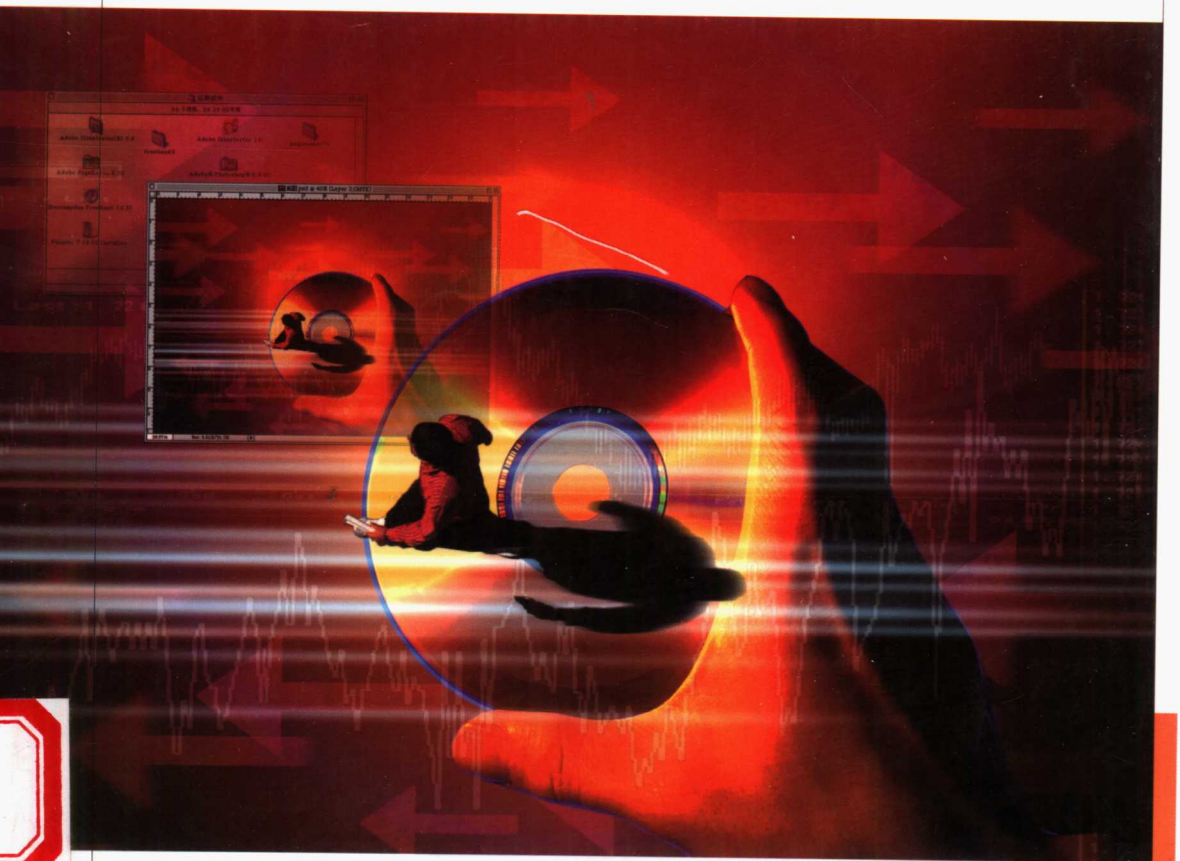
高等工程教育人才培养创新教材出版工程

# 软件工程规范设计

张凯 编著



科学出版社
www.sciencep.com

●高等工程教育人才培养创新教材出版工程

# 软件工程规范设计

张 凯 编著

科学出版社

北 京

## 内 容 简 介

本书阐述了瀑布式模型从上流需求到软件设计过程中的四项主要技术，即需求分析、设计法、文档和设计审查，同时还介绍了软件工程及软件开发工具和环境的相关内容。在“需求分析”的章节中，阐述了需求分析的要点，需求分析的工作过程以及需求分析的有效方法和工具等内容。在“设计法”中，在以时间的顺序概述各个方法的基础上，分析和整理出两大类设计法。在“文档”中，因对象不同，将文档分为以下 3 种：面向顾客的文档、面向软件开发人员的文档、面向软件使用和维护人员的文档。“设计审查”陈述了它的内容和实施方法及文档等。

