目　　录

前　言

尽管拥有五十年积累的经验，但许多软件开发组织仍不得不在收集、编写和管理产品需求中疲于奔命。而缺乏用户参与、不完整的需求及不断变更需求，是导致信息技术项目不能按进度安排和资金预算完成全部功能的主要原因(The CHAOS Report,The Standish Group International.Inc.,1995)。许多软件开发人员不能熟练地收集客户（customer）需求，很多开发者并不知道实用的需求工程技术，而且教学课程中也是技术主题比需求主题占有优势，工程参与者甚至连“需求”是什么也有不同的看法。

软件开发中，信息沟通（交流）至少应与计算占有同等的比重，然而我们往往强调了计算而忽略了信息沟通。本书提供的许多工具将有助于信息交流，同时将帮助软件专业人员、管理者、市场营销者以及客户能应用有效的需求工程方法。本书还介绍了许多方法，用来帮助开发小组和客户一致理解怎样构造一个软件才能满足用户（user）实际的需要，同时也包括了编写文档和管理变更的方法。本书中介绍的技术都代表着需求工程中主流的“良好的习惯做法”，而并非来源于专业领域的高新技术或试图解决所有需求问题的复杂的方法学。

本书有益于读者之处 Top

本书对你着手的所有软件过程改进，对改善需求开发和管理实践都能提供很多的益处。本书是介绍概念和方法的，并不涉及专门的研究方法学或应用领域，所以它适用于各种项目。我尽力以清晰的结构风格介绍大量实用的且经过验证的技术，希望在以下几个方面能给你提供帮助：

- 达到实现更高的客户满意度。
- 减少维护和支持费用。
- 在开发周期早期提高项目需求分析的质量，减少重复劳动，从而提高生产率。
- 通过控制项目范围的扩展(creep)及需求变更，来达到按计划完成预定目标。

我的目标是帮助你改进收集和分析需求、编写和修改需求规格说明以及在整个产品开发周期中管理需求。改进过程的最终目的是使你组织中的人员以新的方式进行工作，从而获得更好的结果。因此，希望你能将所学用于实践，而不仅是“纸上谈兵”。

实例研究

为帮助读者理解怎样应用本书介绍的各种方法，书中提供了几个基于实际项目的实例，其中一个中等规模的信息系统——“化学制品跟踪系统”说明了许多实用技术（不必担忧——你勿需知道任何关于化学的知识也能理解该项目）。这个实例的项目说明还会帮助你了解怎样把不同的策略（方法）较好地融合到一块。本书中还穿插着源于该实例的项目参与者之间的对话实例。不论你的小组是开发什么软件的，这些对话总是常见的，也是类似的。

谁应该读这本书 Top

凡参与一个新的或升级的软件产品的需求定义或说明中的任何人员都能从本书中获得有用的信息。这些人中包括那些必须理解并满足用户期望的分析、开发、测试人员；也包括用户，这些用户想定义一种符合他们功能和使用需要的产品。希望确认产品是否满足业务需要的客户将能更好地理

解需求分析过程的重要性。而负责监督按期完成产品的项目管理者也将学会怎样管理潜在的威胁——需求变更。

在许多训练性讨论会中，我发现那些非技术方面的项目参与者也很容易理解本书所涉及的内容。想提高自己对开发过程的理解和想了解需求在软件成功中扮演的关键角色的任何人都将从本书中受益。

本书结构 Top

本书分为三大部分。第一部分先介绍了一些基本的需求工程定义和一些优秀的需求分析所具有的特性。我希望你与你的重要客户能一起阅读第 2 章（关于客户与开发者之间的伙伴关系）；第 3 章介绍了许多需求开发和管理的改进熟练程度的好方法（良好的习惯做法）。第 4 章有助于计划怎样将新的策略融入小组的开发过程中。方法是基于对附录中当前需求实践自我测试的回答。第 5 章则介绍了一些通常与需求相关的项目风险。

第二部分介绍了许多关于需求开发的技术。首先是定义项目的业务需求，项目视图(wision)及涉及的范围(scope)。接下来的章节介绍怎样为项目寻找合适的客户代表，获取（elicit）用户需求，编写功能需求文档及质量属性文档。第 10 章介绍了一些分析模型，这些模型可用于不同范围的需求分析。第 12 章介绍了软件原型的结构和应用。第二部分中的其它章节还探讨了定义需求优先级的方法及验证需求分析是否正确的方法。

第三部分的主题是需求管理的原则和策略。这部分还特别介绍了处理需求变更和评价每项变更对项目影响的技术。第 18 章介绍了怎样把需求跟踪能力和单个需求相关的内容需求来源到与需求相关的设计、代码等）联系起来。第三部分的内容包括一些商业工具的说明，这些工具能增强你管理项目需求的能力。

从原理到实践

要克服障碍，更改旧的传统做法，把新的知识付诸实践的确不是一件容易的事情。你也许仍想保持原有熟悉（可能并不很有效）的方法，为了帮助你改进需求分析，本书的每一章都包括一个称作“下一步”的内容，它细致地教你如何将本章所学的知识真正开始应用于实践。我提供了许多带有详细注释的模板，包括：需求分析文档、审查校对清单列表、需求优先级电子表格等，所有这些都放在我的网站：http：//www.processimpact.com 上，希望能帮助你尽快把这些技术应用于实践。请从今天就开始，一点一点地逐渐改进你的需求分析。

一些项目参与者在尝试新需求技术时是很勉强的。因为有一些人完全不讲理，而与这些不理解的人合作，所有新技术都是没用的。因此将本书介绍的知识告诉你的同事、客户和管理者，用你们以前项目中遇到的与需求有关的问题或困难来提醒他们，与他们一起讨论这些新技术拥有的潜在优势，共同学习、共同提高。

你不一定非要在一个新项目中开始应用这种改进的需求工程方法。其实就地改变控制过程就是很好的开端。也就是说，你可以从管理需求变更的请求开始，采用这种比过去更好的方法。因为你能开始作系统的影响分析，创建跟踪能力矩阵，从而把新的需求与相应的设计、代码、测试用例都联系起来了。为现存系统回头重新编写整个系统的需求规格说明是不现实的。但你可以写一个更加结构化的新版本，作一些新特点的分析模型，并调查新的需求。逐渐实施改进的需求，其风险较低，并且也可以为你将新技术应用于下一个主要项目奠定基础。

需求工程的目标是做出高质量而并非完美的需求分析，这使得你能在一个可接受的风险限度上实施。为了把返工重做、不被接收的产品及超期未完成这些风险降低到最小程度，你必须花大量的时间来研究需求工程。而本书将有助于确定何时能达到合适的需求分析质量标准，同时还介绍了具体实施方法。

致谢 Top

我深深地感谢那些花费时间、精力审阅我的手稿并提供了许多改进建议的人们。特别要感谢凯茜·弗德，他细致的检查，为我的思考和介绍提供了许多珍贵而深入的想法。我真诚地感谢克瑞丝·凡巴斯，汤米·汉格森，蒂彭·莫勒，麦克·瑞巴克，菲尔·瑞克其，约翰·罗丝曼，路易丝·斯塔茨，多瑞丝·斯芬伯格，布兰卡·哈帕雅，苏格特·瑞特曼，他们为每一章节所作的极耗时间的评注。我也很感激那些为某些章节做出努力的人：斯蒂夫·安道夫，迪克·巴尔科，比尔·坦格，德尔·爱莫瑞，基尔夫·弗兰默克，琳达·弗勒明，凯丝·格茨，吉姆·哈特，迈克·梅尔克，他们找出了草稿中的许多错误，而所余下的任何错误都完全是我的疏漏所至。

我还要感谢斯蒂夫·迈克奈尔，正是他鼓励我写一本软件需求书。还有我的熟人微软出版社的编辑本·瑞思，他帮我拓展了研究途径并与外界建立了良好的工作关系。玛丽·凯本蒂·巴纳德在米切尔·古德曼的协助下管理整个过程，并将初稿精心编辑，最终定稿。美术家罗伯·耐斯把许多先前的草图绘制成清晰的表格和图形，排版员波纳·格里克把它插入到书中。

我非常感激我的客户顾问，特别是斯迪·布洛林，迈特·德沙斯，凯丝·弗德，凯丝·沃勒斯，他们邀请我参与到他们的需求分析过程中`，这对我来说既是指导，又是学习。我特别要感激凯丝·弗德愿意将她的经验在本书中予以讨论。还要感谢罗宾·古德斯密斯提供的业务需求分析想法以及迈特·德沙斯提供的一些典型软件开发工程的极好的术语， 例如：“快速缩小范围阶段”（rapid descoping phase）。本书中介绍的一些技术将它改称为：“受控的缩小范围阶段”（controlled descoping phase）。

在过去几年中，参与我需求分析讨论会的上百参与者给与的反馈和贡献是相当有益的，作为一个学者和顾问，我从我工作过的每一个公司中都学到了很多有用的东西，并从讨论参与者的经验交流中也收获颇丰。那些把这些技术应用并发崛其价值的人们所给与的评价，使我更确信：这是项目需求分析所需的实用方法。如果你认为这能有助于工作或不能，请你告诉我：kwiegers@acm.org。

最后，我将最深的谢意献给我那最富耐心，给予我莫大支持并使我生活充满快乐的妻子——克瑞丝·扎彼得。有这样一位妻子是任何作者梦寐以求的。

第一部分　软件需求：是什么和为什么

第一章　基本的软件需求 Top

“喂，是 Phil 吗？我是人力资源部的 Maria，我们在使用你编写的职员系统时遇到一个问题，就是一个职员想把她的名字改成 Sparkle Starlight，而系统不允许，你能帮帮忙吗？”

“她嫁给了一个姓 Starlight 的人吗？”Phil 问道。

“不，她没有结婚，而仅仅是要更改她的名字，”Maria 回答道。“就是这问题，好像我们只能在婚姻状况改变时才能更改姓名。”

“当然是这样，我从没想过谁会莫名其妙地更改自己的姓名。我记不起你曾告诉我系统需要处理这样的事情，这就是为什么你们只能在改变婚姻状况对话框中才能进入更改姓名的对话框。”Phil 说道。

Maria 说：“我想你当然知道每个人只要愿意都可以随时合法更改他（她）们的姓名。但不管怎样，我们希望在下周五之前解决这个不合理的地方，否则，Sparkle 将不能支付她的账单。你能在此前修改好这个错误吗？“

“这并非是个错误！我从来不知道你需要处理这种情况。我现在正忙着做一个新的性能检测系统，并且我还要处理职员系统的一些需求变更请求”（传来翻阅稿纸的声音）。“哦，这儿还有别的事。我只可能在月底前修改好，一周内不行，很抱歉。下次若有类似情况，请早一些告诉我并把它们写下来。”

“那我怎么跟 Sparkle 说呢？”Maria 追问道，“如果她不能支付账单，那她只能挂帐了。”

“Maria，你要明白，这不是我的过错。”Phil 坚持道，“如果你一开始就告诉我，你要能随时改变某个人的名字，那这些都不会发生。因此你不能因未猜出你的想法（需求）就责备我。”

Maria 不得不很愤怒地屈从道：“好吧，好吧，这种烦人的事使我恨死计算机系统了。等你修改好了，马上打电话告诉我，行吧？”

如果你曾作为客户有过类似的对话，你一定知道：一个不能让你进行一项基本操作的软件产品是多么令人烦恼。你不会感谢开发者，尽管他最终会满足你主要的需求变更。但从开发者角度，你也知道，当用户在整个系统已经完成后，再提出一些功能要求是多么烦人的事。同时，由于修改系统的请求会打断你当前的项目，也是令人很不愉快的。而且往往这种修改请求还要求你优先处理。

其实，在软件开发中遇到的许多问题，都是由于收集、编写、协商、修改产品需求过程中的手续和作法（方法）失误所带来的。例如上面的 Phil 和 Maria，出现的问题涉及到非正式信息的收集，未确定的或不明确的功能，未发现或未经交流的假设，不完善的需求文档，以及突发的需求变更过程。

