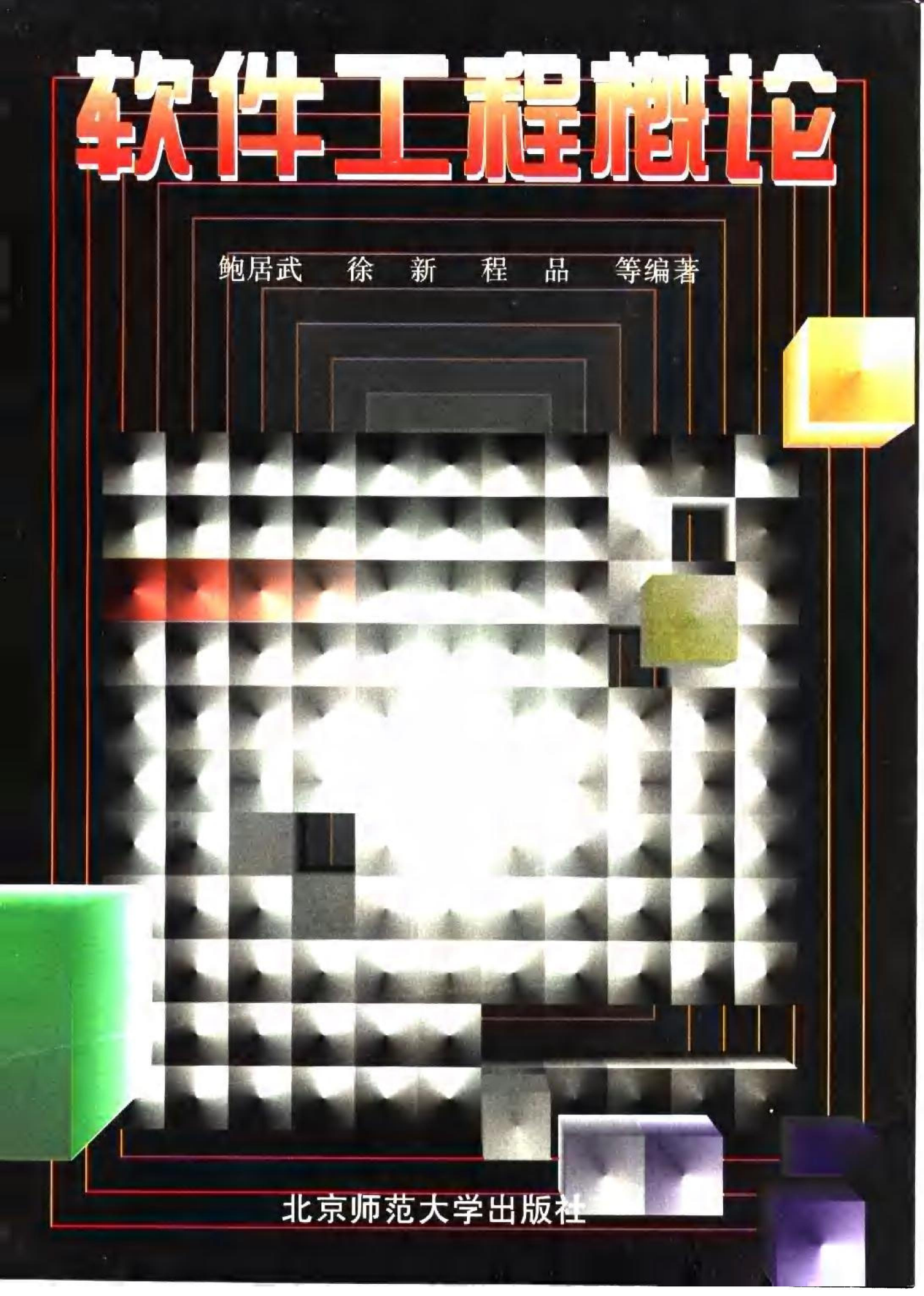
软件工程概论

鲍居武　徐　新　程　品　等编著



北京师范大学出版社

软件工程概论

鲍居武　徐　新　程　品　等编著

北京师范大学出版社

图书在版编目(CIP)数据

软件工程概论/鲍居武等编著. —北京 ：北京师范大学出版社，1996.10

ISBN 7-303-04257-1

Ⅰ.软… Ⅱ.鲍… Ⅲ.软件工程—概论 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（96）第 20012 号