本书体现了工程中的成熟技术和案例，是一本在软件工程领域中理论联系实际且有很强的工程使用价值的参考书。本书适用于相关专业的本科生、软件工程硕士研究生、软件企业培训学员等。

**图书在版编目(CIP)数据**

软件工程规范设计/张凯 编著. —北京：科学出版社，2005.1
（高等工程教育人才培养创新教材出版工程/高林，张凯 主编）
ISBN 7-03-014535-6

Ⅰ.软… Ⅱ.张… Ⅲ.软件工程-高等学校-教材 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2004)第 111325 号

责任编辑：许 远 余 丁 / 责任校对：刘小梅
责任印制：安春生 / 封面设计：王壮波

科学出版社出版
北京东黄城根北街16号
邮政编码 100717
http://www.sciencep.com
双青印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2005 年 1 月第 一 版 开本：B5(720×1000)
2005 年 1 月第一次印刷 印张：20 1/4 插页：1
印数：1—3 000 字数：396 000

**定价：30.00 元**

（如有印装质量问题，我社负责调换〈环伟〉）

# 序

日本的软件工程无论是在预算上还是在工期上，其设计工程、制造工程、测试工程之比为2：1：2，尤其是在文档资料的编写上需花费相当大的代价。因为文档除了能起到向下一步工程传达设计意图和设计内容的作用以外，还有以下三点重要作用：

（1）用户方和开发方的责任依据。

（2）大规模团队开发的信息依据。

（3）软件维护和再开发的设计依据。

正因为如此，日本软件中设计工程和测试工程的人均单价为制造工程人均单价的1.45倍以上，自然也构成了日本软件业的价值观。

与日本软件业相比，我国软件业中普遍存在轻视设计工程和测试工程而重视制造工程的倾向。虽然近十年来，我国很重视培养计算机软件开发人才，但是90%都是位于制造工程层次的人才。这种人才结构，首先决定了我国独自研发的软件产品市场成功率低，市场寿命短的现状。解决我国软件开发人才结构问题的关键在于高等院校的计算机软件专业的教学改革和人才结构的调整以及IT企业人才教育类型的改革。

随着计算机应用的普及和发展，软件开发呈现出国际化的特征。最近几年，我国的软件出口增长较快，特别是沿海地区对日本的软件出口更加显著，其中绝大部分属于承接来自日本的软件外包业务，许多软件企业在这种势态下不断发展壮大。因此，外包型软件企业迫切需要既掌握软件开发技术又懂日语的人才。

张凯主编的《软件工程规范设计》一书，比较详细地介绍了在日本实际应用的软件工程方法，以结构为主线分别阐述了系统分析、设计、编码和测试等内容。内容紧贴工程实际应用，体现了工程中的成熟技术和工程案例，是一本在软件工程领域能够理论联系实际，有一定理论水平和工程使用价值的好书。这种内容及其组织方法在我国已出版的软件工程专著和教材中还不多见。因此该书的出版可以为我们树立应用型著作的典范。

2004年10月

# 前　言

本书阐述了瀑布式模型从上流需求到软件设计过程中的四项主要技术，即需求分析、设计方法、文档和设计审查，同时还介绍了软件工程及软件开发工具和环境的相关内容。本书的目的并不是追求软件工程理论的前沿，而是一本为培养能胜任软件开发和应用工作的各层次实用型、复合型人才而编写的应用性教材。

