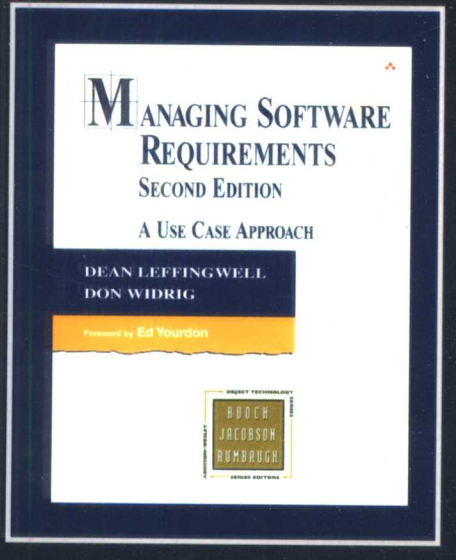
软 件 工 程 系 列

PEARSON
Addison Wesley

Managing Software Requirements
A Use Case Approach Second Edition

软件需求管理
用例方法
（第二版）

[美] Dean Leffingwell 著
Don Widrig
Ed Yourdon 序
蒋慧 译



- 软件权威、原 Rational 软件公司总经理作品
- 软件工程巨擘 Booch、Jacobson 大力推荐
- 最畅销软件需求管理图书的最新版本

Pearson Education
中国电力出版社
www.infopower.com.cn

软 件 工 程 系 列

Managing Software Requirements
A Use Case Approach Second Edition

软件需求管理
用例方法
（第二版）

[美] Dean Leffingwell
Don Widrig 著
Ed Yourdon 序
蒋慧 译

中国电力出版社
www.infopower.com.cn

Managing Software Requirements: A Use Case Approach, Second Editon (ISBN 0-321-12247-X)
Dean Leffingwell, Don Widrig

北京市版权局著作权合同登记号 图字：01-2004-2530

图书在版编目（CIP）数据
软件需求管理：用例方法（第 2 版）/（美）莱芬韦尔等著；蒋慧译. —北京：中国电力出版社，2004
（软件工程系列）
ISBN 7-5083-2190-1
Ⅰ.软... Ⅱ.①莱...②蒋... Ⅲ.软件开发 Ⅳ.TP311.52
中国版本图书馆 CIP 数据核字（2004）第 024598 号

从 书 名：软件工程系列
书　　名：软件需求管理：用例方法（第二版）
编　　著：（美）Dean Leffingwell　Don Widrig
翻　　译：蒋慧
责任编辑：姚贵胜
出版发行：中国电力出版社
　　地址：北京市三里河路6号　　邮政编码：100044
　　电话：（010）88515918　　传　　真：（010）88518169
印　　刷：北京丰源印刷厂
开　　本：787×1092　1/16　　印　　张：21.5　　字　　数：470 千字
书　　号：ISBN 7-5083-2190-1
版　　次：2004 年 5 月北京第 1 版　　2004 年 5 月第 1 次印刷
定　　价：35.00 元

序　言

Ed Yourdon

石头问题

我的一个学生把本书所讨论的问题总结为“石头”问题。她是某研究实验室的软件工程师，她的客户常常会交给她一些项目，她称之为“给我一块石头”。可是当你把石头交给客户时，他们会看着“石头”一会儿，然后说：“是的，但是……实际上我想要的是一块小一点的蓝色石头”。而当你交出一块小一点的蓝色石头时，又会引发“一块圆的小一点的蓝色石头”的要求。

最终的结果可能是，客户一直都在想要一块小一点的蓝色大理石——或者，他也许根本不能 确定他到底要什么，一块小小的蓝色大理石——甚至是猫眼大小的蓝色大理石就已经足够满足他的需要了。而他会在你交出第一块（最大的）石头和第三块（蓝色的小）石子之间不断改变他的要求。

