HZ BOOKS

软件工程技术丛书

基于项目的软件工程
面向对象研究方法

Project-Based Software Engineering
An Object-Oriented Approach

Evelyn Stiller
(美) Cathie LeBlanc 著
(普利茅斯新罕布什尔州立学院)

贲可荣 张秀山 等译

机械工业出版社
China Machine Press

Pearson Education
培生教育出版集团

软件工程技术丛书

基于项目的软件工程

——面向对象研究方法

（美） Evelyn Stiller
Cathie LeBlanc 著
（普利茅斯新罕布什尔州立学院）

贲可荣 张秀山 等译

机械工业出版社
China Machine Press
Pearson Education
培生教育出版集团

本书是集软件工程实用性与实践性于一体的第一本教科书，阐述了软件工程生命周期的各个阶段，并通过对两个现行案例的分析来认识开发过程中的概念化、分析、设计和实现阶段。这两个案例可以由一个小型软件开发组来实践，同时给出各阶段的成品。书中还包括一个完整的软件开发项目，学生可将对各阶段的理解和经验应用于其中。

本书集中讨论面向对象软件开发，并按此范型组织内容。本书不依赖于具体的程序设计语言（必要的代码示例用Java给出），并采用统一建模语言（UML）的子集来为软件建模，解释学生工作中需要用到的符号。

本书为学生而写，可作为计算机专业本科生、非计算机专业研究生的软件工程教材，亦可作为软件开发者、组织者和管理者的参考书。

本书版权登记号：图字：01-2001-3876

图书在版编目（CIP）数据

基于项目的软件工程：面向对象研究方法/（美）斯蒂尔（Stiller, E.），（美）勒布朗（LeBlanc, C.）著；贲可荣等译. -北京：机械工业出版社，2002.6
（软件工程技术丛书）
书名原文：Project-Based Software Engineering: An Object-Oriented Approach
ISBN 7-111-10101-4

Ⅰ. 基… Ⅱ. ①斯… ②勒… ③贲… Ⅲ. 软件工程 - 研究方法 Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2002）第029330号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：姚 蕾
北京忠信诚胶印厂印刷 · 新华书店北京发行所发行
2002年6月第1版第1次印刷
787mm × 1092mm 1/16 · 17印张
印数：0 001-5 000册
定价：30.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

译者序

随着社会对软件的日益依赖，软件产业在全球经济中的作用愈加重要。但是，现今大部分的软件仍是由技术人员利用其掌握的技能和经验编制出来的，往往不能确保软件的质量和进度。人们为了克服“软件危机”，在软件工程领域做了大量的工作，逐渐形成了系统的软件开发理论、技术和方法。软件工程已成为信息社会高技术竞争的关键领域之一。

“软件工程”是高等学校计算机教学计划中的一门核心课程。软件教学的实践表明，如果没有亲身体验，学生很难相信软件工程技术的好处。让学生相信软件工程对其软件开发生涯至关重要的最有效的方法是让他们完成为时一学期的软件项目的开发。以本书为教材的软件工程课是一门以面向对象的软件开发为重点，集实用性与即用性于一体的课程。市面上软件工程方面的教材大多重理论轻实践，而那些基于项目的教材也没有把重点放在面向对象的范型中来。本书正好填补了这一空白。

本书重点介绍软件工程的实际应用，而且把重点放在应用于班级项目的面向对象的软件开发方法之上。本书介绍了面向对象项目的概念、分析、设计及实现，对软件工程技术作了历史综述。本书提供了两个现行的案例分析，作为为时一学期的项目开发的样板。案例分析中含有成品的样本，举例说明了学生在其项目的生命周期中所要交付资料的类型。本书不依赖于具体的程序设计语言（必要的代码示例用Java给出），并采用统一建模语言（UML）的子集来为软件建模，解释学生工作中需要用到的符号。