软件工程已经由当初仅为少数拥护者所实践的蒙胧思想演化成一门正式的工程学科。现在，它已被认为是一个值得认真研究、细心学习和激烈争论的主题。在整个行业中，“软件工程师”已经替代“程序员”成为更受欢迎的头衔。软件过程模型、软件工程方法以及软件工具已经在行业应用中被成功地广泛采用。

1. 关于需求分析

需求分析的作用是，明确用户要求，获得系统和软件外部特性的描述。在需求分析的章节中，阐述了需求分析的要点，需求分析的工作过程以及需求分析的有效方法和工具等内容。

其中需求分析中容易出现的问题有：

(1) 用户问题认识的模糊性——用户常常对问题领域的认识是模糊和抽象的，有时还会存在一些对情景的想像和片面的认识。在需求分析中，经常会把这样连用户自己都没有分析和整理清楚的问题传达给系统开发人员。

(2) 用户与开发人员的交流间隙——需求分析员一般是系统开发部门的技术人员，他未必掌握与用户问题领域相关的专业知识，同时用户通常也不了解有关计算机系统和软件开发的专门技术，这就造成两者之间交流间隙过大和沟通的困难。

(3) 作为创造活动本身的困难——这是需求分析阶段，真正的问题是使目的明确，提出解决策略，而人们知识的创造活动正是需求分析的基础。

(4) 预测需求变化的困难——需要有 2～3 年开发周期的大规模系统，即使在需求分析过程中定义了完全的设计书，也必须做好在设计和制造过程中追加和更改设计的精神准备。

(5) 设计描述和验证的困难——即使明确了用户的需求，但把它作为需求设计完整地记述出来也是很困难的。通常，需求设计书是从开发人员的立场出发，以完整性为优先。由于记述常常让用户难以理解，因此，从用户来看，需求妥当性的验证是很困难的。

(6) 对于需求分析的认识不足——没有人否定需求分析的重要性，可是在实际开发项目时，往往不能足够地投入时间和人力。

(7) 面向需求分析的实际方法论的欠缺——对于需求分析，虽然提出了许多方法和工具，但前面所说的“多”常常是在研究开发阶段，并限定了适用范围。在实际业务程度上的应用方法和工具则是不够的。

在设计法中，按时间的顺序概述各个方法的基础，整理出如下两大类。

——第一类：模块分割设计法

①系统阶层分割设计法；②top-down 设计法；③数据抽象设计法；④结构化设计法。

——第二类：模块设计法

①控制结构主导设计法；②数据结构主导设计法。

对选择模块分割设计法中的结构化设计法及数据结构主导设计法进行详细讲解。对于其余方法，只限于讲解主要特点。

作为设计法的说明，借用被情报处理学会经常使用的“酒类贩卖库存管理”的共通问题，先对问题的理解进行解释，以结构化设计法（复合设计法）和数据结构主导设计法（Warnier 法，Jackson 法）为基准，进行具体的设计。

作为这些设计法的基础，也介绍了被软件工程界广泛应用的“结构化”和“抽象化”的概念。

2. 关于软件文档

为了解决上述问题，努力使需求分析成为能够结合具体事例来讲解实际中需求分析的方法与工具，在文档中，因对象不同，将文档分为以下三种：

——面向顾客的文档。

——面向软件开发人员的文档。

——面向软件使用和维护人员的文档。

讲述了文档种类，各种文档的工作和记述内容，文档的质量管理以及最新的文档支持工具等。

在本书解说的文档体系如下所述：

(1) 面向顾客的文档

①系统开发计划书；②外部设计书。

(2) 面向软件开发人员的文档

①内部设计书；②开发管理-工作顺序书；③记录文档。

(3) 面向软件使用和维护人员的文档

①保守书；②使用日志；③定期检查记录。

特别地，指出了对于外部设计书，为了让销售的软件能够作为广泛使用的商

品，需要迫切地提高质量。作为具体的例子，IBM 公司的生产手册就非常优秀。关于以上理由，从参考文献中归纳出以下要点：

——为了使用而写
——限制使用的词汇
——考虑到易读性和可理解度
——明确使用用语的定义
——完善索引
——具有工作指向性
——写成重点让对方注意
——标明与关联手册的关系
——做成内容可变更的系统
——积极地接受使用者的评论
——以使用者的观点测试
——例子的正确性且是易理解的
——与体裁相比，正确性是第一位的
——遵照公司内部出版物的标准规范