对大多数人来说，若要建一幢 20 万美元的房子，他一定会与建房者详细讨论各种细节，他们都明白以后的修改会带来损失，以及变更细节的危害性。然而，到软件开发，人们却变得“大大咧咧”起来。软件项目中百分之四十至百分之六十的问题都是在需求分析阶段埋下的“祸根”（Leffingwell 1997）。可许多组织仍在那些基本的项目功能上采用一些不合规范的方法，这样导

致的后果便是一条鸿沟（期望差异）——开发者所想开发的与用户所想得到的存在着巨大期望差异。

在软件工程中，所有的风险承担者（stakeholder）都感兴趣的就是需求分析阶段。这些风险承担者包括客户、用户、业务或需求分析员（负责收集客户需求并编写文档，以及负责客户与开发机构之间联系沟通的人）、开发人员、测试人员、用户文档编写者、项目管理者和客户管理者。这部分工作若处理好了，能开发出很出色的产品，同时会使客户感到满意，开发者也倍感满足、充实。若处理不好，则会导致误解、挫折、障碍以及潜在质量和业务价值上的威胁。因为需求分析奠定了软件工程和项目管理的基础，所以所有风险承担者最好是采用下面提供的有效的需求分析过程。

本章将帮助你：

- 了解软件需求开发中使用的一些关键名词。
- 警惕在软件项目中可能出现的与需求相关的一些问题。
- 知道优秀的需求规格说明应该具有的特点。
- 明白需求开发与需求管理之间的区别。

软件需求的定义 Top

软件产业存在的一个问题就是缺乏统一定义的名词术语来描述我们的工作。客户所定义的“需求”对开发者似乎是一个较高层次的产品概念。而开发人员所说的“需求”对用户来说又像是详细设计了。实际上，软件需求包含着多个层次，不同层次是从不同角度与不同程度反映着细节问题。

> IEEE 软件工程标准词汇表（1997 年）中定义需求为：
> （1） 用户解决问题或达到目标所需的条件或权能（Capability）。
> （2） 系统或系统部件要满足合同、标准、规范或其它正式规定文档所需具有的条件或权能。
> （3） 一种反映上面（1）或（2）所描述的条件或权能的文档说明。

一些关于“需求”的解释

IEEE 公布的定义包括从用户角度（系统的外部行为），以及从开发者角度（一些内部特性）来阐述需求。

一个很关键的问题是一定要编写需求文档。我曾经目睹过一个项目中途更换了几乎所有的开发者，客户被迫又与新的需求分析者坐到一起。分析人员说：“我们想与你谈谈你的需求。”客户的第一反应便是：“我已经将我的要求都告诉你们前任了，就是给我编一个系统”。而实际上，需求并未编写成文档，因此新的分析人员不得不从头做起。所以事实上你只有一堆邮件、贴条、会谈过几次或一些零碎的对话，要是你确信你已明白需求，那完全是在自欺欺人。

另外一种定义认为需求是“用户所需要的并能触发一个程序或系统开发工作的说明”（Jones 1994）。需求分析专家 Alan Davis （1993）拓展了这个概念：“从系统外部能发现系统所具有的满足于用户的特点、功能、属性等”。这些定义强调的都是产品是什么样的，而并非产品是怎样设

计、构造的。而下面的定义则从用户需要进一步转移到了系统特性（Sommerville 和 Sawyer 1997）：

需求是……指明必须实现什么样的规格说明。它描述了系统的行为、特性或属性，是在开发过程中对系统的约束。

从上面这些不同形式的定义不难发现：并没有一个清晰、毫无二义性的“需求”术语存在，真正的“需求”实际上在人们的脑海中。任何文档形式的需求（例如：需求规格说明）仅是一个模型，一种叙述（Lawrence 1998）。我们需要确保所有项目风险承担者在描述需求的那些名词的理解上务必达成共识。

需求的层次 Top

下面这些定义是需求工程领域中常见术语的定义说明。

软件需求包括三个不同的层次——业务需求、用户需求和功能需求——也包括非功能需求。业务需求（business requirement）反映了组织机构或客户对系统、产品高层次的目标要求，它们在项目视图与范围文档中予以说明。用户需求(user requirement) 文档描述了用户使用产品必须要完成的任务，这在使用实例（use case）文档或方案脚本（scenario）说明中予以说明。功能需求(functional requirement)定义了开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足了业务需求。所谓特性(feature)是指逻辑上相关的功能需求的集合，给用户提供处理能力并满足业务需求。软件需求各组成部分之间的关系如图 1-1 所示。

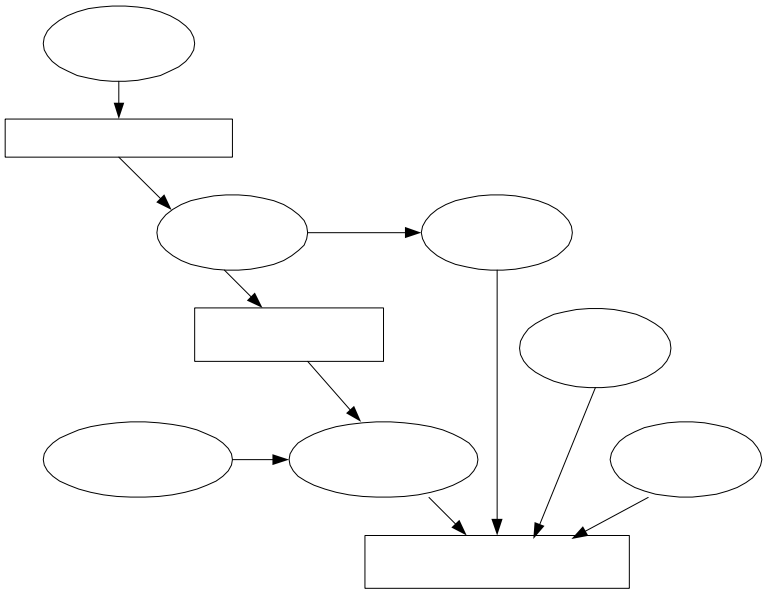

图1-1 软件需求各组成部分之间的关系

在软件需求规格说明（software requirements specification 简称“SRS”）中说明的功能需求充分描述了软件系统所应具有的外部行为。软件需求规格说明在开发、测试、质量保证、项目管理以及相关项目功能中都起了重要的作用。对一个复杂产品来说，软件功能需求也许只是系统需求的一个子集，因为另外一些可能属于软件部件。

作为功能需求的补充，软件需求规格说明还应包括非功能需求，它描述了系统展现给用户的行为和执行的操作等。 它包括产品必须遵从的标准、规范和合约；外部界面的具体细节；性能要求；设计或实现的约束条件及质量属性。所谓约束是指对开发人员在软件产品设计和构造上所具有的选择限制。质量属性是通过多种角度对产品的特点进行描述，从而反映产品功能，多角度描述产品对用户和开发人员都极为重要。

下面以一个字处理程序为例来说明需求的不同种类。业务需求可能是：“用户能有效地纠正文档中的拼写错误”，该产品的包装盒封面上可能会标明这个满足业务需求的拼写检查器。而对应的用户需求可能是“找出文档中的拼写错误并通过一个提供的替换项列表来供选择替换拼错的词”。同时，该拼写检查器还有许多功能需求，如找到并高亮度提示错词的操作；显示提供替换词的对话框以及实现整个文档范围的替换。

管理人员或市场分析人员会确定软件的业务需求，这有利于公司操作更加高效（对信息系统而言）或具有很强的市场竞争力（对商业软件产品而言）。所有的用户需求必须与业务需求一致。用户需求允许需求分析者能从中得到一些功能需求以满足用户使用产品来完成其任务，而开发人员则利用功能需求来设计如何实现必须的功能。

从以上定义可以发现，需求并未包括设计细节、实现细节、项目计划信息或测试信息。需求与这些没有关系，它关注的是充分说明你究竟想开发什么。项目也有其它方面的需求，如开发环境需求或发布产品及移植到支撑环境的需求。尽管这些需求对项目成功也至关重要，但它们并非本书所要讨论的。

每个项目都有需求 Top

Frederick Brooks 在他 1987 年的经典的文章“No Silver Bullet：Essence and Accidents of Software Engineering ”中充分说明了需求过程在软件项目中扮演的重要角色：

开发软件系统最为困难的部分就是准确说明开发什么。最为困难的概念性工作便是编写出详细技术需求，这包括所有面向用户、面向机器和其它软件系统的接口。同时这也是一旦做错，将会最终给系统带来极大损害的部分，而且以后再对它进行修改也极为困难。

每个软件产品都是为了使其用户能以某种方式改善他们的生活，于是，花在了解他们需要上的时间便是使项目成功的一种高层次的投资。这对于商业最终用户应用程序，企业信息系统和软件作为一个大系统的某一部分的产品是显而易见的。但是对于我们开发人员来说，并没有编写出客户认可的需求文档，我们如何知道项目于何时结束？而如果我们不知道什么对客户来说很重要，那我们又如何能使客户感到满意呢？

然而，即便并非出于商业目的的软件需求也是必须的。例如软件库、组件和工具这些供一个开发小组内部使用的软件。当然你可能偶然地勿需文档说明就能与其他人意见接近一致，但更常见的是出现重复返工这种不可避免的后果，而重新编制代码的代价远远超过重写一份需求文档的代价。

近来，我遇到一个开发小组开发包括流程图工具和源代码编辑器在内的一套计算机辅助软件工程工具。不幸的是，在他们开发完这个工具后，发现这个工具不能打印出源代码文件，而用户当然希望有这个功能。结果这个小组只好手工抄写源代码文档以供代码检查。这说明如果我们没有编写文档，那怕这种需求是明确无误和设想的，软件达不到期望目标也只是是咎由自取了。

相反的情况，我曾为一个要集成到“商业错误跟踪系统”中的简单电子邮件界面写了一页需求说明。而 Unix 系统管理员在为处理电子邮件写脚本时发现如此简单的一张需求清单竟是如此有用。我依据需求进行测试时，不仅非常清晰地实现了所有必需功能，而且未发现任何错误。

什么情况将会导致好的群体发生不合格的需求说明 Top

不重视需求过程的项目队伍将自食其果。需求工程中的缺陷将给项目成功带来极大风险，这里的“成功”是指推出的产品能以合理的价格、及时地完成并在功能、质量上完全满足用户的期望。下面将讨论一些需求风险。第 5 章将介绍怎样应用软件风险管理从而防止与需求有关的风险的出现和不适当的需求过程所引起的一些风险。

- 包含的用户数不多将导致无法接受的产品。
- 用户需求的扩展将带来过度的耗费和降低产品的质量。
- 模棱两可的需求说明可能导致时间浪费和返工。
- 用户增加一些不必要的特性和开发人员“画蛇添足(gold-plating)”的追求。
- 过分精简的需求说明以致遗漏某些关键需求。
- 忽略某一部分用户类的需求将导致众多客户的不满。
- 不完善的需求说明使得项目计划和跟踪等无法准确进行。

无足够用户参与

客户经常不明白为什么收集需求和确保需求质量需花费那么多功夫，开发人员可能也不重视用户参与。究其原因：一来因为与用户合作不如编写代码有兴趣；二来因为他们觉得已经明白用户的需求了。在某些情况下，与实际使用产品的用户直接接触很困难，而客户也不太明白用户的真正需求。但还是应让具有代表性的用户在项目早期直接参与到开发队伍中，并一同经历整个开发过程。

用户需求的不断扩展

在开发中若不断地补充需求，项目就越变越庞大以致超过其计划安排及预算范围。计划并不总是贴近项目需求规模与复杂性的实际情况、风险、开发生产率及需求变更情况，这使得问题更难解决。实际上，问题根源在于用户对需求的改变和开发者对新需求所作的修改。