让学生参与一学期的集体项目，使学生体验到专业合作的好处和重要性，对此他们会认为是很大的收获。这一点在我们采用本书译文初稿作为软件工程课程辅助教材时得到了验证。

本书的前言、第9至12章由张秀山翻译，其余各章由贲可荣翻译。全书由贲可荣审校。肖斌、俞立军、徐荣花、吴铁洲、周娟、孙宁和王淑雪等参与了本书的翻译工作。

由于各种原因，译稿难免存在错误和疏漏。欢迎读者批评指正。

本书可作为计算机专业本科生、非计算机专业研究生的软件工程教材，亦可作为软件开发者、组织者和管理者的参考书。

译　者

2002年5月10日

译者简介

贲可荣　男，1963年8月生，江苏省海安县人。现任海军工程大学教授。

1983年在苏州大学数学系获理学学士学位，1986年在南京大学数学系获理学硕士学位。1986年8月至1990年3月在海军工程大学计算机系任教。1994年6月在国防科技大学获工学博士学位。攻读博士学位期间，由国防科技大学计算机学院陈火旺院士指导，主修计算机软件。1994年12月任海军工程大学计算机系副教授。1995年起担任计算机应用技术专业硕士生导师、海军工程大学学位评定委员会委员。2000年3月至2001年3月在武汉大学软件工程国家重点实验室作访问学者。2000年被教育部确定为海军首批十名骨干教师之一。2000年12月晋升为教授。

译著有《净室软件工程——技术与过程》（2001年由电子工业出版社出版）和《能力成熟度模型(CMM)：软件过程改进指南》（2001年由电子工业出版社出版），参与编写了《海军高科技知识教材》（1997年由海军司令部出版）、《计算机科学技术百科全书》（1998年由清华大学出版社出版）。先后承担两项国家自然科学基金项目、一项国家863项目。在《中国科学》、《软件学报》、《数学年刊》等刊物和学术会议发表论文50余篇。成果获军队科技进步奖。主要研究方向包括：软件可靠性、软件质量保证技术、形式化方法等。兼任中国计算机学会计算机理论专业委员会委员、中国造船工程学会电子技术学术委员会委员、海军工程大学职称评审委员会委员，《海军工程大学学报》、《舰船电子工程》等刊物编委。

前　言

软件工程的教学实践表明，如果没有亲身体验，学生很难相信软件工程技术的好处。让学生相信软件工程对其软件开发生涯至关重要的最有效的方法是让他们完成为时一个学期的软件项目的开发。普利茅斯新罕布什尔州立学院开设的软件工程课是一门以面向对象的软件开发为重点、集实用性与实践性于一体的课程。然而，多年来我们却苦于找不到合适的教材。市面上软件工程方面的教材大多重理论轻实践，那些基于项目的教材并没有把重点放在面向对象的范型中来。我们编写此书的目的就是要填补这一空白。

本书重点介绍了软件工程的实际应用，理论性的概念和术语只有在为确保软件成功开发所必需时才予以介绍。软件开发的方法虽有不少，但我们还是把本书的重点放在应用于班级项目的面向对象的软件开发方法之上。

让学生参与一学期的集体项目，可让学生体验到专业合作的重要性，对此他们似乎很乐意。教授一门基于项目的软件工程课程最重要、也最困难的一点是如何选择合适的项目。因为所选项目必须足够复杂，以便让学生以3~5人为单位组成一个开发小组，同时又必须能在15周内完成。比达到目标更具挑战性的是所选的项目必须引起学生的兴趣，能激发学生的积极性。为此，我们在书中提供了一个班级项目。普利茅斯新罕布什尔州立学院学生已经试验了这个项目，由具有不同编程技巧和分析能力的学生组成的4人开发小组以极大的热情成功地完成了这个项目。

本书主要面向缺乏或没有计算机科学理论基础的计算机科学专业的本科生。本书在编写时还做到尽量不依赖于某一具体编程语言，当需要语言细节时，选用Java语言。本书不用作软件工程技术及编程的参考手册，而是提供一种可在15周内完成一个大型软件项目的专门开发方法。