3. 关于设计审查

设计审查是一个确保软件生产品质的有效手段。“设计审查”陈述了它的内容和实施方法及文档等。

在这里使用的设计审查中，错误被认为有以下三类：

(1) 上一个工程的错误原封不动地被引用到下一个工程。

(2) 当前工程的工作诱发出其他工程的错误。

(3) 当前工程产生新的错误。

如不及时实施调查，将会被引入下一个工程的错误个数如下：需求分析 6 个，功能设计 15 个，理论设计 55 个，制造工程 145 个，测试工程 18 个。

在需求分析、功能设计、逻辑设计、制造等各工程中实施审查，错误的找出率分别占 50%，50%，60%，70%。对于错误的产生系统，如果与不实施审查留下 18 个错误的情况相比较，实施审查后最终只在系统中留下 2 个错误。如果用定量计算来表示审查的效果，其费用只占未加审查的 1/6。并且，在实际进行审查时，组织层的体制也非常重要，以下说明了必要的成员构成。

(1) 设计审查的主管、组织者——项目总负责人、项目经理、组长。

(2) 相关资料的准备等——该工程的设计负责人。

(3) 审查参加人员——前续工程设计负责人、后续工程负责人、了解系统的人员、硬件负责人、销售负责人、用户代表（含使用人员）、质量检查部门负责

人。

（4）筹划时间表——干事。

（5）审查实施的管理——审查管理部门。

最后，通过软件开发工具和软件开发环境的利用，能够很好地维护软件设计过程中的各个开发阶段，在极大程度上方便了软件开发人员的开发工作。软件开发环境包含有一组软件工具，这些工具是按照一定的方法或模型组织起来的，这些工具支撑整个软件生存周期的各个阶段或部分阶段。其宗旨是为计算机软件的生产提供计算机辅助手段，改变长期以来软件产品的手工式生产方式，以提高软件产品的生产率，降低软件的成本和改善软件的质量。

在软件生产过程中上流工程的规格化和设计是当前工具支援最少的部分，即可以说是以人的脑力劳动为主体。最近，那些脑力劳动的成果和取得的技术被作为知识基础概括出来，变成供人们可以利用的多样的试用方法。但是，要达到实用的水平尚需时间。

因此，目前在把上述的技术作为自己的工具完全掌握的同时，还要通过实践总结出经验，然后使之发展成为接近实际的有用的方法。本书如能对读者有所帮助，作者将备感荣幸。

此外，清华大学谭浩强教授、北京高等学校计算机基础教育研究会理事长高林教授、日本凯恩克思株式会社总裁金胜光、北京富士通系统工程有限公司高级工程师林泽长和田开亮、日本电气 NEC 信息系统有限公司总裁青木祐一郎、中国教育电子公司总经理宋志德、北京航空航天大学软件学院孙伟院长和姚淑珍副院长、北京大凯凯达电脑技术有限责任公司高级工程师张佶和周裕明对本书进行了全面的评审并作出了高度评价，在此深表感谢。