内 容 简 介

本书目的在于向读者介绍当代软件开发过程中的一些重要知识。本书通过概括性的介绍，力图为读者编制高质量的软件打下一个良好的基础。

为便于读者阅读，本书在章节设计上，既保证全书的系统性，又使各章节自成体系。本书中间几章（4～7 章），每章主要针对软件开发过程中某一主要方面（如测试、维护等）进行专题论述。用于该方面的技术、工具都将在文中得以论及。总之，编写本书的目的是为读者提供一个信息的源泉。

本书可供从事计算机应用软件设计、开发和使用的技术人员学习参考。

北京师范大学出版社出版发行

（100875 北京新街口外大街 19 号）

北京师范大学印刷厂印刷　　全国新华书店经销

开本：787×1092　1/16　　印张：13.75　　字数：329 千

1996 年 10 月北京第 1 版　1996 年 10 月北京第 1 次印刷

印数：1—5000 册

定价：14.00 元

前　言

何人不想在竞争日趋激烈的高科技社会中取胜，在信息社会软件是致胜的关键。本书对当代软件开发过程中的一些重要知识的作了详尽介绍，本书并非是一本纯粹理论性的著作，有关软件工程理论方面的书很多，而是一本寓理论于实践中，向读者介绍软件技术的专著。

对于那些从事与软件有关的人来说，本书将是他的好帮手。本书对软件的几个主要方面作了详尽介绍，包括当代最好的经验、技术以及所用的工具，同时也提到了那些容易犯的错误。

本书的读者面应该是相当广泛的。对那些从未受过正规的软件训练，但其工作与软件越来越有关系的人有所帮助，是本书的最大目的。另外书中所提供的各种知识，无论是对专业的软件管理者，还是对即将从事软件工作的学生来说，都无疑是大有益处的。本书为软件专业人员提供了现代软件工程技术，这并不意味着软件工程技术是他们唯一可用的技术，有些软件专业人员希望把它作为处理问题的另一种手段。

软件自身好比是各种形式的文件资料的综合，对于一个结构不错的软件来说，要求它的各部分都必须易于理解与修改。这就要求必须认真处理软件开发过程中的每一部分。本书的绝大部分论述都集中在保证软件的可靠、长寿的主要步骤上，从需求捕获阶段直到维护阶段以致于最后的软件“退休”。

我们编制软件一定要使机器一直按照正确的指令去运行，使用它的人感到满意。编制软件是一件相当费神的事，它需要动脑。目前，在这方面还没有什么捷径可走，作者的最大心愿是通过本书总结过去几年的经验，帮助读者在编制软件时少走弯路。

在本书的编写过程中，薛成、李芳、孙学成、季文伟、袁芳、洪芳、王振华、王晓华、康振祯、刘守国、鲍文敏、马明、孙毅、鲍文静、等参加了编写工作。

由于我们水平有限，书中难免有缺点和错误，敬请广大读者批评和指正，以便进一步修改。

作者于北京

1996 年 7 月

目　　录

第一章　软件概述

> 九十年代有两种最实用的物品：石油和软件。在石油缺乏时，人们总能找到合适的能源来替代它，然而软件却以其独特的重要性在社会生活扮演着无可替代的角色。
>
> ——布鲁士·邦德(Bruce Bond)

随着计算机及其应用的日益普及和深化，计算机软件也相应在以惊人的速度发展着。现代计算机软件规模庞大，维护困难，有时难免造成大量人力、物力的浪费。为了解决软件开发和维护过程中遇到的问题，在计算机科学技术领域中逐步形成了一门新兴的学科——计算机软件工程学，通常简称为软件工程。本书主要讲述这方面的内容。以后各章中分别讲述了在开发软件系统中所用到的很重要的技术及技巧。在开始之前，让我们先回答一个很重要的问题：软件质量影响究竟有多大?这个问题很难正面回答，但从一些侧面的认识中，我们仍可以总结出问题的答案来。首先让我们看看由于软件失误而带来的难以估计的甚至可怕的损失；其次不妨粗略衡量一下，开发和维护软件所需耗费的人力、物力及财力。下面列举一些与之相关的例子，读者一定会从这些骇人的、代价昂贵的事例中认识到软件质量高低是何等的重要，从而得出中肯的评价来。