开发人员在与客户会谈时总会提这样的问题：“你想要它做什么？”当开发人员按照客户预先的需求，经过艰苦努力终于交出一块大石头却得不到客户的肯定时，总是感到非常沮丧；而客户也同样沮丧，因为即使他可能也发现很难说清自己真正的要求，他还是觉得自己已经讲得很明白，只是开发人员不理解罢了！

大多数实际的项目会涉及更复杂的情况，而不只是涉及到两个人。除了客户和开发人员外——他们当然有不同的头衔和名字——此外还有市场营销人员、测试和质量保证人员、产品经理、总经理，以及一大堆“涉众”（stakeholder），他们的日常工作将受到所开发的新系统的影响。

这些人都会为如何指定一个大家都能接受的“石头”而头疼，尤其在当今快节奏的充满竞争的商业世界，谁都没有时间争论一个 2 年期的“石头项目”，并且重头来过。我们必须使它在第一次就是正确的，同时为客户提供一个迭代的过程，在这个过程中使他们最终发现他们真正想要的石头。

对付石头这样有形的物理实体，已经很难了，而现在多数商业机构和政府部门都是“信息密集”型的，即使名义上他们是从事构建和销售“石头”的工作，也很有可能涉及计算机。即便与计算机无关，也有可能需要利用计算机来跟踪它们的电子商务“石头”销售、客户、对手、供应商等等使它在“石头”商务中保持竞争力的信息。而且，对于当今成千上万个业务是开发和销售软件产品的公司来说，其整个业务都集中在使他们的无形和抽象的产品成为他们的客户能购买、评估和应用的有形的石头。

软件系统天生就是无形、抽象、复杂的，并且，至少从理论上说它们是无限可改变的。

所以，如果客户对“石头系统”的需求从一开始就是含糊的，并且总认为随着时间的推移，自己能对细节进行澄清、改变和补充的话，那么，开发人员以及所有创建、测试、部署和维护人员很难在零时间内以零耗费完成系统开发。

事实是：当今，有一半以上的软件系统项目超出预算并且推迟进度，有25%~33%的项目在完成之前就被放弃了，并且耗费惊人。

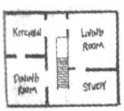
本书的目的是提供一种合理的方法来构造客户真正需要的系统，从而避免以上失败。但是，本书并不是一本有关程序设计的书，它也不仅仅是为软件开发人员写的。这是一本有关管理复杂软件应用系统的需求的书。因此，本书是为软件开发团队的所有成员（包括分析人员、开发人员、测试人员、质量保证人员、项目经理、文档经理以及文档管理人员等等），还有外部“客户”团队（用户以及其他的涉众，包括市场人员、管理人员）——所有对最终的需求有要求和需要的人而写的。

你会发现，让包括外部团队的非技术成员在内的所有团队成员，都掌握定义和管理需求所需要的技能，是非常关键的。原因很简单，因为是他们首先创建需求并且最终决定系统成功与否。让程序员英雄们孤军奋战是过去那个时代的错误，愿他们安息吧。

一个简单的比喻：盖房

如果你是一位建筑承包商，你一定认为有必要和房主进行一系列详细讨论，否则，你会造一座两居室的房子，而客户要求的却是三居室。但是，与负责建筑条例及地区规划的政府当局讨论和协商这些“需求”同样重要，假如要砍掉建房地皮上的树的话，你还得再和邻居商量一下。

建筑督察员、邻居以及要买房和住房的人都是涉众，他们将最后决定盖好的房子是否符合所有的需求。显然在这个系统中有很多非用户（房主）的重要涉众，如邻居和地区规划的官员，同样，他们对这个系统的看法将会有很大的不同。



再次强调，本书讨论的是软件应用，而不是房子或石头的问题。房子的需求（至少部分）可以用一组草图和工程制图来描述；类似地，也可以用模型和图来描述软件系统。但是，就像房子的草图是工程师和外行人员——包括律师、督察人员、邻居等之间进行协商和交流的手段一样，软件系统的技术图也可以以一种“普通人”能够理解的方式创建出来。