编　者

# 目　录

# 第一章　软件与软件工程

## 1.1　软件的发展

人们常说的计算机系统是由硬件和软件两大部分组成的。用计算机求解问题，程序是不可缺少的，程序被称为软件的实体部分，程序只有在硬件载体上运行才可获得所求问题的解，因而硬件和程序是求解问题的最基本条件。然而，仅有程序也会给使用者带来诸多不便，好的程序应有相应的文字资料，如各种规格说明书、设计说明书、用户手册等。我们称这些文字资料为文档。文档不仅对使用者是必要的，而且对程序开发者更是至关重要的。特别是由多个人经过多年才能完成开发任务的大型程序，人与人之间、开发者与使用者之间都需要有规范的书面文档规定程序的功能、使用环境、使用方法等。所以严格地说，程序和软件是两个不同的概念。程序是指计算机可识别的源程序代码或机器可直接执行的程序代码。而软件是指程序加上开发、使用和维护该程序所需要的全部文档。程序是软件的实体，而文档是软件的重要组成部分。这个公认的浅显认识囊括了软件发展的三个阶段。

软件的发展大致可分为三个时期：程序时期（1947 年至 20 世纪 60 年代初），软件＝程序＋说明时期（20 世纪 50 年代末～70 年代初），软件＝程序＋文档时期（20 世纪 70 年代初至今，也称软件工程时期），见表 1.1。

我们这里所说的程序是指在计算机上可执行的代码序列。在程序时期，使用者需直接操作计算机硬件，因而一开始就形成了计算机硬件就是计算机这一概念。这一时期，计算机工作者致力于改善硬件结构和可靠性研究，而仅把程序作为机器运行时必须进行的准备工作。每个程序都是为求解某个特定问题而专门设计的，不考虑程序的通用性，更没意识到程序同计算机硬件一样，也是计算机系统不可分割的部分。这一时期的程序设计工作全凭设计者个人经验和技艺独立进行，是一种典型的手工艺智力劳动。程序中出现的错误只能由设计者本人才能修改，因而也无需向他人做任何交待和说明，当时人们脑中只有程序概念而无软件概念。

20 世纪 50 年代末至 70 年代初这段时期，由于硬件技术的飞速发展，与此同时相继问世了一批高级程序设计语言（如 FORTRAN、ALGOL60、COBOL、BASIC 等）的编译程序、解释程序，以及操作系统等支持程序（也称系统程序）

**表 1.1 计算机软件发展的三个时期及其特点**

| 时间<br>特点 | 程序设计<br>20 世纪 50～60 年代 | 程序系统<br>20 世纪 60～70 年代 | 软件工程<br>20 世纪 70 年代以后 |
|---|---|---|---|
| 软件所指 | 程序 | 程序及说明书 | 程序、文档、数据 |
| 主要程序设计语言 | 汇编机器语言 | 高级语言 | 软件语言 |
| 软件工作范围 | 程序编写 | 包括设计和测试 | 软件生存期 |
| 需求者 | 程序设计者本人 | 少数用户 | 市场用户 |
| 开发软件的组织 | 个人 | 开发小组 | 开发小组及大中型软件开发机构 |
| 软件规模 | 小型 | 中小型 | 大中小型 |
| 决定质量的因素 | 个人程序技术 | 小组技术水平 | 管理水平 |
| 开发技术和手段 | 子程序程序库 | 结构化程序设计 | 数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络及分布式开发、面向对象技术 |
| 维护责任者 | 程序设计者 | 开发小组 | 专职维护者 |
| 硬件特征 | 价格高存储容量小<br>工作可靠性差 | 降价、速度、容量及工作可靠性有明显提高 | 向超高速、大容量、微型化及网络化方向发展 |
| 软件特征 | 完全不受重视 | 软件技术的发展不能满足需要，出现软件危机 | 开发技术有进步，但未获突破性进展，价高，未完全摆脱软件危机 |

和具有较强通用性的应用程序，使得计算机应用从单一的科学计算，迅速发展到数据处理和实时控制等各个应用领域。为使用户方便地使用这些支持程序和通用应用程序，必须提供相应的技术指南、用户手册等说明性文字资料。因而出现了“软件＝程序＋说明”这一概念。这个时期的特征主要表现在三个方面：