由于假定学生在一学期的时间内完成项目，因此在第2章中介绍了学期计划。在此计划中，项目在学期开始后不久即展开。因为我们想让学生尽可能多地实践软件开发的方法，所以有些主题作了一些简化。尤其是，要从与非专业用户的讨论中提取功能需求是一项困难的任务，需要丰富的实践经验才能顺利完成。因而，我们假定在学期开始前需求已差不多由授课教师提取完成。

为增强本书的实用性，我们还提供了两个现行的案例分析，用作为时一学期的项目开发的样板。案例分析中内含有成品的样本，举例说明学生在其项目的生命周期中所要交付的资料的类型。

本书另一重要特点是它用一种几乎是独一无二的方式重点讲述了面向对象的软件开发范型。尽管我们将面向对象的方法看作是业界先前采用的范型的逻辑延伸，但是本书还是将面向对象项目按概念形成、分析、设计及实现的结构予以介绍，并对软件工程技术作了历史综述，向学生介绍了面向对象范型的前身。

虽然本书并没有把长期的软件危机作为使用软件工程技术的独特动机，但是书中还是介绍了一系列软件开发的灾难故事，以便学生了解如果忽略软件工程诸方面会造成什么样的后果。这些

故事不属于入门材料，将它们放在书中的后半部分，以免推迟启动软件工程项目所需的章节。

在介绍组成面向对象范型的技术时，使用了统一建模语言（UML）为软件建模。由于UML大得有些吓人，因此本书按照“当需要时”的原则介绍了该语言的一个子集。本书不用作UML方面的综合参考书，这方面的参考书已经很多。在本书中，UML只是一种工具，正如它在“现实世界”软件开发中被用作工具一样。

适于教学的特点

- 每章开头部分都列出了该章所包含的重要概念。
- 由我们的学生测试过的班级项目贯穿始终。项目大到由3~5名学生在一学期内完成。每一章中都有完成此项目所必须进行的一系列活动，以及随之而来的项目每部分的成品规格说明。
- 虽然书中包含了一个班级项目，但在讲授此书时，可以简单地抛开它。如果教师不用此项目，也可用其他项目（或一组项目）代替。
- 每一章的最后都是复习题，可让学生进一步实践书中涉及的内容，题目的复杂性和难度各异。
- 贯穿每一章的习题通常用作实验。大多数习题可由教师安排在课堂上完成，也可在课下完成。
- UML只在必需时才出现。当在开发方法的特殊步骤需特殊的建模技术时，对该技术才进行介绍并给出示例。用此方法介绍了UML的一个子集。
- 案例分析贯穿始终。第一案例分析出现在书中开始部分，随着对各个步骤的介绍而不断深入。第二个案例分析出现在第6章，作为开发方法的设计及分析阶段的复习，第二个案例分析一直延续到书中后面的章节。
- 要点栏用来快速回顾软件开发方法。
- 章节安排上以能让学生在一学期内完成班级项目为原则。例如，在最后4周，学生正在进行项目的编码和测试工作，实现和测试的内容已作了介绍，此时书中开始讲述项目管理、风险管理、设计模式和软件开发的灾难故事这样的内容。
- 书中包含已在课堂上进行了测试的项目和进度。
- 最后一章的主角转向学生，要求学生回想其班级项目的实践。一些项目开发组很可能没有其他组那么成功，讨论让学生总结他们小组的活动并提出改进措施。最后一章引导学生针对其项目向教师和其他同学进行正式而专业的展示。

教师用补充材料

对以本书为教材的教师，以下材料可供使用：

- 书中每幅图的PowerPoint幻灯片。

- 每章内容的PowerPoint 幻灯片。
- 复习题部分的答案。
- 每一章习题的样板答案或引发讨论的线索。
- 嵌入式班级项目的成品样板集。
- 两份可选的班级项目的材料。
- Game2D案例分析的源代码。