许多关键的重要需求根本不需要用图表示，比如，未来的购房客户可以简单地写道：“我的房子必须有三个卧室，还得有一个能容纳两部汽车和六辆自行车的车库。”在本书中，你也会看到，软件系统的多数重要需求都可以用自然语言来描述。在其他情况下，有一张关于户主想要壁炉的图可能更有用。

为了迎接这一挑战，你需要掌握一些重要的团队技能（team skill），有些技能可用实用的常识性建议表述。对于一个盖房新手，我们可能会建议他“一定要在挖地基之前，而不是在浇铸水泥并开始修墙和房顶后，和建筑督察人员谈谈”。而对于一个软件项目，我们也会有类似的建议：“务必向正确的人提正确的问题，务必理解系统将如何使用，不要假定100%的需求都是重要的，因为你不可能有时间在最后期限之前把它们全部完成。”

关于本书

在这本书中，Leffingwell 和 Widrig 采取了一种实用的方法来描述石头问题的解决方案。他们把全书分成八个部分。“引言”部分为后面的章节提供了环境、背景和定义；第 1 章回顾系统开发这一“挑战”。数据显示有些软件项目的失败是因为编程马虎造成的，但最近大量的研究确切地表明，不良的需求管理可能是项目失败的惟一的最主要原因。在这个序言里我用不严格的、非正式的方式描述了需求管理的基本概念，本书作者将在第 2 章详细地定义这一概念，从而为后续章节奠定基础。第 3 章概述了当今使用的一些软件开发模型，并在结论部分推荐了一种推动需求发现的迭代过程。第 4 章对现代软件团队的特性做了简明介绍，从而把要开发的技能与技能所要应用的团队环境联系起来。

本书根据有效的需求管理所必需的六个团队技能加以组织。

后面的六大部分是为了帮助你和你的团队理解掌握有效需求管理所需要的六大团队技能：

- 当然，在开始时，你要正确地理解新软件系统所要解决的问题。这是就团队技能之一，分析问题；
- 团队技能之二，理解用户和涉众的需要。这也是非常重要的，这些技能形成团队技能之二的基础；
- 团队技能之三，定义系统。描述了为解决这些需要而定义系统的最初过程；
- 团队技能之四，管理范围。讨论绝对重要但却常常被忽略的管理客户预期和项目范围的过程；
- 团队技能之五，细化系统定义。讨论为使团队确切了解要构造什么样的系统以进行设计和实现，而对系统定义进行细化时使用的关键技术；
- 团队技能之六，构建正确的系统。讨论构建满足需求的系统的过程。团队技能之六还讨论了确认系统是否符合需求、确保系统不会对用户有任何恶意、避免出现任何需求没有定义的不愉快行为的技术。同时，由于应用系统的重要需求不可能一成不变，因此作者还讨论了一些团队可用来积极地管理变更而不破坏已设计和构造的系统的方法。作者用章的篇幅总结团队技能之六，建议采用让需求收集过程提高整体项目质量的方法。这一章特别强调了现代程序开发过程的迭代特性以及它为不断的质量评估产生的许多机会。

在描述了这些特定需求管理技术之后，作者简单回顾了极限编程和敏捷方法的演化方法，并且展示了把有效需求管理实践集成到这些软件开发方法框架中的方法。最后，在第 31 章中作者提供了一个你和你的团队可以在下一个项目中使用的解决方案用于管理需求。

我希望，在这些新团队技术的武装下，你也能雕刻出完美的石头。但这个工作永远都不是易事，即使用最好的技术和过程并且有自动化工具的支持，你仍然会发现这是一件艰苦的工作。更进一步地说，这个工作风险很大，即便掌握了团队技能，有些项目仍然会失败，因为我们在为机构开发项目时总想冲开许多组织机构的面罩，企图在更短的时间内构造更复杂的系统。本书所给出的技能能够大大降低开发项目的风险，帮助你到达成功的彼岸。