① 由于程序规模较大，需多人协作才能完成程序编制，我们把这种方式称为“作坊式”生产方式。

② 程序的设计与运行维护再也不能由一个人来承担。

③ 人们已认识到程序再也不是计算机硬件的附属成分，而是计算机系统中与硬件相互依存的不可缺少的部分。

这个时期软件技术取得了很大进展，比较有代表性的是多道程序设计技术、多用户人机交互系统、实时系统以及文件管理等。同时出现了更多的通用和专用程序设计语言，在形式语言理论、编译理论、数据库理论等方面也取得了重大突破，从而诞生了真正的计算机科学。

随着投入市场运行的计算机数量日益巨增，计算机应用领域愈来愈广，软件需求量愈来愈大。因而，美国和西欧一些国家相继建立起庞大的软件公司，专门生产计算机软件，一种新兴产业——软件产业应运而生。

虽然软件需求量不断增大，其复杂度也越来越高，但其生产方式大多停留在

手工作坊式生产阶段。这种方式不仅效率低，而且软件质量差，如 IBM 公司 20 世纪 60 年代开发的 OS/360 系统，耗资几千万美元，花费了5 000多人/年，拖延几年才交付使用，交付使用后却漏洞百出。OS/360 系统开发负责人 Brooks 生动地描述了研制过程中的困难和混乱："……像巨兽陷入泥潭作垂死挣扎，挣扎得越猛，泥浆就沾得越多，最后没有一个野兽能逃脱淹没在泥潭中的命运……。程序设计就像是这样的泥潭，一批批程序员在泥潭中挣扎……。没有料到问题会这样棘手……"。比 OS/360 更糟的软件系统并不少，即花费大量的人力财力，结果半途而废，或完成之日即是遗弃之时。这就是人们常说的"软件危机"期。

为了摆脱这一困境，软件工作者一方面研究程序设计方法和程序正确性验证的方法；另一方面寻找工程化的软件系统开发方法。

以 E. W. Dijkstra 提出的"结构化程序设计方法"为代表，引导人们对程序设计方法的广泛研究。在短短几年时间内就提出了模块化、结构化、由顶向下逐步求精、程序变换、程序的推理与综合、数据类型抽象、符号测试、程序正确性证明等各种程序设计和验证的方法。虽然有些方法至今仍作为理论上的探讨，但这一时期的研究成果使软件设计者深刻认识到，没有科学的方法作指导，不可能生产出高质量的软件产品，同时还提出了以正确性、结构清晰、便于设计、便于测试和维护作为衡量软件质量优劣的重要标准。

## 1.2　软件的定义

计算机软件已经成为一种驱动力。它是进行商业决策的引擎，它是现代科学研究和工程问题寻求解答的基础，它也是鉴别现代产品和服务的关键因素。它被嵌入在各种类型的系统中：交通、医疗、电信、军事、工业生产过程、娱乐、办公等，难以穷举。软件在现代社会中确实是必不可少的。而且我们进入 21 世纪，软件将成为从基础教育到基因工程的所有领域新进展的驱动器。

### 1.2.1　软件的含义

软件一词具有三层含义。一为个体含义，即指计算机系统中的程序及其文档；二为整体含义，即指在特定计算机系统中所有上述个体含以下的软件的总称，亦即计算机系统中硬件除外的所有成分；三为学科含义，即指在研究、开发、维护以及使用前述含义下的软件所涉及的理论、方法、技术所构成的学科。

1. 软件的特点

要理解软件（以及最终理解软件工程），先了解软件的特征是很重要的，据此你能够明白软件与人类建造的其他事物之间的区别。当建造硬件时，人的创造

性的过程（分析、设计、建造、测试）最终被转换成有形的形式。如果我们建造一个新的计算机，初始的草图、正式的设计图纸和试验版的原型一步步演化成为一个有形的产品（芯片、线路板、电源等等）。