要控制变更范围的不断扩展，必须一开始就对项目视图、范围、目标、约束限制和成功标准给予明确说明，并将此说明作为评价需求变更和新特性的参照框架。说明中包括了对每种变更进行变更影响因素分析的变更控制过程，这也将有助于所有风险承担者明白业务决策的合理性，为何进行某些变更，以及相应消耗的时间、资源或特性上的折中。

产品开发中不断延续的变更会使其整体结构日渐紊乱，补丁代码也使得整个程序难以理解和维护。插入补丁代码也使模块违背强内聚、松耦合的设计原则，特别是如果项目配置管理工作也不完善的话，收回变更和删除特性也会带来问题。如果你能尽早区别这些可能带来变更的特性，你就能开发一个更为健壮的结构，并能更好地适应它，这样设计阶段需求变更不会直接导致补丁代码，同时也有利于控制因变更导致质量的下降。

模棱两可的需求

模棱两可是需求规格说明中最为可怕的问题(Lawrence 1996)。它的一层含义是指诸多读者对需求说明产生了不同的理解；另一层含义是指单个读者能用不止一个方式来解释某个需求说明。

模棱两可的需求会使不同的风险承担者产生不同的期望，它会使开发人员为错误问题而浪费时间，并且使测试者与开发者所期望的也不一致。一位系统测试人员曾告诉我，她所在的测试组经常对需求理解有误，以致不得不重写许多测试用例和重做许多测试。

模棱两可的需求带来不可避免的后果便是返工——重做一些你认为已做好的事情。返工会耗费开发总费用的百分之四十，而百分之七十至八十五的重做是由于需求方面的错误所导致的（leffingwell 1997）。想像一下如果你能减少一半的返工会是怎样的情况？你能更快地开发出产品，在同样的时间内开发更多、更好的产品，或甚至能偶尔回家休息休息。

处理模棱两可需求的一种方法是组织好负责从不同角度审查需求的队伍。仅仅简单浏览一下需求文档是不能解决模棱两可问题的。如果不同的评审者从不同的角度对需求说明给予解释，但使得每个评审人员都有所了解，这样二义性就不会直到项目后期才被发现，那时才发现的话会使得更正代价很大。其它检测模棱两可的技术由 Gause 和 Weinberg（1989）给予了介绍，本章的后面也有所涉及。

不必要的特性

“画蛇添足”是指开发人员力图增加一些“用户欣赏”但需求规格说明中并未涉及的新功能性。经常发生的情况是用户并不认为这些功能性很有用，以致在其上耗费的努力“白搭”了。开发人员应当为客户构思方案和出于他们考虑的一些创新思路，具体确定哪些功能是要在客户所需与开发人员在允许时限内的技术可行性之间求得平衡，开发人员应努力使其简单易用，而不要未经客户同意，擅自脱离客户要求，自作主张。

同样，客户有时也可能要求一些看上去很“酷”，但缺乏实用价值的功能，而这些只能徒耗时间和成本。为了将“画蛇添足”的危害尽量减小，应确信：你明白为什么要包括这些功能，以及关于这些功能的“来龙去脉”，这样使得需求分析过程始终是注重那些能使用户完成他们业务任务的核心功能。

过于精简的规格说明

有时，客户并不明白需求分析有如此重要，于是只作一份精简之至的规格说明，仅涉及了产品概念上的内容，然后让开发人员在项目进展中去完善，结果很可能出现的是开发人员先建立产品的结构之后再完成需求说明。这种方法可能适合于尖端研究性的产品或需求本身就十分灵活的情况（Mc Connell 1996）。但在大多数情况下，这会给开发人员带来挫折（使他们在不正确的假设前提下和极其有限的指导下工作），也会给客户带来烦恼（他们无法得到他们所设想的产品）。

忽略了用户分类

大多数产品是由不同的人使用其不同的特性，使用频繁程度也有所差异，使用者受教育程度和经验水平也不尽相同。如果你不能在项目早期就针对所有这些主要用户进行分类的话，必然导致有的用户对产品感到失望。例如，菜单驱动操作对高级用户太低效了，但含义不清的命令和快捷键又会使不熟练的用户感到困难。

不准确的计划

“上述是我对新产品的看法，好，现在你能告诉我你什么时候能完成吗？”许多开发人员都遇到这种难题。对需求分析缺乏理解会导致过分乐观的估计，而当不可避免的超支发生时，会带来颇多麻烦。据报道，导致需求过程中软件成本估计极不准确的原因主要有以下五点：频繁的需求变更；遗漏的需求；与用户交流不够；质量低下的需求规格说明和不完善的需求分析（Davis 1995）。

对估计时间要求提问的正确响应是“等我真正明白你的需求时，我就会来告诉你”。基于不充分信息和未经深思的不成熟估计很容易被一些因素推迟。当要作出估计时，最好还是给出一个范围（如最好的情况下，很可能的，最坏情况下）或一个可信赖的程度（我有 90%的把握，我能在 8 周内完成）。通常未经准备的估计是作为一种猜测而给出的，听者却认为是一种承诺。因此我们要尽力给出可达到预期的期望并坚持一贯如此。

高质量的需求过程带来的好处 Top

实行有效的需求工程管理的组织能获得多方面的好处。也许最大的好处是在开发后期和整个维护阶段的返工重做大大减少了。Boehm（1981）发现要改正在产品付诸应用后所发现的一个需求方面的缺陷比在需求阶段改正这个错误多出 68 倍的成本。近来很多研究表明这个错误导致成本放大因子可以高达 200 倍这么高。强调需求质量并不能引起某些人的重视，他们错误地认为在需求上消耗多少时间就会导致产品开发推迟多少时间。从传统的质量成本角度分析来看，揭示了需求及其它早期质量工作的重要性（Wiegers 1996a）。

正确的需求过程强调产品开发中的通力合作，包括在整个项目过程中多方面风险承担者的积极努力。由于收集需求能使开发小组更好地了解其市场，而市场因素是任何项目成功的一个关键因素。在产品开发前了解这些比在遭到客户批评后才意识到要节约很多成本。让用户积极参与需求收集过程能使产品更富有吸引力，而且能拥有忠实的客户关系。通过了解用户的任务需求而不是仅仅局限于一些“华丽”的特性，你能避免在无用功能上白耗精力，并且用户的参与能弥补用户期望获得和开发者实际开发之间的“鸿沟（期望差异）”。

将选定系统的需求明确地分配到各软件子系统，强调采用产品工程的系统方法。这样能简化硬软件的集成，也能确保软硬件系统功能匹配适当。而有效的变更控制和影响分析过程也能降低需求变更带来的影响。最后，将需求编写成清晰、无二义性的文档将会极大的有利于系统测试，确保产品质量，以使所有风险承担者感到满意。

优秀需求具有的特性 Top

怎样才能把好的需求规格说明和有问题的需求规格说明区别开来？下面讨论单个需求陈述说明的几个特点（Davis 1993；IEEE 1998）。让风险承担者从不同角度对 SRS 需求说明进行认真评审，能很好地确定哪些需求确实是需要的。只要你在编写、评审需求时把这些特点记在心中，你会写出更好的（尽管并不十分完美）需求文档，同时你也会开发出更好的产品。在第 9 章中，我们将使用这些特点找到一些需求陈述中的问题并改进之。

需要说明的特征

完整性

每一项需求都必须将所要实现的功能描述清楚，以使开发人员获得设计和实现这些功能所需的所有必要信息。

正确性

每一项需求都必须准确地陈述其要开发出的功能性。判断正确的参考是需求的来源，如客户或高层的系统需求规格说明。若软件需求与对应的系统需求相抵触则是不正确的。只有用户代表才能确定用户需求的正确性，这就是为何一定要有用户的积极参与的原因。没有用户参与的需求评审将

导致类似说法：“那些毫无意义，这些才很可能是他们所要想的。”其实这完全是评审者凭空猜测。

可行性

每一项需求都必需是在已知系统和环境的权能和限制范围内可以实施的。为避免不可行的需求，最好在获取（elicitation）需求(收集需求)过程中始终有一位软件工程小组的组员与需求分析人员或考虑市场的人员在一起工作，由他来负责技术可行性上的检查。

必要性

每一项需求都应把客户真正所需要的和最终系统所需遵从的标准记录下来。“必要性”也可以理解为每项需求都是用来授权你编写文档的“根源”。要使每项需求都能回溯至某项客户的输入，如使用实例或别的来源。

划分优先级

给每项需求、特性或使用实例分配一个实施优先级以指明它在特定产品中所占的分量。如果把所有的需求都看作同样重要，那么项目管理者在开发或节省预算或调度中就丧失控制自由度。第 13 章将更详细地讨论如何划分优先级。

无二义性

对所有需求说明的读者都只能有一个明确统一的解释，由于自然语言极易导致二义性，所以尽量把每项需求用简洁明了的用户性的语言表达出来。避免二义性的有效方法包括对需求文档的正规审查，编写测试用例，开发原型以及设计特定的方案脚本。

可验证性

检查一下每项需求是否能通过设计测试用例或其它的验证方法，如用演示、检测等来确定产品是否确实按需求实现了。如果需求不可验证，则确定其实施是否正确就成为主观臆断，而非客观分析了。一份前后矛盾，不可行或有二义性的需求也是不可验证的。

需求规格说明的特点

完整性

不能遗漏任何必要的需求信息。遗漏需求将很难查出。注重用户的任务而不是系统的功能将有助于你避免不完整性。如果知道缺少某项信息，用 TBD(“待确定”)作为标准标识来标明这项缺漏。在开始开发之前，必须解决需求中所有的 TBD 项。

一致性

一致性是指与其它软件需求或高层（系统，业务）需求不相矛盾。在开发前必须解决所有需求间的不一致部分。只有进行一番调查研究，你才能知道某一项需求是否确实正确。

可修改性

在必要时或为维护每一需求变更历史记录时，应该修订 SRS。这就要求每项需求要独立标出，并与别的需求区别开来，从而无二义性。每项需求只应在 SRS 中出现一次。这样更改时易于保持一致性。另外，使用目录表、索引和相互参照列表方法将使软件需求规格说明更容易修改。

可跟踪性

应能在每项软件需求与它的根源和设计元素、源代码、测试用例之间建立起链接链，这种可跟踪性要求每项需求以一种结构化的，粒度好（fine-grained）的方式编写并单独标明，而不是大段大段的叙述。第 18 章将详细说明需求的可跟踪性。

需求的开发和管理 Top

需求中名词术语的混淆将导致对科目（规范）（discipline）叫法的不一致。一些作者把整个需求范围称之为“需求工程”，而另一些则称之为“需求管理”。我认为可把整个软件需求工程研究领域划分为需求开发（本书的第二部分）和需求管理（本书第三部分）两部分更合适，如图 1-2 所示：

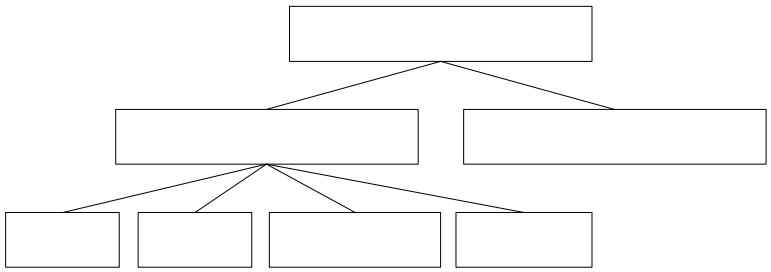

图1-2 需求工程域的层次分解示意图

需求开发可进一步分为：问题获取（elicitation）、分析(analysis)、编写规格说明（specification）和验证（verification）四个阶段（Thayer 和 Dorfman 1997）。这些子项包括软件类产品中需求收集、评价、编写文档等所有活动。需求开发活动包括以下几个方面：