1.1　软件失败

由于一些非常明显的原因，软件开发者，尤其是涉及到对生命安全要求严格保证的领域，一般不愿意提及这样一种事实：那就是由于软件失败而造成的严重后果。

但事实上，这样的后果确实已经产生了。它提出了一个发人深省的问题：软件质量的提高已经刻不容缓。

下面让我们看几个真实的例子：

1.1.1　透视仪器

有一种透视仪器是这样设计的：一般情况下这种仪器释放极少量的射线；给病人作检查时，大量的辐射释放出来，但这必须仅局限于病人的病变部位，而其余部位则需采取严格的屏蔽措施以防辐射。在一次意外的事故中，由于仪器对一位病人没有采取上述的有效屏蔽，大剂量的射线辐射到病人全身，导致了病人的不治而亡。

与早期的硬线性(hardwired version)仪器不同，现在的透视仪器由一软件系统所控制。

早期的硬线型性仪器，有一种自动保护装置，假如病人没有被有效的屏蔽，仪器将自动防止类似上述事故的发生，但现在软件型仪器却缺少这一装置。究其原因，问题在于设计软件时没有充分地考虑到这一情况将会出现。

1.1.2 失去控制的鱼雷

在设计软件的时候，其可行性和可靠性是很必要的。同样在设计控制鱼雷的软件系统时，必须考虑到假如鱼雷错误返回时，如何避免误伤自家船只。在设计中，通常使鱼雷掉头重新攻击目标或者在鱼雷错误返回时自动引爆以免造成自伤。

但在一次实弹测试过程中，自动发射装置失灵了，鱼雷停留在发射管内。船长决定放弃这一次测试，返回基地。然而就在船只掉头的时候，鱼雷却尽到了自己的责任，在发射管内爆炸了。

1.1.3 自动着陆系统

最初设计的飞机自动着陆系统分为两阶段工作。第一阶段由飞机发射导航波束来探索着陆路径。假如波束发射没有成功，则系统供给飞机能量，使之继续绕行，再进行一次努力；第二阶段当地面指挥中心接收到几英里外的飞机发出的信号后，指挥飞机关闭发动机然后仰起机头，引导飞机安全着陆。事后分别对这两种系统进行测试，证明其符合设计需求。

第一次上天试飞，证实了这两种系统工作完好。几年以后又开发研制了几种与它相类似的着陆系统，经实验证明虽不是完美的，但也没有发生太严重的失误，每次实验飞机都能正确无误地着陆。但是几周过后，整个系统却被迫修改(即相应的软件也需改动)。这并非是软件出现了错误，只因为试飞跑道断裂而不适于起飞了。

这个例子说明了很重要的一点：尽管并非所有的系统故障都由软件错误而造成的，但是当一些很难由人为改动的系统因素发生变化时，则系统软件应作相应的变动处置。

1.1.4 化工厂

程序设计员被告知必须考虑这一设计需求：如果系统监测到某一处出现了问题，则系统应发出警报并立即停止运行。但事实上程序设计员并非如化学工程师那样事事考虑周到。

有一次，系统监测到油箱储油太少，按照设计需求，系统应发出警报并立即停止运行。但凑巧的是，系统刚刚给反应器加进催化剂，由于往制冷装置里加进的冷却水量不够，反应器过热，压力太大，阀门被冲开，大量的有毒气体泄入大气中。

上述的事实并不是危言耸听，实际生活中类似的悲剧常常重演。最近有一则关于高速列车制动系统的报告，与自动着陆系统相类似，列车制动系统也分为两类：一类是 50 千米/小时以下的制动系统，另一类是 50 千米/小时以上的制动系统。系统设计好之后进行实验，却发现列车时速刚好为 50 千米/小时，制动系统丝毫不起作用。