请在www.aw.com/cssupport上核实本书的在线资料并获得这些补充材料的更多信息。

除了这些资源之外，我们还预计每两年出版一本包含与两个不同班级项目有关的材料及习题的学生用书。

班级项目

我们鼓励授课教师用自己的项目，或用补充材料中提供的项目代替书中的项目。班级项目开发的每一章中均有说明班级项目的目标及目的的内容。这些部分具有一般性，既适合书中的项目也适合补充材料中的项目。

书中有些章节专门介绍了所含的班级项目。之所以这样做是因为样本项目可作为极好的例子来讲述书中涉及的一些设计目标。要理解这些章节，学生并不需要对所含项目的细节很熟悉，而只需要理解在因特网上进行多人游戏的思想。示例则说明了启动这一多人游戏或对奕走子所包括的一系列事件。

下面几节包含专门针对班级项目的内容：

- 1.10节包含需求规格说明。可用一个可选项目描述来代替。如果正在使用一个可选项目，则该节应跳过。
- 2.2.2节介绍了一个易被任何熟悉普通棋类游戏的人所理解的非正式场景。即使你采用的是另一个班级项目，我们仍推荐此例。
- 4.9节讲述了样本班级项目中进程间通信的建模。这样的讲述易被那些熟悉因特网多人游戏的人所理解，我们建议那些采用一个可选班级项目的教师使用它。

致谢

感谢Addison Wesley出版公司的Maite Suarez-Rivas和Katherine Haruntunian在本书的编写过程中给予的大力支持。Jody Girouard为那些以此书作为教材的教师开发了大量材料，并从学生的角度为此书提供了反馈意见。John Girouard、Mark Henwood、Nick Rago和David Sleeper是使用UCCD（以用况为中心的开发）方法的第一期软件工程课程的学生。他们完成了“银河侦探”（*Galaxy Sleuth*）游戏，并对上述方法提出了大量反馈意见。泽维尔（Xavier）大学的Liz Johnson在其软件工程课程中使用了本书的较早版本，帮助我们了解了在本书编写方面还有哪些内容表达得不清楚。最后要说明的是，有许多审阅者在原稿写作过程中提出了宝贵的反馈意见。有些审阅

者我们至今不知其名，在此一并表示感谢　下面各位审阅者对最新手稿进行了审核，并帮助我们精练文字以便读者更好地理解本书的内容：Michael Beeson（圣何塞加利福尼亚洲立大学），Jorge L.Diaz-Herrera（南方州立工业大学），Jozo Dujmovic（旧金山加利福尼亚洲立大学），Mohamed Fayad（内布拉斯加大学林肯分校），J. W. Fendrich（伊利诺依州立大学），J. A. "Drew" Hamilton（海军研究生院），Alex Iskold（纽约大学），Jonathan Maletic（孟菲斯大学），Michael McCracken（佐治亚理工学院），Fatma Mili（奥克兰大学），Robert Noonan（威廉和玛丽学院），Srini Ramaswamy（田纳西理工大学），Steve Roach（德克萨斯大学埃尔帕索分校），Don Shafer（阿森斯集团公司），Bill Shay（威斯康星大学格林贝分校），James E. Tomayko（卡内基－梅隆大学），David Umphress（奥本大学），Shon Vick（马里兰大学巴尔的摩县分校），Linda Werner（加利福尼亚大学圣克鲁斯分校），Janusz Zalewski（中央佛罗里达大学）。

目　录

第1章
软件工程引论

1.1 关键概念

以下列出本章的关键概念与技巧：

- 软件开发的复杂性根源
- 软件开发项目为什么失败以及怎样失败
- 为什么人与人之间的交流是困难的
- 为什么维护需要大量人力资源
- 问题的概念化
- 问题刻画
- 结构化程序设计
- 功能分解
- 抽象层次
- 结构化分析与设计
- 抽象数据类型
- 继承
- 数据建模
- 软件开发范型的要素
- 软件开发范型的目标