第二版前言

Dean Leffingwell

从 1999 年本书第一版出版以来发生了很多事情。20 世纪 90 年代后期“dot.com”泡沫经济（部分是由互联网、软件和相关技术引起）的破灭，给许多人的生活带来很大的破坏、经济的不确定和混乱。然而，一个看起来“失去理智”很长时间的自由市场大概已经恢复了有序和稳健。

但是，软件技术的创新却没有衰退，软件业作为一个整体仍然成长迅速。遍及全球的互联网不断改变我们的生活，并驱动多种多样新的通信形式，从方便货物和服务交换的全球电子市场，到似乎占用孩子们过多家庭作业时间的课后即时消息聊天服务，这些服务也似乎占用了昂贵的过去十年大量铺设的互联网带宽。

我们以每周 7 天、每天 24 小时的方式与我们的商业合作者、朋友、家人保持联系。互联网网吧一天 24 小时开放，不管是在澳大利亚、苏格兰，还是在阿拉斯加边界上的巡逻舰上。我们在食品杂货店里自己的 PDA 上接收电子邮件。如果不和软件打交道，我们就不能做早餐，不能开车上班，不能乘电梯，或者不能进入办公大楼。软件已经成为世界上许多智力和知识的化身，开发和部署软件的业务已经显现为世界上最重要的一种产业。

软件开发的实践还在向前推进。统一建模语言（UML）于 1997 年被采纳，现在已经成为事实上的表达架构、模式和设计机制的手段。Rational 统一过程和类似基于 UML 的过程被业内许多人采纳作为应对软件开发挑战的标准方式。

我们的个人生活也改变了。在 Rational 软件公司工作四年后，我最近进入 IBM 转而帮助独立的软件公司达到他们的目标。有的团队希望改变世界；有的希望通过提高卫生保健来对个人生活产生显著影响；其他人则希望提高其客户的制造效率或希望通过把产品数据翻译成其他语言来帮助业务增长。但是，所有这些团队都有一个共同点：他们都在接受一个困难的挑战，即以一种可以被他们自己、他们的客户、他们的市场团队、他们的内部开发部门和测试团队所理解的方式定义软件解决方案——事实上，所有这些人必须在正确的详细程度上理解所建议的解决方案从而达到恰当的结果。做不到这个他们就完不成使命。由于他们的使命对于他们个人以及其产品要帮助的人们的生活的重要性，所以绝不能失败。

因此，尽管在短短几年内软件业的许多方面都改变了，但有的事情，包括管理软件需求的挑战，仍然大致相同。所以我们的工作在这个第二版中在继续。

关于第二版

改变第二版内容的动机是基于几个不同的但非常集中的因素。

第一组因素是基于本书在市场上的成功，我们得到了很多正面的评价和鼓励以及建设性的批评。评价涉及面很广，其中有两个非常一致的主题：

- “更多用例”主题。第一版（副标题为“统一方法”）把有关需求技术的两种主要观点结合起来。第一种，大概是更为传统的一种，描述了如何采用描述性技术（“系统应该……”）来创建和细化需求规格说明，从而指定系统行为。第二种，用例方法，描述了如何使用用例来定义系统的大部分功能性行为。我们在第一版中把这些技术结合到一起，是为了产生一种共同的、希望是更全面的方法。根据得到的反馈，在这一点上我们取得了一些成功。但是对此的批评在于，尽管我们推荐和描述了用例方法，却没有足够深入地帮助读者开发或应用这一技术。而且，为了同时说明这两种技术，我们还使一部分想了解什么时间采用什么技术的读者产生了疑惑。
- “这是一本有太多技术的巨著，能否更具有指导性”主题。本书第一版的意图是成为一本更全面的著作，为可能需要为任意类型的系统定义需求的任何技术读者提供“一站式”的参考手册。我们希望这为读者提供了帮助，因为我们坚信，对每个特定软件工程的挑战来说，绝对没有一个“包治百病”的解决方案。而且，读者的主题仍然是：“这真是这样困难吗？你能否更有指导性一些？”