再一次重提这些不幸的事实仅为了说明灾难与完美的界限是那么模糊。

从上述事例中我们不难发现，软件失败的根本原因是软件开发人员不能真正全面、准

确、具体地理解用户的需求，熟悉系统的结构。软件质量的好坏并不取决于程序代码组织的优劣，重要的一点是看它能否全面反映出整个系统的结构和是否易于维护。

软件影响究竟有多大?现实生活中，从核电站到各种运输工具的制动功能等各个方面都不同程度地与软件有关。软件的失败往往导致灾难性的后果，凡事要做到尽善尽美是不现实的，而消除那些易见的错误却是很有价值的。

1.2　软件成本

许多人对软件的成本和价格比较关心，为了满足这方面的需求，有许多调查资料表明了软件在今天及今后的价值究竟有多大。

首先，让我们看一些反映软件价格发展趋势的资料。

- 据查，软件行业的总产值已占英国国民生产总值的5%，而且这种趋势还在上升，图1.1(A)给出了过去十年软件发展的大致趋势。
- 在整个计算机系统费用中，软件所占比例由1980年的40%上升到1990年的80%(图1.1(B)所示)。
- 在整个欧洲大陆，软件服务市场的营业额，在1990年大约为400亿ECU(欧洲货币单位)，到1993年已达到600亿ECU多。

除了以上这些数据外，这里还提到一些关键性的问题，主要有：

- 一般情况，大的软件系统常滞后于计划。
- 仅10%的软件能按时完成并交付使用。
- 需求比较高的软件有25%根本不能完成。
- 60%多的工厂产品经营者仅有一点或几乎没有现代软件工程这方面的经验与技术。

在英国，每年由于这几个方面的原因所造成的损失约有20多亿英镑。如果再算上由于软件失败所造成的损失，那么数字将是惊人的。

以上仅是泛泛而淡，为了让读者进一步了解软件开发具体情况，现在让我们看一个很具代表性的软件开发公司。

这个公司实际并不存在，但所有的假设都是合乎实际的，并能合情合理地代表整个软件行业全貌。

假设：

- 该公司共有100名从事软件开发与维护的工程师。
- 根据美国近年来发表的调查资料表明，从事软件开发与维护的人员比例为47：53，最合乎软件这个行业的需求。
- 从语句代码这个角度看，软件开发人员平均每天编写20条语句，维护人员一年大约可维护17000条语句。

上面的数字基本上合乎实际。前者是从经验中作出的最合理的假设，后者则是一些统计数据的结果。

结论：

- 若公司从事软件维护的人员为53人，那么他们一年可维护900 000条语句。
- 其余47个人从事软件开发，按一年45周，每人平均每天编写20条语句计算，这样一年总共可编写210 000条语句。实际上由于各种各样的原因，一年并不能编写这么多语句。如果有些语句需要修改(假设软件生存周期为10年，那么每年约有90 000条语句需修改)，那么计划就不能按时完成(如果有25%的任务根本没干，那么一年总共才增加155 000条语句，其中90 000条是过去需修改的语句，而仅有65 000条是新编写的。
- 对于一个软件开发公司每年新增加65 000条语句，这个增长率实际是很低的。