- 确定产品所期望的用户类。
- 获取每个用户类的需求。
- 了解实际用户任务和目标以及这些任务所支持的业务需求。
- 分析源于用户的信息以区别用户任务需求、功能需求、业务规则、质量属性、建议解决方法和附加信息。
- 将系统级的需求分为几个子系统，并将需求中的一部份分配给软件组件。
- 了解相关质量属性的重要性。
- 商讨实施优先级的划分。
- 将所收集的用户需求编写成规格说明和模型。
- 评审需求规格说明，确保对用户需求达到共同的理解与认识，并在整个开发小组接受说明之前将问题都要弄清楚。

需求管理需要“建立并维护在软件工程中同客户达成的契约”（CMU/SEI 1995）。这种契约都包含在编写的需求规格说明与模型中。客户的接受仅是需求成功的一半，开发人员也必须能够接受他们，并真正把需求应用到产品中。通常的需求管理活动包括：

- 定义需求基线（迅速制定需求文档的主体）。
- 评审提出的需求变更、评估每项变更的可能影响从而决定是否实施它。
- 以一种可控制的方式将需求变更融入到项目中。
- 使当前的项目计划与需求一致。
- 基于估计变更需求所产生影响的基础上，协商新的承诺（约定）。

◆ 让每项需求都能与其对应的设计、源代码和测试用例联系起来以实现跟踪。

◆ 在整个项目过程中跟踪需求状态及其变更情况。

由图 1-3 中可以从另一个角度来看需求开发和需求管理之间的区别：

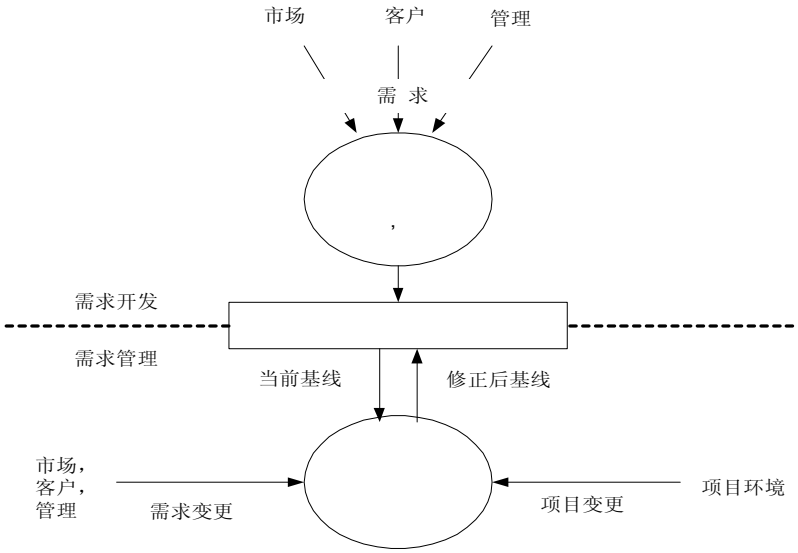

图1-3　需求开发与需求管理之间的界限

下一步

◆ 记录你在当前项目或以前项目中所遇到的与需求相关的问题。指明每一个是需求开发问题还是需求管理问题，以及这些问题带来的影响及其产生的根本原因。

◆ 与你的组员和其他风险承担者（客户，市场调查人员，项目管理者）一起讨论当前或以前项目中的需求问题，及其产生的根源和带来的影响。向所有参与者指明，如果想解决这些困难，必须正视它，大家是否为此做好准备了呢？

◆ 整理出对整个项目人员一天训练用的软件需求课程，人员要包括重要的客户，市场人员和管理人员。训练是一种有效的团队学习与合作的方法。大家将会在训练中达成术语与技术上的共识，以有利于相互交流沟通与协作。

第 2 章　客户的需求观 Top

Contoso 制药公司的高级管理长官 Gerhard ,会见 Contoso 公司的信息系统开发小组的新管理员 Cynthia。“我们需要建立一套化学制品跟踪信息系统”，Gerhard 说道。“该系统可以记录在库房或某个实验室中已有的化学药品，这样，化学专家可以直接从楼下的某人那里拿到所需的药品，而不必再买一瓶新的。另外，卫生保健部门也得为联邦政府写些关于化学药品的使用报告。你们小组能在五个月内开发出该系统来吗？”

“我已经明白这个项目的重要性了，Gerhard”Cynthia 说道。“但在我制定计划前，我们必须收集一些系统的需求。”

Gerhard 觉得很奇怪“你的意思是什么？我不是刚告诉你我的需求了吗？”

“实际上，你只说明了整个项目的概念与目标”，Cynthia 解释道。“这些高层次的业务需求并不能为我们提供足够的详细信息以确定究竟要开发什么样的软件，以及需要多长时间。我需要一些分析人员与一些知道系统使用要求的化学专家进行讨论，然后才能真正明白达到业务目标所需的各种功能和用户的要求。我们甚至并不需要开发一个新的软件系统，这样可节省许多钱。”

Gerhard 此前还从未遇到过类似这位系统开发人员的看法。“那些化学专家都非常忙”他坚持道，“他们没有时间与你们详细讨论各种细节，你不能让你的手下的人说明吗？”

Cynthia 尽力解释从使用新系统的用户处收集需求的合理性。“如果我们只是凭空猜想用户要求，结果不会令人满意。我们只是软件开发人员，而并非化学专家。我们并不能真正明白化学专家们需要这个化学制品跟踪系统做些什么。我曾经尝试过，未真正明白这些问题就匆忙开始编码，结果没有人对产品满意。”

“行了，行了，我们没有那么多时间”Gerhard 坚持道。“我来告诉你需求，请马上开始开发系统。随时将你们的进展情况告诉我。”

像这样的对话经常出现在软件开发过程中。要求开发一个新信息系统的客户通常并不懂得从系统的实际用户处得到信息的重要性。市场人员在有了一个很不错的新产品想法后，也就自认为能充分代表产品用户的兴趣要求。然而，直接从产品的实际用户处收集需求有着不可替代的必要性。通过对 8380 个项目的调查发现，导致项目失败的最主要两个原因是缺乏用户参与和不完整的需求以及不完整的规格说明(Standish1995)。

引起需求问题的一部分原因是对不同层次需求（业务、用户、功能）的混淆所致。Gerhard 说明了一些业务需求，但他并不能描述用户需求，因为他并不是“化学制品跟踪系统”的实际使用者。只有实际用户才能描述他们要用此系统必须完成的任务。但他们又不能指出完成这些任务所有具体的功能需求。

本章说明客户与开发人员之间的关系，它对软件项目开发的成功极为关键。我建议写一份软件客户需求权利书和相应的软件客户需求义务书，以强调客户（和实际用户）参与需求开发过程的重要性。

谁是客户？ Top

通常意义下，客户是指直接或间接从产品中获得利益的个人或组织。软件客户包括提出要求、支付款项、选择、具体说明或使用软件产品的项目风险承担者(stakeholder)或是获得产品所产生的结果的人。

Gerhard 代表支付、采购(procure)或投资软件产品的这类客户，处于 Gerhard 层次上的客户有义务说明业务需求。他们应阐明产品的高层次概念和将发布产品的主要业务内容。在第 6 章中讨论到业务需求应说明客户、公司和想从该系统获利的风险承担者或从系统中取得结果的用户他们所要求的目标。业务需求为后继工作建立了一个指导性的框架。其它任何的说明都应遵从业务需求的规定，然而业务需求并不能为开发人员提供许多开发所需的细节说明。

下一层需求——用户需求——必须从使用产品的用户处收集。因此这些用户（通常称作最终用户），构成了另一种软件客户。他们能说清楚要使用该产品完成什么任务和一些非功能性的特性，而这些特性会对使用户很好接收具有该特点的产品是重要的。

说明业务需求的客户有时将试图替代用户说话，但通常他们根本无法准确说明用户需求。因为对信息系统、合同(contract)或是客户应用程序的开发、业务需求应来自风险承担者，而用户需求则应来自产品的真正使用操作者。

不幸的是，这两种客户可能都觉得他们没有时间与(收集、分析与编写需求说明)需求分析者讨论。有时客户还希望分析人员或开发人员无须讨论和编写文档就能说出用户的需求。除非遇到的需求极为简单，否则不能这样做。如果你的组织关注软件的成功，那必须要花上数天时间来消除需求中模糊不清的地方和一些使程序人员感到困惑的方面。

商业软件开发的情况有些不同，因为通常其客户就是用户。正如市场部这类客户代理，可能想确定究竟软件产品的购买者会喜欢什么。但即使是商业软件，也应该让实际用户参与到收集需求的过程中来（第 7 章将谈及）。如果你不这样作，那产品很可能会因缺乏足够用户提供的信息而出现不少隐患。

客户与开发人员之间的合作关系 Top

优秀的软件产品是建立在优秀的需求基础之上的。而高质量的需求来源于客户与开发人员之间有效的交流与合作。但通常，开发人员与客户或客户代理人，如市场人员间的关系反而会成为一种对立关系。双方的管理者都只想自己的利益而搁置用户提供的需求从而产生摩擦，在这种情况下，不会给双方带来一点益处。

只有当双方参与者都明白要成功自己需要什么，同时也应知道要成功合作方需要什么时，才能建立起一种合作关系。由于项目压力与日渐增，所有风险承担者有着一个共同的目标这一点容易被遗忘。其实大家都想开发出一个既能实现商业价值，又能满足用户需要，还能使开发者感到满足的优秀软件产品。

软件客户需求权利书列出了十条关于客户在项目需求工程实施中与分析人员、开发人员交流时的合法要求。每一项权利都对应着软件开发人员、分析人员的义务。而软件客户需求义务书也列出了十条关于客户在需求过程中应承担的义务。如果愿意，可以将其作为开发人员的权利书。

软件客户需求权利书

客户有如下权利：

1. 要求分析人员使用符合客户语言习惯的表达。
2. 要求分析人员了解客户系统的业务及目标。
3. 要求分析人员组织需求获取期间所介绍的信息，并编写软件需求规格说明。

4. 要求开发人员对需求过程中所产生的工作结果进行解释说明。
5. 要求开发人员在整个交流过程中保持和维护一种合作的职业态度。
6. 要求开发人员对产品的实现及需求都要提供建议，拿出主意。
7. 描述产品使其具有易用、好用的特性。
8. 可以调整需求，允许重用已有的软件组件。
9. 当需要对需求进行变更时，对成本、影响、得失（trade-off）有个真实可信的评估。
10. 获得满足客户功能和质量要求的系统，并且这些要求是得到开发人员同意的。

软件客户需求义务书

客户有下列义务：

1. 给分析人员讲解业务及说明业务方面的术语等专业问题。
2. 抽出时间清楚地说明需求并不断完善。
3. 当说明系统需求时，力求准确详细。
4. 需要时要及时对需求做出决策。
5. 要尊重开发人员的成本估算和对需求的可行性分析。
6. 对单项需求、系统特性或使用实例划分优先级。
7. 评审需求文档和原型。
8. 一旦知道要对项目需求进行变更，要马上与开发人员联系。
9. 在要求需求变更时，应遵照开发组织确定的工作过程来处理。
10. 尊重需求工程中开发人员采用的流程（过程）。

当为内部集团使用而开发软件时，这些权利和义务可以直接应用于客户。这也适用于那些有合同关系或者有明确的主要客户集的情况。对普遍市场产品的开发，这些权利和义务更适于像市场部这样的客户代理者。

作为项目计划的一部分，客户和开发人员应评审上述两张列表并达成共识。一些很忙的客户可能不愿意积极参与需求过程（也即，他们不太接受义务书#2），而缺少客户参与将很可能导致不理想的产品。故一定要确保需求开发中的主要参与者都了解并接受他们的义务。如果遇到分歧，通过协商以达成对各自义务的相互理解，这样能减少今后的摩擦（如一方要求而另一方不愿意或不能满足要求）。