第二组驱动同一种主题的因素来源于我自己为一些公司工作并帮助他们实现软件开发的目标时使用本书的经验。有的公司的软件应用系统需要多种技术；有的公司能够执行相当严格的需求管理制度；而另一些公司需要为特定软件应用系统记录特定的需求，并且在短时间内应用，比如明天。他们没有时间也没有兴趣争论关于什么技术最有效或者技术之间的细微差别，他们说：“给我一种最简单的技术，让我可以立刻开始工作。”

值得庆幸的是，这两组输入非常集中，答案也非常明确。对多数团队来说，在大多数情况下，下面的组合对于管理软件就是足够和适当的了：① 一个考虑深入的前景文档；②确定和细化要实现的关键用例；③ 对于非功能需求的补充规格说明。此外，如果可能成为一种选择的方法的话，细化的用例可以直接成为系统测试的基础。

到此为止，这本书的第二版就有了新的内容、新的主题和新的副标题：“用例方法”。在这一版，用例技术是基本技术，并且我们采用了一种更具指导性的方法和表示。例如，增加了第 14 章“用例入门”，为理解和应用用例奠定基础。这一章作为一个指南，使一个没有任何知识的读者能够学习并开始应用这一技术。我们对 HOLIS 个案研究也做了改动，反映出一种更以用例为中心的方法。另外我们还增加了第 26 章“从用例到测试用例”，来展示用例如何直接派生出完整的测试策略以及用例本身如何直接作为测试用例的输入。

此外，我们还纯粹为了一个想法做了一个实质性的提高。第 17 章（在第一版中是第 18 章“负责人”）已经被更名为“产品管理”，内容上也增加了新的材料，目的是帮助团队理解如何把一个软件应用系统变成我们所谓的“整体产品解决方案”。因为需求的“正确”本身不能保证商业的成功，所以这一章为这些活动（如定价、许可证、定位和宣传）以及其他把

一个有用的软件应用转变成一个人们想购买的软件产品的商业因素提供洞察和指南。

而且，因为现代软件开发过程的迭代性更强了，所以我们决定把第一版许多章节重新定位于质量。这一版的章节将在现代软件过程这一背景下，对质量进行更全面的观察。因此，第 29 章“在迭代开发过程中评估需求质量”，直接讨论在整个迭代开发的框架下，获取和提高需求的迭代技术。

最后，我们借机谈到了业界的一种新动态，向所谓的更轻（lighter）的、不太形式化的方法发展。最极端的是 Beck 等人主张的极限编程（Extreme Programing，XP），可以解释为完全排除过程。更准确一点，XP 合并了某些核心过程，如把客户需求直接放到到编程过程中，但要注意到这里有意回避“软件过程”和“M”（方法论）概念。大概不太极端并被一些人认为更实用的是 Cockburn 等人倡导并已经扎根的敏捷方法（Agile Methods）。虽然经过几轮争论，但不能忽略这几种轻型方法，我们将在需求这一背景下展开讨论，参见新增加的第 30 章“敏捷需求方法”。

当然，没有哪本书可以为所有人提供所有需要的信息。为了尽可能使这一版更具可读性，我们删除去了上一版的一些题目和章节，并缩短了某些章节的篇幅。

我们衷心希望这一版更容易理解、更容易使用和应用，从而更有助于你和你的团队管理你们的需求。

致谢

作者要感谢 John Altinbay、Jim Heumann 和 Dan Rawsthorne 细致深入地审读这个第二版。我们还要感谢其他许多对本书做出贡献的人，包括 Al Davis、Ed Yourdon、Grady Booch、Philippe Kruchten、Leslee Probasco、Ian Spence、Jean Bell、Walker Royce、Joe Marasco、Elemer Magaziner 以及以下第一版的审读人员：Ag Marsonia、Frank Armour、Ralph R. Young 博士，David Rine 教授和 Dan Rawsthorne。