上面所提到的数据并不是凭空捏造的，而是根据大众经验与各种调查资料得出的最为合理的假设。下面让我们看两个关键性的难题：

- 假如这些小软件公司不得不改变过去编写的软件，无论怎么说，这种代价都是极昂贵的。
- 每年用于软件开发与维护的预算非常庞大，而且有逐年增长的趋势。

上面两个难题表明一个质量不好的软件的隐性成本是极高的。

$$\text{比例}=\frac{\text{当年国民收入}}{\text{1979年的国民收入}}$$

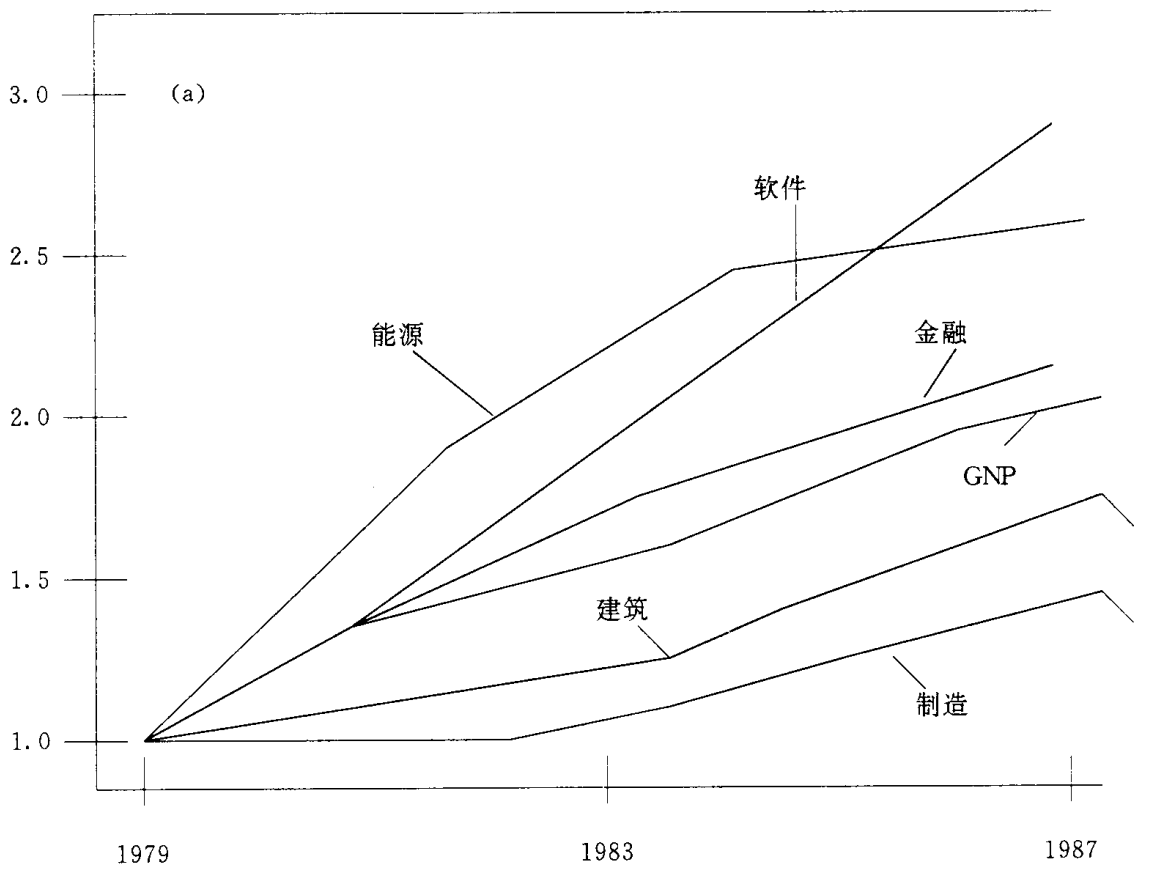


图1.1(a) 软件在国民收入中所占比重逐年上升(英国)