软件客户需求权利书 Top

权利#1：要求分析人员使用符合客户语言习惯的表达

需求讨论应集中于业务需要和任务，故要使用业务术语，你应将其教给分析人员，而你不一定要懂得计算机的行业术语。

权利#2：要求分析人员了解客户的业务及目标

通过与用户交流来获取用户需求、分析人员才能更好地了解你的业务任务和怎样才能使产品更好地满足你的需要。这将有助于开发人员设计出真正满足你的需要并达到你期望的优秀软件。为帮助开发人员和分析人员，可以考虑邀请他们观察你或你的同事是怎样工作的。如果新开发系统是用来替代已有的

系统，那么开发人员应使用一下目前的系统，这将有利于他们明白目前系统是怎样工作的，其工作流程的情况，以及可供改进之处。

权利#3：要求分析人员编写软件需求规格说明

分析人员要把从你和其他客户那里获得的所有信息进行整理，以区分开业务需求及规范、功能需求、质量目标、解决方法和其它信息。通过这些分析就能得到一份软件需求规格说明。而这份软件需求规格说明便在开发人员和客户之间针对要开发的产品内容达成了协议。SRS 可以用一种你认为易于翻阅和理解的方式组织编写。要评审编写出的规格说明以确保它们准确而完整地表达了你的需求。一份高质量的软件需求规格说明能有助于开发人员开发出真正需要的产品。

权利#4：要求得到需求工作结果的解释说明

分析人员可能采用了多种图表作为文字性软件需求规格说明的补充。因为如工作流程图那样的图表能很清楚地描述出系统行为的某些方面。所以需求说明中的各种图表有着极高的价值。虽然它们不太难于理解，但是你很可能对此并不熟悉。因此可以要求分析人员解释说明每张图表的作用或其它的需求开发工作结果和符号的意义，及怎样检查图表有无错误及不一致等。

权利#5: 要求开发人员尊重你的意见

如果用户与开发人员之间不能相互理解，那关于需求的讨论将会有障碍，共同合作能使大家“兼听则明”。参与需求开发过程的客户有权要求开发人员尊重他们并珍惜他们为项目成功所付出的时间。同样，客户也应对开发人员为项目成功这一共同目标所作出的努力表示尊重与感激。

权利#6：要求开发人员对需求及产品实施提供建议，拿出主意

通常，客户所说的“需求”已是一种实际可能的实施解决方案，分析人员将尽力从这些解决方法中了解真正的业务及其需求，同时还应找出已有系统不适合当前业务之处，以确保产品不会无效或低效。在彻底弄清业务领域内的事情后，分析人员有时就能提出相当好的改进方法。有经验且富有创造力的分析人员还能提出增加一些用户并未发现的很有价值的系统特性。

权利#7：描述产品易使用的特性

你可以要求分析人员在实现功能需求之上还要注重软件的易用性。因为这些易用特性或质量属性能使你更准确、高效地完成任务。例如，客户有时要求产品要“用户友好”或“健壮”或“高效率”，但这对于开发人员来说，太主观了并无实用价值。正确的应是：分析人员通过询问和调查了解客户所要 的友好、健壮、高效所包含的具体特性（第 11 章将详细讨论它）。

权利#8 调整需求，允许重用已有的软件组件

需求通常要有一定的灵活性。分析人员可能发现已有的某个软件组件与你描述的需求很相符。在这种情况下，分析人员应提供一些修改需求的选择以便开发人员能够在新系统开发中重用一些已有的软件。如果有可重用的机会出现，同时你又能调整你的需求说明，那就能降低成本和节省时间，而不必严格按原有的需求说明开发。所以说，如果想在产品中使用一些已有的商业常用组件，而它们并不完全适合你所需的特性，这时一定程度上的需求灵活性就显得极为重要了。

权利#9：要求对变更的代价提供真实可信的评估

有时人们面临更好、也更昂贵的方案时，会做出不同的选择。而这时，对需求变更的影响进行评估从而对业务决策提供帮助，这是十分必要的，所以，你有权利要求开发人员通过分析给出一个的确真实可信的评估，包括影响、成本和得失等评估。开发人员不能由于不想实施变更而随意夸大评估成本。

权利#10：获得满足客户功能和质量要求的系统

每个人都希望项目获得成功。但这不仅要求你要清晰地告知开发人员关于系统“做什么”所需的所有信息，而且还要求开发人员能通过交流了解清楚取舍与限制。一定要明确说明你的假设和潜在的期望。否则，开发人员开发出的产品很可能无法让你满意。

软件客户需求义务书 Top

义务#1：给分析人员讲解你的业务

分析人员要依靠你给他们讲解的业务概念及术语。但你不能指望分析人员会成为该领域的专家，而只能让他们真正明白你的问题和目标。不要期望分析人员能把握你们业务的细微与潜在之处，他们很可能并不知道那些对于你和你的同事来说理所当然的“常识”。

义务#2：抽出时间清楚地说明并完善需求

客户很忙，经常在最忙的时候还得参与需求开发。但无论如何，你有义务抽出时间参与“头脑风暴”会议的讨论，接受采访或其它获取需求的活动。有时分析人员可能先以为明白了你的观点，而过后发现还需要你的讲解。这时，请耐心一些对待需求和需求的精化工作过程中的反复，因为它是人们交流中的很自然的现象，何况这对软件产品的成功极为重要。

义务#3：准确而详细地说明需求

编写一份清晰、准确的需求文档是很困难的。由于处理细节问题不但烦人而且又耗时，故很容易留下模糊不清的需求。但是，在开发过程中，必须得解决这种模糊性和不准确性。而你恰是为解决这些问题作出决定的最佳人选。不然的话，你就只好靠开发人员去正确猜测了。

在需求规格说明中暂时加上 TBD 的标志是个不错的办法。用该标志可指明了哪些需要进一步探讨、分析或增加信息的地方。不过，有时也可能因为某个特殊需求难以解决或没有人愿意处理它而注上 TBD 标志。尽量将每项需求的内容都阐述清楚，以便分析人员能准确的将其写进软件需求规格说明中。如果你一时不能准确表述，那就得允许获取必要的准确信息这样一个过程。通常使用所谓的原型技术。通过开发的原型，你可以同开发人员一起反复修改，不断完善需求定义。

义务#4：及时地作出决定

正如一位建筑师为你修建房屋，分析人员将要求你做出一些选择和决定。这些决定包括来自多个用户提出的处理方法或在质量特性冲突和信息准确度中选择折衷方案等。有权做出决定的客户必须积极地对待这一切，尽快做处理、做决定。因为开发人员通常只有等你做出了决定才能行动，而这种等待会延误项目的进展。

义务#5：尊重开发人员的需求可行性及成本评估

所有的软件功能都有其成本价格，开发人员最适合预算这些成本（尽管许多开发人员并不擅长评估预测）。你所希望的某些产品特性可能在技术上行不通，或者实现它要付出极为高昂的代价。而某些需求试图在操作环境中要求不可能达到的性能或试图得到一些根本得不到的数据，开发人员会对此作出负面的评价意见，你应该尊重他们的意见。

有时，你可以重新给出一个在技术上可行、实现上便宜的需求，例如，要求某个行为在“瞬间”发生是不可行的，但换种更具体的时间需求说法（“在 50 毫秒以内”），这就可以实现了。

义务#6： 划分需求优先级别

绝大多数项目没有足够的时间或资源来实现功能性的每个细节。决定哪些特性是必要的，哪些是重要的，哪些是好的，是需求开发的主要部分。只能由你来负责设定需求优先级，因为开发者并不可能按

你的观点决定需求优先级。开发者将为你确定优先级提供有关每个需求的花费和风险的有关信息。当你设定优先级时，你帮助开发者确保在适当的时间内用最小的开支取得最好的效果。

在时间和资源限制下，关于所需特性能否完成或完成多少应该尊重开发人员的意见。尽管没有人愿意看到自己所希望的需求在项目中未被实现，但毕竟是要面对这种现实的。业务决策有时不得不依据优先级来缩小项目范围或延长工期，或增加资源，或在质量上寻找折衷。

义务#7：评审需求文档和原型

正如我们将在第 14 章讨论的，无论是正式的还是非正式的方式，对需求文档进行评审都会对软件质量提高有所帮助。让客户参与评审才能真正鉴别需求文档是否的确完整、正确说明了期望的必要特性。评审也给客户代表提供一个机会，给需求分析人员带来反馈信息以改进他们的工作。如果你认为编写的需求文档不够准确，就有义务尽早告诉分析人员并为改进提供建议。

通过阅读需求规格说明，很难想象实际的软件是什么样子的。更好的方法是先为产品开发一个原型。这样你就能提供更有价值的反馈信息给开发人员，帮助他们更好地理解你的需求。必须认识到：原型并非是一个实际产品，但开发人员能将其转变、扩充成功能齐全的系统。

义务#8：需求出现变更要马上联系

不断的需求变更会给在预定计划内完成高质量产品带来严重的负面影响。变更是不可避免的，但在开发周期中变更越在晚期出现，其影响越大。变更不仅会导致代价极高的返工，而且工期也会被迫延误，特别是在大体结构已完成后又需要增加新特性时。所以一旦你发现需要，变更需求时，请一定立即通知分析人员。

义务#9：应遵照开发组织处理需求变更的过程

为了将变更带来的负面影响减少到最低限度，所有的参与者必须遵照项目的变更控制过程。这要求不放弃所有提出的变更，对每项要求的变更进行分析、综合考虑，最后作出合适的决策以确定将某些变更引入项目中。

义务#10：尊重开发人员采用的需求工程过程

软件开发中最具挑战性的莫过于收集需求并确定其正确性。分析人员采用的方法有其合理性所在。也许你认为需求过程不太划算，但请相信花在需求开发上的时间是“很有价值”的。如果你理解并支持分析人员为收集、编写需求文档和确保其质量所采用的技术，那么整个过程将会更为顺利。尽管去询问分析人员为什么他们要收集某些信息，或参与与需求有关的活动。

“签约”意味着什么？

为所开发产品的需求签订协议是客户与开发人员关系中的重要部分。许多组织在需求文档中使用“签约”这个概念来作为客户同意需求的标志行为。故要让所有需求参与者都真正明白“签约”的意思。

但存在这样一个问题：客户代表经常把“签约”看作是毫无意义的。“他们要我在一张纸的最后一行文字下面签上名字，于是我就签了，否则这些开发人员不开始编码。”这种态度会给将来带来麻烦。譬如客户想更改需求或对产品有不满时。“不错，我是在需求上签署了名字，但我并没有时间去读完所有的内容。我是相信你们的，是你们非要让我签字的。”

同样的问题也会发生在仅把签约看作是完成文档的管理人员身上。一旦有需求变更出现，他便指着软件需求规格说明说道：“但你已经在需求上签约了，所以这些便是我们所要开发的。如果你想要别的什么，你应早些告诉我们。”

这样的态度都是不对的，不可能在项目早期就了解所有需求，而且毫无疑问需求将会出现变更。在需求上签约是终止需求开发过程的正确方法。然而，参与者必须明白他们的签约意味着什么。

更为重要的是签约名字是建立在一个需求协议的基线上，因此在需求规格说明上的签约应该这样理解：“我同意这份文档表述了目前我们对项目软件需求的了解。进一步的变更可在此基线上通过项目定义的变更过程来进行。我知道变更可能会使我们要重新协商成本、资源和项目工期任务等”。

关于基线达成一定共识会易于忍受将来的摩擦，这些摩擦来源于项目的改进和需求的误差或市场和业务的新要求等。给初步的需求开发工作画上双方都明确的句号会有助你形成一个持续良好的客户与开发人员的关系，为项目的成功奠定了基础。

下一步：

- 让客户提供项目的业务需求和用户需求。对权利书和义务书的条目，哪些被客户接受、理解并付诸实践了？那些没有？
- 与你的重要客户一起讨论权利书和义务书，以达成协议，并付诸实践。这些行为会有助于客户和开发人员更好地互相理解，以形成更融洽的关系。
- 如果你是软件开发项目中的客户参与方，你感到你的需求权利书没有被充分尊重，就可以与软件项目的领导人员或业务分析人员一起讨论权利书的内容。要想建立一种合作的工作关系，就要尽力使对方对你的义务书感到满意。