1.2 为什么要建造软件

假想你已经大学毕业，正在为一家国际性软件开发公司工作。你乘坐一架飞机，跨越这个国家飞往一个用户所在地。航班机舱光线很暗，大部分乘客都在打盹，机舱外面很黑，由于浓厚的云层，可见度很低。突然，飞机向左转向，你看到外面是另一架飞机的机翼，它危险地贴近你乘坐飞机的机翼飞过。飞机的突然转向惊醒了大部分乘客，机舱中立即喧哗起来。机组人员要么不知道，要么不说到底发生了什么事情。当你在一个半小时后最终着陆，你感到幸运，但不知道能不能产生足够的安全感再去乘坐飞机。

据我们目前所知，最近没有像刚刚描述的情景发生。我们感觉只是幸运而不是好的计划防止了此类事故。实际上，在1998年12月初的一个星期中，报道了3个相互独立的事故，在这些事故中由于计算机暂停导致两架飞机非常接近地飞行[17]。美国大部分空域中目前控制空中交通的系统已使用25年以上。软件非常陈旧，预计到2003年仅能处理一小部分交通问题[72]。软件同时运行在已有25年历史的硬件上，找替代部件的困难不断增加。目前，此类系统故障开始多了起来[33]。整个系统需要更新。

自20世纪80年代早期，更换空中交通控制系统已成为美国联邦航空管理局（FAA）非常优先的任务。1989年IBM联邦系统公司获得更换该系统的合同，截止期为2001年，预计投入25亿美元[21]。由于面临着极苛刻的需求，因此该软件项目是已进行的最复杂的项目之一。例如，空中

交通控制系统必须具备全局完整性并且每周7天，每天24小时不能停止工作，甚至在升级时或正常维护时，也不允许有停顿时间。任何错误的数据都会引起重大伤亡，任何停机均会导致世界范围出行的延误及潜在的危险。该系统的反应时间不能超过2到3秒。另外，该系统设计时必须考虑到允许小飞机的私人驾驶员和拥有者继续使用其旧设备，并且允许软件可以移植到将来开发出的新硬件上[30]。

在本书中，作为一个可怕故事描述的空中交通控制系统升级是失控的软件开发的一个例子。当IBM获得了该合同后，该系统的花费主要用于软件开发，而仅有8万美元用于硬件[78]。在1993年，IBM联邦系统公司这家负责合同的IBM的子公司被卖给了Loral公司[7]。1994年年初，该系统已花费了23亿美元，但尚未提交系统的任何程序段。预计到1994年年底，整个系统花费将增至50亿美元[21]。1994年晚些时候，FAA局长David Hinson调查了该项目惨败的原因。在调查中，Hinson解释说该系统的问题出在过度急躁的开发和实现计划（包括费用与进度的估计），尤其是给定的软件系统的复杂性，没有充分考虑IBM公司的生产力，特别是在开发的早期阶段需要投入的生产力。最终，实际上FAA并没有明确掌握该系统的某些基本需求[3]。另外，FAA代表与承包商的代表说过，技术的发展比使用这些技术的能力要快得多[3]。作为本次调查的结果，FAA取消或修改该系统四个主要部分[63]。由于已经认识到当前空中交通控制系统中的计算机非常落伍而使许多旅客处于危险之中，因此FAA订购了一套作为权宜之计的系统，将由Formation公司开发，它是一家位于美国新泽西州穆尔镇的软件开发公司[33]。1996年1月，Loral公司被Lockheed Martin收购[71]。Lockheed Martin公司那时失去了1996年9月与Raytheon公司签订的合同中的主要部分[77]。

这个故事并没有结束。我们仍不清楚是否一个可靠的空中交通控制系统能够在我们所描述的混乱中诞生。解决复杂的问题是困难的，但并不是不可能的。软件工程——本书的主题——是找到一条途径，避免软件开发过程走进迷宫和尽可能地节约资源。