如果软件不能很好地满足用户需求，那么就会成为用户沉重的负担。

那么究竟由谁来承担由软件失败所造成的损失呢？发展趋势告诉我们：是软件开发人

员，而且还要承担应有的法律责任。在今天的欧洲，软件开发人员不得不尽最大努力来确保自己的软件能安全运行。

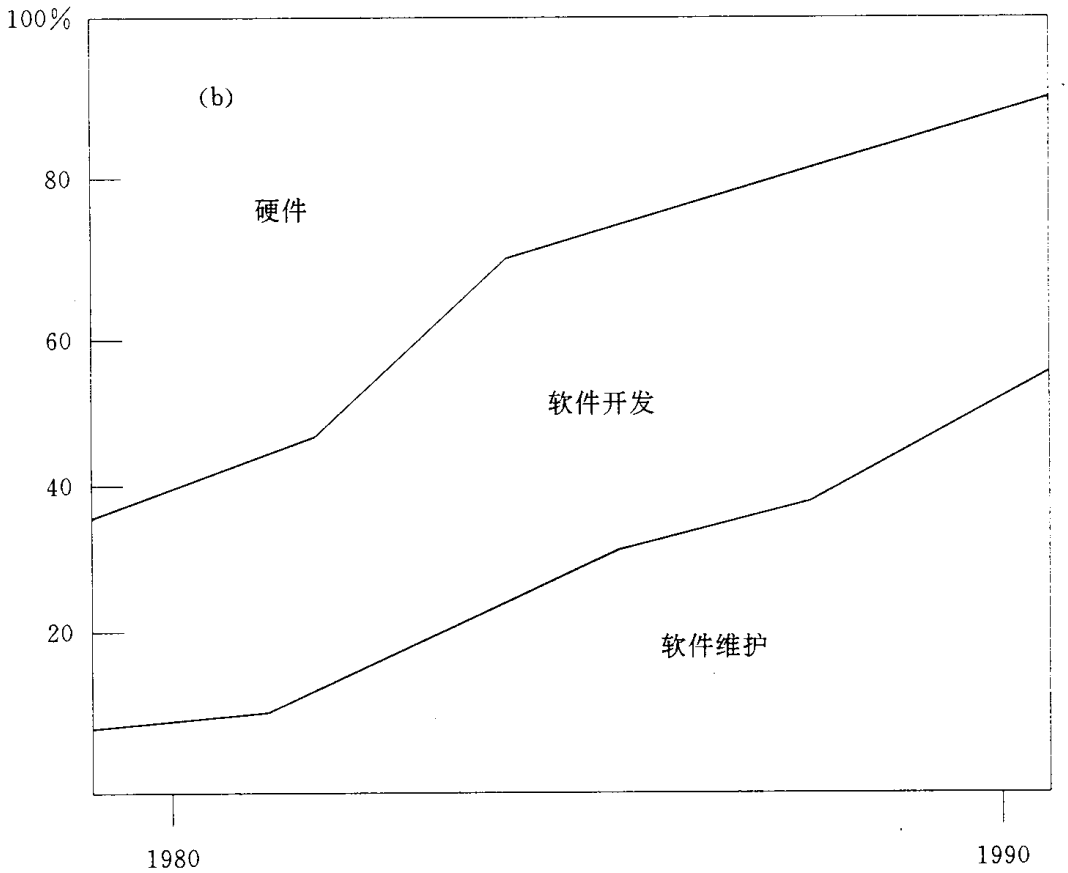


图 1.1(b)　软件在系统中地位越来越重要

无论是从安全，还是从金钱的角度来讲，提高软件质量都是刻不容缓的，但问题是怎么提高软件质量。

回答这个问题之前，先让我们根据需求给“质量”下定义：软件质量好坏就是软件能不能很好地满足用户需求和能不能全面代表系统的所有特征。软件质量的好坏常由下面几个方面来衡量：可操作性、可靠性和可用性等。使用者可以对软件某些方面需求有所放松，但对可靠性却丝毫不能含糊，否则将会铸成大错。下面是一个用户总结的软件质量衡量标准：

- 可靠性

  一个好的软件系统不会总因为外来干扰而发生事故。软件系统日趋复杂，组成整个系统的各独立模块总需求有一定的可靠性。

- 安全性

  软件应具有很好的保密性，能正确访问各种信息。这是软件所必备的性能。

- 适应性

  软件能很好地适应于外部环境的变化，如果它不能迅速地适应市场需求和各种法律限制，就有可能被淘汰。

- 可操作性

  用户希望它能与原系统其他环节相配合，其响应时间应能满足这个需求。

- 可用性

  易于使用，便于学习，同时还富有趣味。用户最大的期望是它能同时适用于各种现代系统。

对于软件质量标准很难一一枚举，上面列举的是软件所应达到的最基本的要求。表 1.2 列举了用户所关心的软件质量的几个方面。

要满足用户所需求的质量标准确实不容易，如果是举手之劳的话，这本书也就要写成一本小册子了。现在问题的关键是软件开发人员只能保证在某一整体范围达到技术和结构的需求，而没有固定的公式与技巧来帮助他们确定满足用户的真正需求。

用户需求	用户关心	系统实现
功能	它是否安全	安全性
	出错次数是否频繁	可靠性
	是否容易学习	使用性
符合需求	效率如何	可操作性
	是否容易受到干扰	适应性
	是否容易操作	使用性
	是否容易维护	可维护性
灵活性	是否容易修改	适应性
	是否具有移植性	重复使用性