第 3 章 需求工程的推荐方法 Top

十年前，我曾热衷于追求软件的开发方法论的研究，将整套整套的模型、技术等用于解决项目难题。但现在我更注重应用“最佳方法”。最佳方法强调将软件工具包拆分成多个子包以分别应用于不同的问题，而并不是去设计或购买一整套的解决方案。即使你采用了一套商业上的方法，你也应当从中提取出好的技术，将它有效地应用于实践。

“最佳方法”这个词值得讨论一下：谁能确定什么是“最佳”的呢？而且，得到这个结论的依据何在？一种方案是把这方面的专家召集起来分析众多不同组织中成功和失败的项目（ Browm 1996）。专家们将那些成功项目中提供高效的方法和失败项目中导致低效甚至无效的方法都归纳出来。这样，专家们就能找到公认的能收到实效的关键方法。这些方法即是“最佳方法”，其本质就是有助于项目成功的有效方法。

本章的标题是“需求工程的推荐方法”而非“最佳方法”。下面分七类介绍了四十余种方法，能有助于开发小组的需求工作，推荐方法如表 3-1 所列。

表 3-1　　需求工程推荐方法

需求		
知识技能	管理	项目管理
●培训需求分析人员	●确定变更控制过程	●选择合适的生存期
●培训用户代表和管理人员	● 建立变更控制委员会	●确定需求的基本计划
●培训开发人员了解应用领域	●进行变更影响分析	●协商约定
●汇编术语	●跟踪影响工作产品的每项变更	●管理需求风险
	●编写需求文档的基线和控制版本	●跟踪需求工作
	●维护变更历史记录	
	●跟踪需求状态	
	● 衡量需求稳定性	
	●使用需求管理工具	

需求开发			
获取	分析	编写规格说明书	验证
●编写项目视图与范围	●绘制示意图	●采用软件需求规格说明模版	●审查需求文档
●确定需求开发过程	●创建开发原型	●指明需求来源	●依据需求编写测试用例

●用户群分类　●分析可行性　●为每项需求注上标号　●编写用户手册
●选择产品代表　●确定需求优先级　●记录业务规范　●确定合格的标准
●建立核心队伍　●为需求建立模型　●创建需求跟踪能力矩阵
●确定使用实例　●编写数据字典
●召开应用程序开发联系（JAD）会议　●应用质量功能调配（QFD）
●分析用户工作流程
●确定质量属性
●检查问题报告
●需求重用

并非上面所有的条目都是最佳方法，或许也并非全部经过了系统的评估。但无论如何，我和许多实践者都觉得这些技术是很有效的(Sommerville 和 Sawyer 1997)。其中每一条都在本书中给予了简要介绍或在其它资料中给予了更详细的讨论。

表 3-2 把表 3-1 中的方法按重要性和实施难度进行了分组。由于所列的方法都是有所裨益的，故最好是循序渐进，先从那些相对容易实施而对项目有很大影响的方法开始。

表 3-2　　实施需求工程的推荐方法

优先级别	难度		
	高	中	低
高	• 确定需求开发过程 • 确定需求的基本计划 • 协商约定	• 确定使用实例 • 确定质量属性 • 确定需求优先级 • 采用 SRS 模板 • 确定变更控制过程 • 建立变更控制委员会 • 审查需求文档	• 培训开发人员了解应用领域 • 编写项目视图与范围 • 用户群分类 • 绘制示意图 • 指明需求来源 • 为每项需求注上标号 • 编写需求文档的基线和控制版本
中	• 培训用户代表和管理人员 • 为需求建立模型 • 管理需求风险 • 使用需求管理工具 • 创建需求跟踪能力矩阵	• 培训需求分析人员 • 建立核心队伍 • 创建开发原型 • 分析可行性 • 确定合格的标准 • 进行变更影响分析 • 跟踪影响工作产品的每项变更 • 选择合适的生存期	• 汇编术语 • 选择产品代表 • 编写数据字典 • 记录业务规范 • 依据需求编写测试用例 • 跟踪需求状态
低	• 召开应用程序开发联系 • 分析用户工作流程		

会议　　• 检查问题报告

低　• 需求重用　　• 编写用户手册

• 应用质量功能调配　　• 维护变更历史记录

• 衡量需求稳定性　　• 跟踪需求工作

不要想着把所有这些方法都用于你的下一个项目。而应该考虑将其中的一些方法推荐到你的需求工具包中，不管你的项目处在开发的哪个阶段，你都可以马上开始应用某些方法，譬如变更管理的处理。其它如需求获取等可以在你的下一个项目开始时付诸应用。当然其它一些方法也可能并不适合你目前的项目。

第 4 章介绍了一些用于评价需求工程的方法，并可设计一张实施需求方法改进的步骤图。具体的改进方法在此处和第 4 章中都给予了介绍。

知识技能

绝大部分的软件开发人员都没有接受过高效需求工程所需技能的正规培训，但许多开发人员在职业生活中的某个阶段总会扮演一个需求分析员的角色，与客户一起工作：收集，分析，编写需求文档。不能过高期望开发人员在需求工程的信息沟通中的“天份”。一定的培训将有助于提高需求分析员的能力和水平。

因为需求对项目成功极为重要，所有的项目风险承担者都应该对需求工程的重要性、合理性及其方法有一个基本的了解。把项目风险承担者（例如开发人员，市场人员，客户，测试人员和管理人员）召集起来进行为期一天的需求过程概要学习，这对建立一个合作团队是很有效的。所有参与者都会更好的明白各自所面临的挑战是什么，以及为了整个团队获得成功大家都需要作些什么。同样，开发人员也能对应用领域的术语和一些基本概念有大致的了解。

培训需求分析人员。所有的开发人员都应接受一个基本的需求工程培训。但那些负责收集(capturing)、编写文档和分析用户需求的人员应当进行为期一周或更长时间的培训。把高水平的需求人员组织起来，通过良好的信息交流，了解应用领域并有效地应用需求工程中的成熟技术。

培训软件需求的用户代表和管理人员。参与软件开发的用户代表应接受为期一天左右关于需求工程的培训，开发管理者和客户管理者也应参加。这样的培训将使他们明白强调需求的重要性，以及忽略需求将带来的风险。参加过我组织的需求讨论会的一些用户表示，他们在此之后更能理解软件开发人员了。

让开发人员了解应用领域的基本概念。组织一些简短的关于客户业务活动、术语、目标等方面的讨论会以帮助开发人员对应用领域有个基本了解。这能减少误解及工程中的返工。你可能要为每位开发人员安排一个用户伙伴以便在项目过程中解释业务术语和概念。产品代表就应该扮演这样的角色。

编写项目术语汇编。为减少沟通方面的问题，编一部术语汇编将项目应用领域的专用词汇给予定义说明，既要包括那些有多种含义与用法的术语，也要包括那些在专用领域和一般使用中有不同含义的词。

需求获取

第 1 章讨论了需求的三个层次：业务，用户和功能。在项目中它们在不同的时间来自不同的来源，也有着不同的目标和对象，并需以不同的方式编写成文档。业务需求（或产品视图和范围）不应包括用户需求（或使用实例），而所有的功能需求都应该源于用户需求。同时你也需要获取非功能需求，如质量属性。你可以在下列章节中找到相关主题的详细内容：

- 第 4 章—确定需求开发过程。
- 第 6 章—编写项目视图和范围文档。

- 第 7 章—将用户群分类并归纳其特点，为每个用户类选择产品代表（product champion）。
- 第 8 章——让用户代表确定使用实例。
- 第 11 章——确定质量属性和其它非功能需求。

确定需求开发过程。确定如何组织需求的收集、分析、细化并核实的步骤，并将它编写成文档，对重要的步骤要给予一定指导，这将有助于分析人员的工作，而且也使收集需求活动的安排和进度计划更容易进行。

编写项目视图和范围文档。项目视图和范围文档应该包括高层的产品业务目标，所有的使用实例和功能需求都必须遵从能达到的业务需求。项目视图说明使所有项目参与者对项目的目标能达成共识。而范围则是作为需求或潜在特性的参考。

将用户群分类并归纳各自特点。为避免出现疏忽某一用户群需求的情况，要将可能使用产品的客户分成不同组别。他们可能在使用频率、使用特性、优先等级或熟练程度等方面都有所差异。详细描述出它们的个性特点及任务状况，将有助于产品设计。

选择每类用户的产品代表。为每类用户至少选择一位能真正代表他们需求的人作为那一类用户的代表并能作出决策。这对于内部信息系统的开发是最易实现的，因为此时，用户就是身边的职员。而对于商业开发，就得在主要的客户或测试者中建立起良好的合作关系，并确定合适的产品代表。他们必须一直参与项目的开发而且有权作出决策。

建立起典型用户的核心队伍。把同类产品或你的产品的先前版本用户代表召集起来，从他们那里收集目前产品的功能需求和非功能需求。这样的核心队伍对于商业开发尤为有用，因为你拥有一个庞大且多样的客户基础。与产品代表的区别在于，核心队伍成员通常没有决定权。

让用户代表确定使用实例。从用户代表处收集他们使用软件完成所需任务的描述——使用实例，讨论用户与系统间的交互方式和对话要求。在编写使用实例的文档时可采用标准模版，在使用实例基础上可得到功能需求。

召开应用程序开发联系会议。召开应用程序开发联系（JAD）会议是范围广的、简便的专题讨论会（workshop），也是分析人员与客户代表之间一种很好的合作办法，并能由此拟出需求文档的底稿。该会议通过紧密而集中的讨论得以将客户与开发人员间的合作伙伴关系付诸于实践（ Wood 和 Silver 1995）。

分析用户工作流程。观察用户执行业务任务的过程。画一张简单的示意图（最好用数据流图）来描绘出用户什么时候获得什么数据，并怎样使用这些数据。编制业务过程流程文档将有助于明确产品的使用实例和功能需求。你甚至可能发现客户并不真的需要一个全新的软件系统就能达到他们的业务目标（ McGraw 和 Harbison 1997）。

确定质量属性和其它非功能需求。在功能需求之外再考虑一下非功能的质量特点，这会使你的产品达到并超过客户的期望。这些特点包括性能、有效性、可靠性、使用性等，而在这些质量属性上客户提供的信息相对来说就非常重要了。

通过检查当前系统的问题报告来进一步完善需求。客户的问题报告及补充需求为新产品或新版本提供了大量丰富的改进及增加特性的想法，负责提供用户支持及帮助的人能为收集需求过程提供极有价值的信息。

跨项目重用需求。如果客户要求的功能与已有的产品很相似，则可查看需求是否有足够的灵活性以允许重用一些已有的软件组件。

需求分析（requirement analysis）

需求分析包括提炼、分析和仔细审查已收集到的需求，以确保所有的风险承担者都明白其含义并找出其中的错误、遗漏或其它不足的地方。分析员通过评价来确定是否所有的需求和软件需求规格说明都达到了第 1 章中优秀需求说明的要求。分析的目的在于开发出高质量的需求，这样你能作出实用的项目估算并可以进行设计、构造和测试。

通常，把需求中的一部分用多种形式来描述，如同时用文本和图形来描述。分析这些不同的视图将揭示出一些更深的问题，这是单一视图无法提供的（Davis 1995）。分析还包括与客户的交流以澄清某些混淆，并明确哪些需求是更为重要的。其目的是确保所有风险承担者尽早地对项目达成共识并对将来的产品有个相同而清晰的认识。下面几章对需求分析中的任务进行了详细讨论：

- 第 6 章——绘制系统上下文示意图。
- 第 9 章——建立数据字典。
- 第 10 章——为需求建立模型。
- 第 12 章——建立用户接口原型。
- 第 13 章——确定需求优先级。

