并使程序执行速度最佳。但如果程序设计得不好，文档也残缺不全，那么，将其翻译成机器语言之后就会使问题变得更加严重。

尽管用机器码语言编程可能获得吸引人的执行速度，并且能够充分利用存储单元，但它存在以下较为严重的缺陷。

（1）开发时间将会由于语言编写的复杂性，以及描述信息的抽象层次太低而大大延长。

（2）编写出的源程序可读性差。

（3）测试困难。

（4）维护十分困难。

（5）不同处理器之间的移植难以实现。

与此相比，使用高级语言编写的源程序可以不依赖于具体的机器，并且，翻译程序越先进，相应的语言的词汇、语法、句法以及语义就比机器码语言更好。实际上，高级语言的编译程序或解释程序产生的输出就是机器码语言。

目前已在使用的编程语言虽有数百种，但广泛使用的高级语言不到 10 种。如 COBOL 和 FORTRAN 语言从推出以来，就一直被广泛应用；能够支持数据和处理过程现代设计实践的现代语言，象 PASCAL、C 和 Ada 等语言也正在得到广泛应用。为某一专门领域设计的特殊语言，象 Smalltalk、APL、LISP 和 Prolog 等语言，作为从实验室走向实际应用不久的、新

的应用技术正越来越得到广泛的接受。

前面已间接论述了编程语言翻译程序的功能，它的功能是将用编程语言编写的源程序转换成相应的机器可执行的指令序列。如汇编程序，它是一种低级编程语言的翻译程序，它仅完成将用机器指令助记符编写的程序转换成机器可执行的指令序列这样比较简单的工作。解释程序是一种基于逐句翻译、逐句执行原则的高级编程语言翻译程序。然而最普遍的高级编程语言翻译程序是编译器（compiler）。通过对整个源程序进行分析、评估，编译器能够优化其输出的机器码程序的存储空间大小，或/和执行速度。

机器可执行的指令码序列是软件构成中的最底层。一般用十六进制或者其它特殊代码描述某 CPU 执行的处理过程的二进制位码指令序列。

机器指令码、汇编语言和一般的高级编程语言，常常被称为计算机编程语言的最初三代语言。使用这些语言编程时，程序设计者必然既要考虑信息结构的数据描述，又要考虑程序处理过程的控制结构。因而，最初三代编程语言都被称为过程语言。

在最近 10 年里，一些第四代编程语言，或称非过程语言已被引入应用。这些非过程语言只要求软件设计者描述的期望的结果，而不用描述如何去获得所期望的结果，也就是说软件设计者不必去描述有关的详细处理过程。非过程语言的支撑软件将所描述的期望结果变换成机器可执行的相应处理程序。目前，第四代语言已经在数据库应用领域以及其它经营管理数据处理领域中得到了应用。

1.2.3 软件应用领域种类

只要处理过程（或算法）已经描述清楚，就可以用程序将其实现，而信息的内容和确定程度是决定软件应用特征的重要因素。其中信息的内容是指软件输入、输出信息的含义及其格式。例如，许多经营管理应用软件的输入是高度结构化的数据，其输出则往往是格式很好的报表。而自动控制机器（如数字控制系统）使用的软件，其输入的是离散型数据，并且数据的结构化程度不高，这种软件 的输出往往是机器运行的单个命令码。

信息的确定性则是指软件输入数据的输入时间和秩序的可预测性。如对工程分析程序而言，它接受的数据有预先定义好的秩序，它在执行中不会被中断，并且产生的结果通常以报表或图形方式给出，这样的应用信息是确定性的。而另一种情况则不同，象多用户操作系统软件，它接受的输入内容是变化的，并且输入时间也是任意的，运行过程中可由外部条件中断，产生的输出也将随时间和环境变化而变化，具有这些特征的应用信息是非确定性的。