表 1.2 用户需求和系统要求之间的联系

后面的几章将具体阐述有关软件质量的几个方面。现在我们只是简单提一下，使读者对软件质量有一个初步认识。

1.3 质量保证

目前，在开发软件的过程中采取了一系列措施以保证质量，符合某些标准，诸如固定的格式、传统习惯、代码标准等等。这些措施都是软件开发过程中最基本的需求。

软件开发采取的一般步骤：

- 记下要做的一切。
- 开始干。
- 结果证明做到了。

为保证软件质量，可依据的标准有 BS 5750，ISO 9000 和 AQAP，无论那一个质量组织所制定的有关标准，也只能从最基本的情况出发，某些地方难免会有疏漏。

- 所有引导软件开发的方法和标准仅能提供一个大致范围。
- 它们都提到“该怎么干”，但对“为什么这样干”却说不出来，如果它们被认为是很不中肯的，那么就有可能被废弃。
- 实际上这些标准组织往往过于书本化，对软件开发并不能起到真正有益的帮助。

这些方法、标准为软件设计打下良好的基础，对今后的各个步骤也是必要的。

制定一项标准必须统计大量的数据及分析许多具有代表性的事例，制定软件标准也不例外。在软件标准公布实施之前，必须对软件开发过程中被称为“质量成熟”的这一关键阶段

的各个过程进行分析。这一阶段的各个过程如下所述：

- 自由阶段

  系统各部分都不受控制，一切处于无序状态，各种情况都是暂时的。

- 强化阶段

  软件开发过程的核心阶段，一切工作有条不紊，按部就班，它的作用无可非议。

- 测试阶段(质量测试)

  用已知的输入在已知的环境中动态地运行系统，观察测试结果与预期结果是否一致。测试阶段往往能发现系统中的错误。

- 改进阶段(质量提高)

  现实生活中没有十全十美的事物，因此通过测试观察结果，分析问题，从而改进系统是很必要的。

- 理论阶段(质量设计)

  一旦很好地了解了某一过程，就可以用理论来具体地阐述其因果关系。

上述各阶段的联系如图 1.3 所示，软件设计人员在设计过程中没有什么定理可遵循，但是可以通过“强化——测试——强化”的循环过程，使软件逐渐从设想走向实际，再从实际走向科学。尽管我们可以一步一步地使软件质量走向成熟，但关键的第一步却是万万不能忽视的。

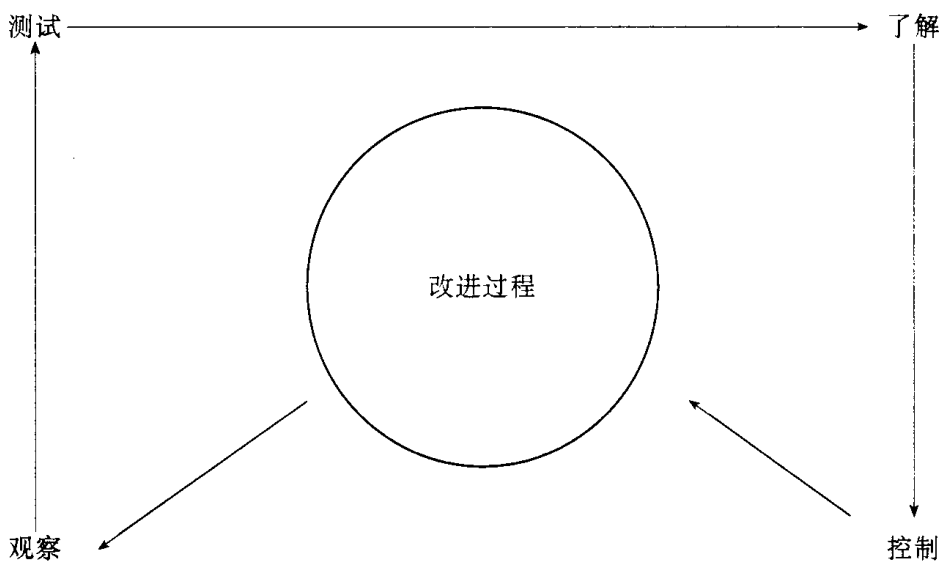


图 1.3　提高质量的工作过程

1.4　质量管理

为了更好地说明质量管理在软件开发过程中的重要性，让我们先看一看其它领域是怎样做的？

军队的严格管理可以说是这方面的榜样，命令一旦下达，谁也不能违背。这一切可追溯