我们承认如果没有他们富有洞察力的意见，我们不可能写出一本有价值的书。此外，我们还要感谢 Kim Arney Mulcahy 以及 Addison-Wesley 的其他编辑和支持人员，他们推进本书的出版过程，并使其成为一个真实的产品，正如序言中提到的“石头”。最后，我们还要感谢我们亲爱的家人，感谢他们理解我们为本书所付出的那些本应和他们分享的周末。

第一版前言

Dean Leffingwell 1999

背景和感谢

本书所传达的知识代表了一群优秀的业内人士所积累的宝贵经验，他们曾经定义、开发和交付了许多世界一流的软件系统。本书并不是一本关于需求工程的学术讨论。20 世纪 80 年代，Don Widrig 和我是一家小公司的业务主管，为客户提供软件帮助他们解决问题。当我们积累了一些本书所介绍的需求管理经验时，我们的着眼点在于既保证所交付软件系统的质量又保证客户的利益。因为所交付软件的性能对于这个商业投机本身的成功至关重要，所以我们不鼓励在其中掺杂些许成见、个人喜好，或采用不成熟的技术进行实验。

过去十年，技术在不断演化并且不断由新的经验增强，被不同公司的专家在不同的环境下加以扩展。但本书提及的所有技术都经过现实社会的证明并经受了时间的考验。更为重要的是，大概还在于它们还经受住了这一时期行业内的技术变革。事实上，本书中的许多原理都不受软件技术潮流变化的影响，因此我们希望这里所表述的知识能具有更长久的利用价值。

为其他行业制作软件得到的需求经验和教训

一开始，我讨厌计算机。（“什么？我必须整晚呆在这，把这批工作再做一遍只是因为我漏了一个“空格”？你疯了吗？让我在呆在那个房间里……”。）我第一台“真正的计算机”是一个小型机（minicomputer），尽管按照今天的标准，它的性能低得无法想像，但它却是惟一我可以接触、编程的机器，我可以让它做任何我想让它做的事，它是我的。

我最早的研究是用计算机分析来自人体的生理信号，主要是 EKG，对于这项工作，计算机是一个奇妙的工具。此后，我开始把我的程序设计技能和经验应用到工业的实时软件系统中。

最后，我组建了 RELA 股份有限公司，开始了一个漫长而异常艰难的职业：软件开发承包商的首席执行官。本书的另一位作者 Don Widrig 也加入了我的 RELA，最早是研究和开发部的副总经理，他为我们许多系统的成功做出了重要贡献。

随后几年，公司蓬勃发展，今天，公司已经有几百名员工，业务不仅包括提供软件，还提供某种意义上完整的医疗设备和系统，包括机械、电子、光学和射流处理子系统。在每台

机器的核心，包括最新临床诊断实验室的DNA识别仪，都有一台以上的计算机通过一个实时的多任务处理系统的调节，可靠而例行地传递出患者稳定的心律。

开始，我们为任何人编任何程序，从天线定位软件到激光标签游戏，为娱乐公司做自动引导汽车，还有教育产品、焊接机器人以及自动机械控制。我们还开发了一个大的分布式计算机系统，能够自动检测和计算电视上广告片的数量（那时，我们的口号是“让计算机来看广告片，把你解放出来！”）。我们做的所有软件的共同点大概在于，我们是为别人开发的——我们不是这些领域的专家，所以我们无法突破自己的不足。我们完全依靠客户的满意度来决定产品最终是否成功。在许多方面，这种环境非常有助于培养有效的需求管理。原因在于：