软件种类的含义明确的严格划分较困难。随着软件复杂性的增加，简洁的种类划分已经消失。下面给出各种软件应用领域的一些潜在的应用前景。

• 系统软件

系统软件是给其它程序提供服务的程序集合。一般来说，有些系统软件（如：编译程序、编辑程序，以及用于文件管理的实用工具）处理过程复杂，但其处理的信息比较确定，而且信息结构化程度较高，而有些系统软件（如操作系统中的一些基本程序、有关设备的驱动程序、通信处理程序）处理的信息，在很大程度上是不确定的。但这两类系统软件存在下述共同的特征：与计算机硬件系统有很强的交互性，并且要对共享资源进行调度管理，因而系统软件具有复杂的过程管理程序以解决并发性操作处理中存在的协调问题。而且系统软件中的数据结构复杂，它的外部接口多样化。将有许多用户反复使用它。

• 实时软件

对现实世界中随时发生的事件进行监测、分析和控制使用的软件称为实时软件。实时软件一般有用于收集数据的基本模块，其作用是从外部环境中收集信息，并将其结构化。实时软件中的分析模块将把收集来的信息变换成相应的应用需要的信息。实时软件中的控制、输出模块将根据分析模块提供的信息对外部环境作出正确的响应。实时软件中的监控模块将协调上述基本模块的运行，从而保证实时响应性（一般要求的范围是在 1ms 到 1min 之间）。应该注意区分“实时”、“交互”、“分时”之间的不同。实时系统常常存在严格的响应速度限制、交互式系统的响应时间的延长，一般来说不会导致灾难性的结果。

• 管理应用软件

管理信息处理是最大的软件应用领域。管理信息系统应用软件，往往将集成各种分离的应用软件，如将工资管理，财务应收、应付管理和库存管理等分离型应用软件集成起来以支持企业的各项管理工作。管理信息系统应用软件将使用一个或多个大型数据库用来存放有关管理信息，并对数据库中存放的结构化数据进行再一次的结构化处理，以辅助经营管理和决策制定。除了完成传统的数据处理任务外，管理应用软件应提供交互式处理功能（如对销售单据的处理就必须以交互方式进行）。

• 工程分析和科学计算软件

工程分析和科学计算软件是以对数据进行按部就班地处理为特征的，它的应用范围十分广泛，从天文学研究到火山研究、从自动应力分析到空间运行轨道动力学研究，从分子生物学研究到自动生产过程研究等都能看到它的存在。但是新的工程分析和科学计算领域中的应用软件已脱离传统的数字运算概念。象计算机辅助设计、系统仿真，以及其它的交互式应用软件，已经具有实时，甚至是系统软件的一些特征。

• 嵌入式应用软件

智能产品在几乎所有的家用市场和工业市场上处处可见。嵌入式软件一般常储存在只读存储器中，用来控制家用或工业市场上的各种产品或系统。一般来说，嵌入式软件仅完成一些非常有限但又比较奥妙的功能（如微波炉上的操作键的监控）；或者提供一些重要的功能和控制能力（如汽车中使用的各种数字显示功能、油门控制以及制动系统等）。

• 个人计算机应用软件

个人计算机应用软件市场在过去 10 年内发展异常迅速。它有许多优秀的代表：象文字处理、电子算表、计算机图形、娱乐游戏、数据库管理、个人和企业财务管理等系统，还有网络支持软件等。实际上，个人计算机应用软件仍然是软件应用领域中一个最富有革新意识的软件设计代表。

• 人工智能应用软件

人工智能（AI）应用软件利用非数字算法解决复杂问题，这些问题往往不是单纯计算和直觉分析所能解决的。当前，人工智能领域最活跃的是专家系统，也称为基于知识的问题求解系统。人工智能应用软件还有一些范例，如模式识别（图象和声音）、理论证明以及博奕等。

1.3 软件危机

软件危机指的是计算机软件开发中遇到的一些困难问题，这些问题并不局限于软件功能