而软件是逻辑的而不是有形的系统原件。因此，软件具有与硬件完全不同的特征。

软件是被开发或设计的，而不是传统意义上被制造的。虽然在软件开发和硬件制造之间有一些相似之处，但两类活动在本质上是不同的。对于这两种活动，都可以通过良好的设计得到高质量，但硬件在制造过程中可能会引入质量问题，这种情况对于软件而言几乎不存在（或是很容易改正）；两者都依赖于人，但参与的人和完成的工作之间的关系不同；两种活动都要建造一个“产品”，但方法不同。

软件成本集中于开发上，这意味着软件项目不能像制造项目那样管理。

2. 软件不会“磨损”

图 1.1 描述了作为时间的函数的硬件故障率。其关系，常常被称作“浴缸曲线”，表明了硬件在其生命初期有较高的故障率（这些故障主要是由于设计或制造的缺陷）。这些缺陷修正之后，故障率在一段时间中会降到一个稳定的水平上（理想情况下，相当低）。然而，随着时间的流逝，故障率又提升了，这是因为硬件构件由于种种原因会不断受到损害，例如灰尘、振动、不合理的使用、温度的急剧变化以及其他许多环境问题。简单讲，硬件已经开始磨损了。

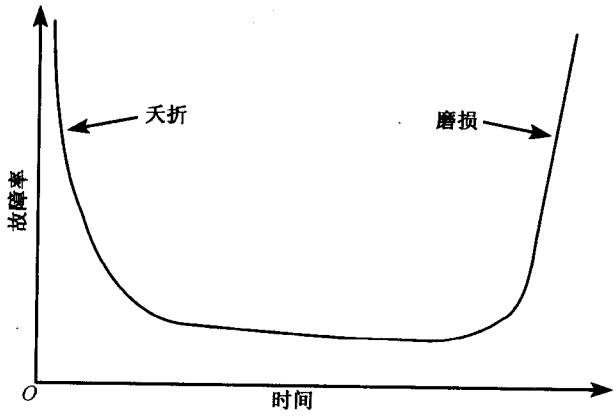



图 1.1 硬件的故障曲线

软件并不受到这些引起硬件磨损的环境因素的影响。因此，理论上，软件的故障率曲线呈现出如图 1.2 所示的“理想曲线”形式。未发现的错误会引起程序在其生命初期具有较高的故障率，然而，当这些错误改正之后（理想情况下，不

引入其他错误），曲线就如图 1.2 所示那样趋于平稳。理想曲线是软件的实际故障模型非常粗略的简化，但其含义很清楚——软件不会磨损，不过它会退化。

这个说法表面上似乎是矛盾的，我们可以通过图 1.2 中的实际曲线来解释清楚。在其生存期中，软件会经历修改（维护），随着这些修改，有可能会引入新的错误，使得故障率曲线呈现为图 1.2 所示的锯齿形。在该曲线能够恢复到原来的稳定状态的故障率之前，又需要新的修改，又引起一个新的锯齿。慢慢地，最小故障率水平就开始提高了——软件的退化是由于修改。

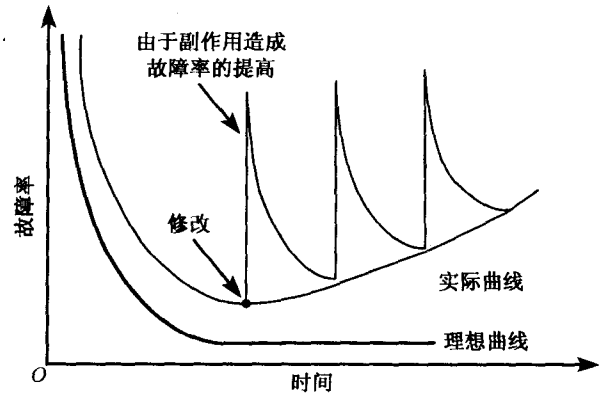



图 1.2　软件的理想故障曲线和实际故障曲线

关于磨损的另一方面也表明了硬件和软件之间的不同。当一个硬件构件磨损时，可以用另外一个备用零件替换它，但对于软件就没有备用零件可以替换了。每一个软件故障都表明了设计或是将设计转换成机器可执行代码的过程中存在错误。因此，软件维护要比硬件维护复杂得多。