- 我们对这些领域了解得很少，因此我们要依靠客户提出需求。我们必须学会在正确的时间，用正确的方式提出正确的问题；
- 客户通常对计算机知道得很少，所以他们依靠我们来把他们的需要和希望解释成技术需求；
- 开发人员和客户之间泾渭分明，皆因为金钱作祟；
- 质量很容易度量：你要么有所收获，要么一文不名。

第一个重要问题：“这个软件到底要做什么？”

就是在这种环境下，我们发现了软件开发人员在所有项目中都会面对两个最基本问题。这两个问题主宰我们的行为很多年，并且今天可能仍然是所有软件项目中最难回答的问题。第一个问题：“这个软件到底要做什么？”

团队技能之一，分析问题；团队技能之二，理解用户和涉众的需要；团队技能之三，定义系统；这些部分提到的原理和技术已经有十多年的历史了，它们就是用来回答这一重要问题的。这些技术本身证明了它们自身的价值，并且在许多实际的项目中显示了它们的有效性。也正是在这期间，我第一次注意到Donald Gause和Jerry Weinberg的著作，尤其是他们的《*Exploring Requirements: Quality Before Design*》（1989年）。因为他们的著作很大程度地影响了我们的工作，所以本书中许多概念也取自这本著作，不仅因为这些概念的确有用，而且我们认为你也能分享Gause和Weinberg的成果才算公平。

构造高质量系统得到的经验和教训

随着时间的推移，RELA逐渐在开发基于计算机的医疗设备和系统方面走向专业化：手提式通风机（呼吸机）、注射泵、起搏器编程器、临床诊断系统、血泵、病人监视设备以及其他许多诊断和治疗设备。

正是在呼吸机项目的开发中，才使我们为自己正在做的工作而震惊：哇，如果我们把它拧紧点的话，有人就会没命了！我们主要关注的是与这台仪器紧紧系在一起的病人以及病人家属，这台仪器是我们做的也是世界上最早的时限最严的资源受限制的软件。（想像一下α测试和β测试时我们所接受的考验吧！）

显然，这种高风险的努力使我们在嵌入式系统业发展的早期就非常注重软件的质量。很快我们就明白，持续的成功需要以下几方面的结合：

- 一个能够定义和管理软件需求的实用的过程；
- 一个用于设计和开发软件的具体的方法；
- 多种经过证明的验证和确认软件是安全和高效的创新技术的应用程序；
- 软件开发和软件质量确保团队更多的技能和责任。

那时我非常相信甚至今天也十分确信，要交付任何具有一定广度、可靠的软件系统，这些元素都是必需的。在 RELA 公司，只有通过这种方式我们才能确保每个病人的生命安全，我们的公司才能赖以生存，公司的员工以及他们的家庭才有了经济保证。

第二个重要问题："我们怎么才能知道软件完成了所要求的工作，而不是其他工作?"

过去我们成功地开发和应用了多种回答第一个重要问题的技术，现在我们来看一下全世界软件开发团队所面临的第二个基本问题："我们怎么才能知道软件完成了所要求的工作，而不是其他工作？"

回答这个问题的技术组成了团队技能之五（细化系统定义）和团队技能之六（构建正确的系统）的基础。

可以确信的是，本书给出的技术都是稳定的并经过证明的。而且，即便不是开发安全性要求非常高的系统，这些技术也提供了非常有效、实用和划算的建议，可以用于开发高质量的软件系统。

尽管我们在 RELA 公司用来解决这两个重要问题时所借用、修改、开发和应用的技术非常有效，但必须承认，在项目的某些关键时期还是有不确定性因素：

"由于需求高度人为化的特点，所以多久我们会出一个存在潜在危险的错误？"

当然也有成本的问题，因为手工验证和确认费用昂贵并且容易出错。在这一期间，机械工程学科已超越了机械制图手，成为了一个 3D 计算机辅助设计系统。同时，软件的进步是有限的，在程序设计语言中为达到实用的目的增加了抽象的层次：从某种意义上是一件好事，但故障率、代码行生产率因子以及质量度量却相对不变。这一时期我们使用 CASE 工具的实验得到了一些混合的结果。坦率地说，作为一个软件工程师和软件企业家，我认为"软件工程"的现状令人尴尬。

尽管自动化永远不能替代软件开发中的思考技能，但我确信，把过程中的手工记录、变更管理等内容自动化可以解放宝贵的资源，转而进行具有更高价值的活动。当然，我们希望开发的成本更低而可靠性更高!