绘制系统上下文示意图。这种示意图是用于定义系统与系统外部实体间的界限和接口的简单模型。同时它也明确了通过接口的信息流和物质流。

创建用户接口原型。当开发人员或用户不能确定需求时，开发一个用户接口原型———个局部的可能实现——这样使得许多概念和可能发生的事更为直观明了。用户通过评价原型将使项目参与者能更好地相互理解所要解决的问题。注意要找出需求文档与原型之间的所有冲突之处。

分析需求可行性。在允许的成本、性能要求下，分析每项需求实施的可行性，明确与每项需求实现相联系的风险，包括与其它需求的冲突，对外界因素的依赖和技术障碍。

确定需求的优先级别。应用分析方法来确定使用实例、产品特性或单项需求实现的优先级别。以优先级为基础确定产品版本将包括哪些特性或哪类需求。当允许需求变更时，在特定的版本中加入每一项变更，并在那个版本计划中作出需要的变更。

为需求建立模型。需求的图形分析模型是软件需求规格说明极好的补充说明。它们能提供不同的信息与关系以有助于找到不正确的、不一致的、遗漏的和冗余的需求。这样的模型包括数据流图、实体关系图、状态变换图、对话框图、对象类及交互作用图。

创建数据字典。数据字典是对系统用到的所有数据项和结构的定义，以确保开发人员使用统一的数据定义。在需求阶段，数据字典至少应定义客户数据项以确保客户与开发小组是使用一致的定义和术语。分析和设计工具通常包括数据字典组件。

使用质量功能调配。质量功能调配(QFD）是一种高级系统技术，它将产品特性、属性与对用户价值联系起来。该技术提供了一种分析方法以明确那些是客户最为关注的特性。QFD 将需求分为三类：期望需求，即客户或许并未提及，但如若缺少会让他们感到不满意；普通需求；和兴奋需求，即实现了会给客户带去惊喜，但若未实现也不会受到责备（Zultner 1993;Pardee 1996）。

需求规格说明（requirement specification）

无论你的需求从何而来，也不管你是怎样得到的，你都必须用一种统一的方式来将它们编写成可视文档。业务需求要写成项目视图和范围文档，用户需求要用一种标准使用实例模板编写成文档。而软件需求规格说明则包含了软件的功能需求和非功能需求。你必须为每项需求明确建立标准的风格，并在 SRS 中采用，以确保 SRS 的统一风格，同时读者也会明白怎样解释它。下列章节讨论了关于编写需求文档的几个方面：

- 第 8 章——记录业务规范。
- 第 9 章——采用 SRS 模板；为每项需求注上标号。
- 第 18 章——指明需求来源；创建需求跟踪能力矩阵。

采用 SRS 模板。在你的组织中要为编写软件需求文档定义一种标准模板。该模板为记录功能需求和各种其它与需求相关的重要信息提供了统一的结构。注意，其目的并非是创建一种全新的模板，而是采用一种已有的且可满足项目需要并适合项目特点的模板。许多组织一开始都采用 IEEE 标准 830-1998(IEEE 1998)描述的 SRS 模板。要相信模板是很有用的，但有时要根据项目特点进行适当的改动。

指明需求的来源。为了让所有项目风险承担者明白 SRS 中为何提供这些功能需求，要将每项需求都能追溯其来源，这可能是一种使用实例（use case）或其它客户要求，也可能是某项更高层系统需求、业务规范、政府规则、标准或别的外部来源。

为每项需求注上标号。制定一种风格来为 SRS 中的每项需求提供一个独立的可识别的标号或记号。这种风格应当很健全，允许增加、删除和修改。作了标号的需求使得需求能被跟踪，记录需求变更并为需求状态和变更活动建立度量。

记录业务规范。业务规范是指关于产品的操作原则，比如谁能在什么情况下采取什么动作。将这些编写成 SRS 中的一个独立部分，或一独立的业务规范文档。某些业务规范将引出相应的功能需求；当然这些需求也应能追溯相应业务规范。

创建需求跟踪能力矩阵。建立一个矩阵把每项需求与实现、测试它的设计和代码部分联系起来。这样的需求跟踪能力矩阵同时也把功能需求和高层的需求及其它相关需求联系起来了。在开发过程中建立这个矩阵，而不要等到最后才去补建。

需求验证（requirement verification）

验证是为了确保需求说明准确、完整地表达了必要的质量特点。当你阅读软件需求规格说明（SRS）时，可能觉得需求是对的，但实现时，却很可能会出现问题。当以需求说明为依据编写测试用例时，你可能会发现说明中的二义性。而所有这些都必须改善，因为需求说明要作为设计和最终系统验证的依据。客户的参与在需求验证中占有重要的位置，第 14 章还将进一步讨论它。

审查需求文档。对需求文档进行正式审查是保证软件质量的很有效的方法。组织一个由不同代表（如分析人员，客户，设计人员，测试人员）组成的小组，对 SRS 及相关模型进行仔细的检查。另外在需求开发期间所做的非正式评审也是有所裨益的。

以需求为依据编写测试用例。根据用户需求所要求的产品特性写出黑盒功能测试用例。客户通过使用测试用例以确认是否达到了期望的要求。还要从测试用例追溯回功能需求以确保没有需求被疏忽，并且确信所有测试结果与测试用例相一致。同时，要使用测试用例来验证需求模型的正确性，如对话框图和原型等。

编写用户手册。在需求开发早期即可起草一份用户手册，用它作为需求规格说明的参考并辅助需求分析。优秀的用户手册要用浅显易懂的语言描述出所有对用户可见的功能。而辅助需求如质量属性、性能需求及对用户不可见的功能则在 SRS 中予以说明。

确定合格的标准。让用户描述什么样的产品才算满足他们的要求和适合他们使用的。将确认合格的测试建立在使用情景描述或使用实例的基础之上（ Hsia,Kung,和 Sell 1997）。

需求管理（requirement management）

当你完成需求说明之后，你不可避免的还会遇到项目需求的变更。有效的变更管理需要对变更带来的潜在影响及可能的成本费用进行评估。变更控制委员会与关键的项目风险承担者要进行协商，以确定哪些需求可以变更。同时，无论是在开发阶段还是在系统测试阶段，还应跟踪每项需求的状态。

建立起良好的配置管理方法是进行有效需求管理的先决条件。许多开发组织使用版本控制和其它管理配置技术来管理代码，所以你也可以采用这些方法来管理你的需求文档，需求管理的改进也是将全新的管理配置方法引入项目的组织中的一种方法。下列章节讨论了需求管理涉及到的各种技术：

- 第 16 章—建立需求基线和需求控制版本文档。
- 第 17 章—确定需求变更控制过程；建立变更控制委员会。
- 第 18 章—进行需求变更影响分析；跟踪所有受需求变更影响的工作产品。
- 第 19 章—使用需求管理工具。

确定需求变更控制过程。确定一个选择、分析和决策需求变更的过程。所有的需求变更都需遵循此过程，商业化的问题跟踪工具都能支持变更控制过程。

建立变更控制委员会。组织一个由项目风险承担者组成的小组作为变更控制委员会，由他们来确定进行哪些需求变更，此变更是否在项目范围内，估价它们，并对此评估作出决策以确定选择哪些，放弃哪些，并设置实现的优先顺序，制定目标版本。

进行需求变更影响分析。应评估每项选择的需求变更，以确定它对项目计划安排和其它需求的影响。明确与变更相关的任务并评估完成这些任务需要的工作量。通过这些分析将有助于变更控制委员会作出更好的决策。

跟踪所有受需求变更影响的工作产品。当进行某项需求变更时，参照需求跟踪能力矩阵找到相关的其它需求、设计模板、源代码和测试用例，这些相关部分可能也需要修改。这样能减少疏忽使需求变更带来的产品变更变得相对轻松些。

建立需求基线和需求控制版本文档。确定一个需求基线，之后的需求变更就遵循变更控制过程即可。每个版本的需求规格说明都必须是独立说明，以避免将底稿和基线或新旧版本相混淆。最好的办法是使用合适的配置管理工具在版本控制下为需求文档定位。

维护需求变更的历史记录。记录变更需求文档版本的日期以及所做的变更、原因，还包括由谁负责更新的和更新的新版本号等。版本控制工具能自动完成这些任务。

跟踪每项需求的状态。建立一个数据库，其中每一条记录保存一项功能需求。保存的每项功能需求的重要属性，它包括状态（如已推荐的，已通过的，已实施的，或已验证的），这样在任何时候都能得到每个状态类的需求数量。

衡量需求稳定性。记录基本需求的数量和每周或每月的变更数量（添加、修改、删除）。过多的需求变更“是一个报警信号”意味着问题并未真正弄清楚，项目范围并未很好的确定下来或是政策变化较大。

使用需求管理工具。商业化的需求管理工具能帮助你在数据库中存储不同类型的需求，为每项需求确定属性，可跟踪其状态，并在需求与其它软件开发工作产品间建立跟踪能力联系链。

项目管理（project management）

软件工程管理方法在本质上与项目的需求过程是紧密相关的。项目计划建立在功能基础之上，而需求变更会影响这些计划。因此，项目计划应能允许一定程度的需求变更或项目范围扩展。如果刚开始，需求不能确定，你可以选择一种软件开发方法生存期以允许这种不确定性，并在弄清后逐渐实施。关于需求工程项目管理方法的更详细内容将在以下章节讨论：

- 第 5 章——编写文档和管理与需求相关的风险。
- 第 15 章——基于需求的项目计划。
- 第 16 章——记录需求开发和管理中的工作。

选择一种合适的软件开发方法生存期。经典的瀑布法只适用于需求说明在项目早期即可全部完成的情况。你的组织应根据不同类型的项目和需求说明的不同程度选择几种不同的方法（McConnell 1996）。如果需求说明在项目早期无法全部确定，则从最为清晰易懂的需求开始，建立一个健壮可修改的结构再逐渐增加补充。实现了部分特性的产品可作为早期版本发布（Gilb 1998）。

基于需求的项目计划。随着需求细节不断变得清晰、完善，项目开发计划的进度安排将会不断改变。一开始可以根据项目视图和范围对开发功能需求所需的工作量作一估算，建立在不甚完善的需求基础之上的成本、进度安排的估计很不可靠，但随着需求说明的完善，估计也会得到不断改善。

发生需求变更时协商项目约定。当在项目中添加新的需求时，估计一下你是否能在目前安排下，利用现有资源保质保量完成。如果不能，将项目的实现与管理联系起来，协商一下新的，切实可行的约定（Humphrey 1997）。如果协商不成功，则交流可能的后果和更新的项目风险管理计划联系起来，以反映出对项目成功的新的不利因素。

编写文档和管理与需求相关的风险。采用“头脑风暴”方法并将与需求相关的项目风险编写成文档。利用各种方法来减轻或阻止这些风险，实施这些方法并跟踪其发展及效果。

跟踪需求工程所耗的工作量。记录需求开发和管理活动所花费的工作量。利用这些数据可以评估是否计划的需求活动已达到所期望的要求，并可以为将来项目的需求工程提供更好的所需资源的计划。

下一步

- 回到第 1 章的下一步中你已明确了与需求有关的一些问题。从本章中学到的推荐的实践方法，可能会有助于解决这些问题。针对建议的每一种方法，要在你组织中发现那些可能给其实现带来困难的障碍。
- 将在先前步骤中被确认是好的需求方法整理成一张表。针对每个方法，指明你的项目的能力水平：专家，熟练者，生手或新手。如果你的队伍中无一人熟练的话，就得要求项目的某个参与者学习更多的知识，并将他所学的教给队伍中的其他人。

第 4 章 改进需求过程 Top

第 3 章介绍了几十种需求工程中的好方法，你应当考虑在实践中应用它们。把理论方法付诸于实践活动是改进软件过程（process）的核心所在。从根本上说，改进过程包括使用更多有效的方法避免使用过去使用过的令人头痛的方法。然而，改进之路却是从失败、错误开始，还要历经那些受影响人的抵制及因任务时间紧迫导致搁置改进这样的挫折。