在过去的30年中，软件开发的状况可被描述成一种社会性的苦恼[88]。大规模系统软件开发好像一条陷入沥青坑的恐龙[18]。这样的描述表明，虽然有一些成功的例子，但大规模系统软件开发都有一段漫长而痛苦的失败历史。一些成功的例子包括给人印象深刻的当前一些办公应用软件。然而，每一个成功的软件开发项目都存在许多已经渐渐被遗忘的不为人们注意的初期失败。由于那些制造软件的公司想避免负面影响，因此这样的失败很少被公之于众。一般说来，大规模系统软件的开发是一项具有冒险性的工作[9]。

核心的问题是，为什么会有这么多的软件开发项目失败？答案只有一个词，即**复杂性**（complexity）。许多软件项目在某些层面上是非常复杂的。例如，在某些应用领域，软件的操作依赖于专业知识，这种软件往往非常复杂。另外，懂得应用领域专业知识的人很可能不是实际编写软件的人。因此，这些应用领域的专家必须将他们的需求告诉软件开发的技术人员。在处理完理解问题和将来软件用户的需求这个复杂过程后，就要进行开发软件。任何写过程序的人都知道，开发实际上用于完成特殊任务的软件是一个复杂过程。最终，许多已着手进行的软件开发项目的规模都非常大，太大，事实上已经不可能由一个人完成。因而开发技术小组必须将开发任务分成易于管理的模块，一旦每个部分全部完成，小组要将它们组装起来使它们作为一个协同的整体来工作。将大项目分割成几个小部分，每个小部分由不同的个人来开发，并且

保证这些部分可以在一起工作，这样的过程已成为软件开发过程复杂性的另一个来源。

考虑一下创建一个遗传信息库的工作。你作为一个技术人员来创建管理该库的软件，很可能对遗传信息的所有可能形式与应用只有一个肤浅的了解。你的信息收集过程从哪里开始？你能够仅简单地存储与生物体有关的遗传序列吗？应该考虑分子的排序类型吗？你如何表示一个三维分子？科学家是否想在三维空间处理遗传信息？当你考虑你所面对的任务时，很可能还有别的成百上千的问题出现。

为了使你的软件开发成功，你必须与懂得这些问题的人交谈，这些人应该是遗传学领域的专家。你可以通过与顶尖的遗传学家交谈而开始你的信息收集工作，这样你很快就会发现遗传学家的专业词汇与你自己所掌握的词汇完全不同。当她试图解释三维分子结构的相关特性时，你对大多数词汇感到不熟悉。当你鼓励她刻画她脑子中的特别体系时，往往她不能够用你所能理解的方式讲清楚这个体系。这种不同领域专业技术词汇的不相容性被称为**阻抗不匹配**（impedance mismatch）[14]。在某个科学领域工作的人员没有必要去体验阻抗不匹配。许多行业已经开发了他们自己的词汇。

口头、书面语言中所固有的含糊性为领域专家与开发软件的技术人员之间的交流又增添了一层复杂性。例如，或许权威遗传学家表述了如下系统需求“在计算机屏幕上再现三维结构，这样科学家就可以围绕图形表示来操纵分子”。你可能认为你已明白了这个需求，因此你开发了相关软件。假如显示屏不能完全显示一个分子，则用户可以四处卷动画面来查看超出显示屏边缘的部分。然而遗传学家在过去已经使用了一种允许用户从分子内部叠合部分产生三维视图的软件。那位遗传学家将会喜欢这个新软件具备相同的功能。使人与人之间的交流变得复杂的真正原因在于我们都拥有自己不同的**背景知识**（background knowledge），这种差异会使我们基于以往的经验对相同的现象做出不同的解释。

对于未来的用户，不论他们是否是领域专家，都应该让他们参与系统开发的每个阶段。在软件开发过程中，软件开发的计划有必要允许最终用户来评价该系统。不幸的是，许多软件企业的人认为只需要在系统开发早期阶段和开发完成时，向用户了解专业知识。这种态度很可能会导致许多软件开发项目的失败。