需求管理业务中得到的教训

REQUISITE

所以，Requisite 公司在 1993 年诞生了，许多人都参与开发和推销一种新的需求管理工具：RequisitePro。在这期间，随着我们不断地帮助客户解决他们的需求管理问题，产生了本书中许多增补材料。为此，我们要向我们的客户以及 RELA 的客户表示感谢，是他们从本质上教会了我们在这个主题上知道的任何事情。

我职业生涯的这一时期受到 Alan Davis 博士的很大影响，他是《*IEEE Software*》杂志的主编，是由 El Pomar 基金资助的位于 Calorado Springs 的科罗拉多大学的软件工程专业的教

授。Alan 早期作为董事和顾问加盟公司，对公司的技术和商业方向起着重要的作用。另外，他以其在需求工程领域的领导才能而著名，从事这方面的咨询活动并开发了帮助公司提高需求管理的大量技术。这些技术与 RELA 开发的一些技术合并成为一项职业培训课程——“需求学院（Requirements College）”的基础，也是本书这部分的基础。

此外，在一种目前还不太流行的商业理论——“永远都不可能有太多的职业帮助”的指导下，我们还吸收了著名的软件著作家（同时也是专家）Ed Yourdon 加入公司的董事会。Ed 对指导公司的技术和商业方向上也具有重要的影响。Ed 和 Alan 都是本书的早期特约撰稿人，本书中使用的许多词汇都是他们创造的。事实上，我们本想在几年前联合出版这本书的，虽然时间推迟了，但 Ed 和 Alan 都把他们的工作无私地贡献给我们。希望通过这寥寥数语，读者将会经常想起他们。

在 Rational 软件公司的经验

Rational unifying software teams

Rational 软件公司于 1997 年收购了 Requisite 公司。在 Rational 公司，我们在把需求管理应用于开发和发布一组完整的应用开发工具并继续帮助客户解决需求问题的过程中，又获取了很多重要的经验。Don 继续和我们一起奋战，帮助改进技术。此外，在 Rational，我还有幸和一些著名的软件专家和作家一起工作，包括 Grady Booch、Ivar Jacobson、James Rumbaugh、Walker Royce 和 Philippe Kruchten 等，他们对我形成有关需求管理的想法都有重要的影响，Walker 和 Philippe 最早还对本书进行了审阅。

我们也开始使用用例技术来获取需求，并在设计模型中利用用例概念来提供一个共同的线索以驱动架构、实现和测试。

我也是 Rational 的迭代开发方法的忠实宣传者，它是一种完整的生命周期软件开发过程，我认为我们在 RELA 以及 Rational 统一过程中都实践了这种方法。

Rational 帮助我完成了这本书，我表示衷心地感谢，并且 Rational 还慷慨地允许我使用它的一些重要思想、文本和图。

小结

本书中的思想即便有也很少是我们自己创造的。事实上，它代表了 20 年来软件开发经验的集成，并且以需求的挑战为集中、连贯和度量的重点。这样做，我们希望本书能融合业界中有关需求这一特殊和困难的挑战中最好的经验和思想。我们坚定地相信，本书的成果——高效地管理需求所要求的六大团队技能——将有助于你在预算内按时提交高质量的软件系统。

目 录

第二部分　团队技能之一——分析问题

第三部分　团队技能之二——理解用户和涉众的需要

第四部分　团队技能之三——定义系统

第五部分 团队技能之四——管理范围