3. 虽然软件产业正在向基于构件的组装前进，大多数软件仍是定制的

考虑一个基于计算机的产品的控制硬件被设计和建造的方式。设计工程师画一个简单的数字电路图，做一些基本的分析以保证可以实现预定的功能，然后查询所需的数字零件的目录。每一个集成电路（通常称为“IC”或“芯片”）都有一个零件编号、一个定义的和确认过的功能、定义好的接口和一组标准的集成指南。每一个选定的零件，你都可以在货架上买到。

随着一个工程化学科的演化，一组标准设计构件被创建。标准的螺丝钉和购买的集成电路仅仅是机械和电气工程师设计新系统时所使用的数千种标准构件中的两种。可复用的构件被创建，使得工程师可以专注于设计的真正创新的部分，即设计中表示某种新东西的部分。在硬件世界，构件复用是工程过程的自然的一部分。在软件世界，它是刚刚开始起步的事物。

软件构件应该被设计和实现已使得它能够被复用于很多不同的程序。在20世纪60年代，我们建造了科学子例程序，它们是在一个广泛的工程和科学应用领域内可复用的。这些子例程序以一种有效的方式复用良好定义的算法，但是，其应用于有限。今天，我们已经扩展我们复用的视角到不仅包含算法还包含数据结构。现代的可复用构件封装了数据和应用于数据的处理，使得软件工程师能够从可复用的部件创建新的应用。例如，今天的图形用户界面是使用可复用的构件创建的，这些构件设计图形窗口、下拉菜单和大量的交互机制。建造界面所需的数据结构和处理细节被包含在界面构造的可复用构件库中。

### 1.2.2 软件的种类

软件可以应用于任何情况，只要定义了一组预先定义好的程序步骤（即一个算法，但也有例外，如专家系统和人工神经网络软件）。信息的内容和确定性是决定一个软件应用的特性的重要因素。内容指的是输入和输出信息的含义和形式，例如，许多商业应用使用高度结构化的输入数据（一个数据库）且产生格式化的输出“报告”，而控制一个自动化机器的软件（如一个数控系统）则接受限定结构的离散数据项并产生快速连续的单个机器指令。

信息的确定性指的是信息的处理顺序及时序的可预定性。一个工程分析程序接受预定顺序的数据、不间断的执行分析算法，并以报告或图形格式产生相关的数据。这类应用是确定的。而一个多用户操作系统则接受可变化内容和任意时序的数据，执行可被异常条件中断的算法，并产生随环境和时间的某个函数变化的输出。具有这些特点的应用是非确定的。

在某种程度上我们难以对软件应用给出一个通用的分类。随着软件复杂性的增加，其间已没有明显的差别。下面列出的软件应用领域指明了潜在应用的广度。

（1）系统软件

系统软件是一组为其他程序服务的程序。一些系统软件（如编译程序、编辑程序和文件管理实用程序）处理复杂的但也是确定的信息结构，其他的系统应用（如操作系统、驱动程序和电信处理器等）则处理大量的非确定的数据。不管哪种情况，系统软件均具有以下特点：与计算机硬件频繁交互，多用户支持，需要精细调度、资源共享及灵活的进程管理的并发操作，复杂的数据结构，多外部接口。

（2）实时软件

管理、分析、控制现实世界中发生的事件的软件成为实时软件。实时软件的组成包括：一个数据收集部件，负责从外部环境获取和格式化信息；一个分析部件，负责将信息转换成应用所需的形式；一个控制/输出部件，负责相应外部环境；一个管理部件，负责协调其他各部件使得系统能够保持一个可接受的实时响应时间（一般从1ms到1s）。