在构建软件时，应用本身同样具有复杂性。大部分软件开发项目要求几个不同的开发者共同努力使该项目按时完成。这些关系意味着为了一致地、有效地使一个中等规模的系统概念化，我们需要一种能书写其结构并在几个不同的人之间交流该结构的机制。系统的这种表示方法可以降低它在我们头脑中的复杂性，同时方便相互协作人员之间的交流，包括某个领域的专家与最终用户之间的交流。请见要点栏1-1。

要点1-1 为什么开发软件是困难的

1) 应用领域复杂。
2) 不同的背景知识和不同的专业术语使人与人之间的有效交流变得困难。
3) 自然语言的含糊性。
4) 深刻领会大的开发项目的细节比较困难。

本章所提到的大部分内容涉及到软件开发项目的失效。关于**软件失效**（software failure）这个词汇本身，有些问题值得澄清。软件失效可以有许多原因。

- 软件开发过程能够导致没有功能的软件。
- 如果开发出有功能的软件，它也许没有充分地针对某些领域专家或用户的需要。该软件可能在很大程度上达不到预期的要求。
- 最终软件可能表面上满足了需求，但底层的运算也可能不正确。这种软件的运行结果可能会发生错误。
- 软件可能已满足了用户需求，其计算也是正确的，但可能由于用户的错误而产生故障。对用户来说，这种软件的使用方法可能并不直观，因此用户不知道如何正确使用它。
- 软件能够完成用户的一切需要，但由于响应时间太慢而失去了实用价值。

软件企业所关注的另一个主要问题是人们花在软件**维护**（maintenance）上的精力。一个成功开发出的软件将被长期应用。该软件要定时地进行修改以适应使用该软件的组织的需求变化，加强其功能，修改其缺陷。与开发新软件相比，人们花在软件维护上的精力是非常大的[9]。

由于软件系统的维护是不可避免的，实际上也是值得做的，因此建造软件的一个主要目的是减少维护所需的人力。使用模块化构件构建的结构良好的系统需要的维护精力就比较少，因为一组模块或单个模块可以在不影响系统其他功能的情况下被修改。随着修改的进行，新的错误来源充其量将被局限在少数几个模块。另外，如果软件用模块方式来书写，那么即使某种操作要在软件中多个地方执行，也只需写一段代码。当需要执行这种操作时可以从软件中各个地方调用。模块被修改后，整个系统发生了相同的改变。在系统中多个地方利用同一代码的这种行为被称为**代码重用**（code reuse）。代码重用的基本思想是开发通用模块使其在系统中充当多重角色。

图1-1展示了企业中一个典型的场景，即过多的技术人员从事结构不好的系统的维护工作，而留下从事新系统开发工作的人却不够。

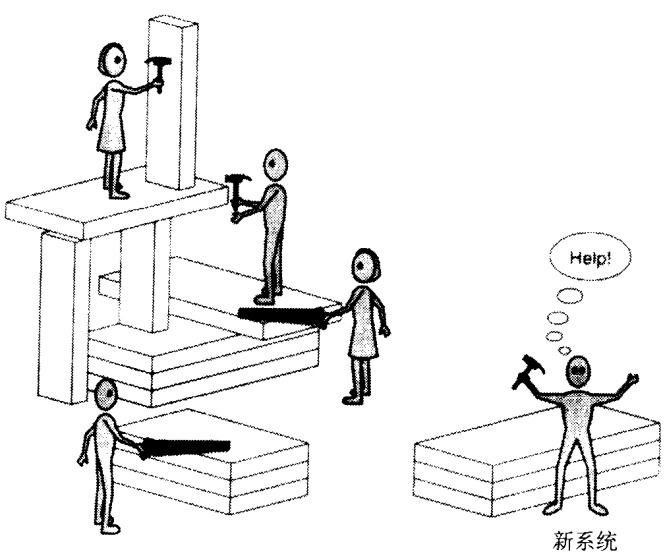

图1-1　失控的维护