软件开发过程的改进有以下两个主要目标：

1．解决在以前项目中或目前项目中遇到的问题。

2．防止和避免你可能在将来的项目中要遇到的问题。

如果你目前采用的方法好象也挺有效的，你可能觉得没有必要改变你的方法。但是，即便是很成功的软件组织在面临大项目、不同的客户群、紧迫的进度安排或全新的应用领域时也会感到力不从心。因此，至少你应该知道其它一些很有价值也颇有效的需求工程方法，并把它们加入到你的软件工程中。

本章介绍了需求与其它主要的项目过程和风险承担者之间的联系。关于软件开发过程改进的一些基本概念，并推荐了一种经改进的生存期。我把一些重要的需求“过程精华”罗列出来以供参考使用。本章还介绍了实施改进需求工程实践的一个流程蓝图。

需求与其它项目过程的联系

需求是软件项目成功的核心所在，它为其它许多技术、管理活动奠定了基础。变更你的需求开发和管理方法将对其它项目过程产生影响，反过来也会产生影响。需求与其它过程的联系见图 4-1。下面简要介绍各过程间的接口。

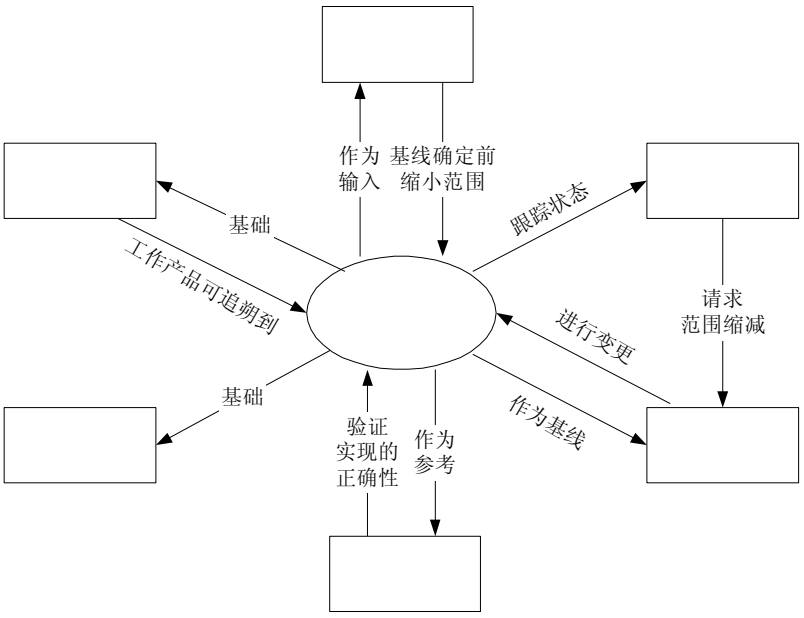

图4-1 需求与其它项目过程的关系

制定项目计划。需求是制定项目计划的基础。因为开发资源和进度安排的估计都要建立在对最终产品的真正理解之上。通常，项目计划指出所有希望的特性不可能在允许的资源和时间内完成，因此，需要缩小项目范围或采用版本计划对功能特性进行选择。

项目跟踪和控制。监控每项需求的状态，以便项目管理者能发现设计和验证是否达到预期的要求。如果没有达到，管理者通常请求变更控制过程来进行范围的缩减。

变更控制。在需求编写成文档并制定基线以后，所有接下来的变更都应通过确定的变更控制过程来进行。变更控制过程能确保：

- ✧ 变更的影响是可以接受的。
- ✧ 受到变更影响的所有人都接到通知并明白这一点。
- ✧ 由合适的人选来作出接受变更的正式决定。
- ✧ 资源按需进行调整。
- ✧ 保持需求文档是最新版本并是准确的更新文档。

系统测试。用户需求和功能需求是系统测试的重要参考。如果未说明清楚产品在多种多样条件下的期望行为，系统测试者将很难明确正确的测试内容。反过来说，系统测试是一种方法，可以验证计划中所列的功能是否按预期要求实现了。同时，也验证了用户任务是否能正确地执行。

用户编制文档。我曾在一个办公室里工作，办公室里有为商业产品准备用户文档的技术写作人员。我咨询其中一位写作人员为什么他们要工作那么长时间。“我们是食物链的终结者”她回答道,“我们要编写出用户显示界面及性能的最终变更版本 ”。产品的需求是编写文档的重要参考，低质量和拖延的需求会给编写用户文档带来极大的困难。

构造。软件项目主要产品是交付可执行软件，而不是需求说明文档。但需求文档是所有设计、实现工作的基础。要根据功能要求来确定设计模块，而模块又要作为编写代码的依据。采用设计评审的方法来确保设计正确地反映了所有的需求。而代码的单元测试能确定是否满足了设计规格说明和是否满足了相关的需求。跟踪每项需求与相应的设计和软件代码。

软件需求对其它项目风险承担者的影响

当软件开发队伍改变他们的需求过程时，与其它项目风险承担者沟通的接口也会发生变化。图 4-2 说明了一些外部组织功能，这些功能是通过一定的接口与软件开发队伍联系的，这些接口对项目需求活动起着重要作用。

为保持能顺利进行这些接口操作，要与其它领域的合作者多交流，让他们知道你的改进想法和调整计划。你要向他们说明改进后的新过程会带来什么好处。如在改进过程中需要获得合作时，可以从这样的谈话开始：“这些是我们曾经经历过的问题，而我们认为进行这些变更将会有助于问题的解决。这就是为什么我们要这样做的原因，我们需要得到你们的帮助。而我们的这些工作也会给你们予以帮助的”。反对变更是由于害怕变更带来的影响，因此要指明你进行的过程变更所可能带来的影响，从而减少大家的恐惧感。

向各个功能领域的人说明你从他们那里所需要获取的信息和帮助，从而有助于成功地开发整个产品。在开发过程中要遵从开发组与其它功能领域之间重要交流接口的规范和内容，如系统需求规格说明文档或市场需求文档。通常重要项目的文档从写作者角度是严格规范的，但往往不能给客户提供他们所真正需要的全部信息。

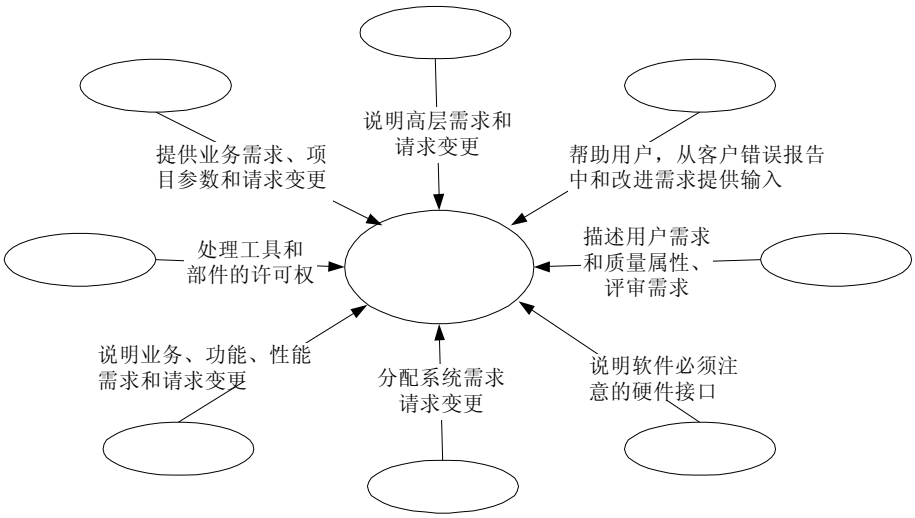

图4-2 软件开发组与其它组织间的重要需求接口

另一方面，询问其它组织需要从开发队伍中获取什么以有助于他们的工作。技术可行性方面哪些能帮助市场部更好地完成产品计划？什么样的需求状态报告能使管理者更充分地看清楚项目的进展情况？与系统工程部之间怎样的合作才能确保系统需求在软、硬件间的分配合理？努力在开发组和其它需求过程风险承担者之间建立合作关系以便所有人都能更有效地促进项目成功。

人们都不喜欢被迫离开他们已经习惯的环境，因此，你可能会在需求过程变更中将面临着抵制与反对。要做好思想准备尽量理解这些反对的缘由，以便你既能尊重它又能化解它。许多反对是由于不了解情况而引起的恐惧所造成的，因此一开始就要给他们说清楚为什么要作这些变更，变更后他们将受到怎样的影响，将会带来什么好处以及为什么你在过程改进一开始就需要他们的参与等等。下面是一些你可能遇到的抵制情况：

- ✧ 需求变更控制过程可能被看成变更是很难进行的一个障碍而被丢弃。而实际上，它提供了结构化和有条理的变更过程，并使得知道的人能作出更好的业务决定。你的任务是要确保变更过程真正能起作用。如果新的过程不能带来更好的结果，那大家将会“绕道而行”了。
- ✧ 一些开发人员把编写和审查需求文档看作是浪费时间的官僚做法，妨碍他们的“真正工作”——编写代码。如果你能向他们讲清一旦发生重写代码所带来的惨重代价，开发人员和管理人员将更能明白为什么需要做好需求工作。
- ✧ 如果客户支持的费用没有和开发过程联系起来，开发小组可能会缺少变更的动力，因为他们并不会因最终产品的低质量后果而带来损失。
- ✧ 如果改进后的需求过程的目标是通过创建高质量产品以减少技术支持费用，那么提供技术支持的管理者可能会感到要受到威胁。谁希望看到自己的帝国衰败呢？

软件过程改进的基础

阅读这本书可能是因为你想改变目前在需求工程中采用的一些方法。在为优秀的需求而努力工作时，请铭记下面四条改进软件的原则（Wiegers 1996a）：

1. **改进过程应该是革命性的、彻底的、连续的、反复的**。不要期望一次就能改进全部的过程，并且要能接受第一次尝试变更时，可能并没做好每一件事。不要奢求完美，要从某一些过程的改进、实施开始。当你有一些新技术的经验后，可逐渐调整你的方法。

2. **人们和组织机构都只有在他们获得激励时才愿意变更**。而变更引起的最强烈的刺激是痛苦。我的意思并不是要人为地制造痛苦（比如管理者强加的进度安排压力使开发人员工作异常痛苦），而是你曾在以前项目中经历过的真正艰辛。这些痛苦的激励作用远远超过管理者说："这本书告诉我们必须做这些新的事情，因此让我们开始吧！"下面是一些历史问题的例子，也许能为需求过程的变更提供驱动力：

- 项目没有时限，因为需求说明变得超过想象的复杂。
- 开发人员不得不大量超时工作，因为误解或二义性的需求直到开发后期才发现。
- 系统测试白费了，因为测试者并未明白产品要做什么。
- 功能都实现了，但由于产品的低性能、使用不方便或其它因素用户不满意。
- 维护费用相当高，因为客户的许多增强要求未在需求获取阶段提出。
- 开发组织落得交付一项客户并不想要的产品的名声，声誉受损。

3. **过程变更是面向目标的**。在开始运用高级过程之前，先确保你知道变更的目标。是想减少需求问题引起返工的工作量？还是想更好地控制需求变更？或是想在实施中不要遗漏某项需求？有一份明确规定的实施蓝图将会很有助于你在改进过程中取得成功。

4. **将改进活动看作为一些小项目**。许多改进活动一开始就失败了。因为缺乏计划或是因为所需资源并未给予。为避免这些问题，把每个改进行为看作一个项目。把改进所需的资源和任务纳入工程项目的总计划中。执行计划、跟踪、衡量和报告那些已在软件开发项目中所做的改进，缩减改进项目的规模。为每个过程改进领域写一份活动计划。跟踪风险承担者们执行计划的情况，看是否获得了预期的资源并知道改进过程实际消耗的费用。

过程改进周期

图 4-3 说明了一个软件过程改进的生存期。这种方法我曾经采用过，很有实效。这个周期反映出在活动前知道自己处于哪个阶段的重要性，为改进活动制定计划的必要性以及从自己经验中所学到的作为持续地过程改进的重要性。

评估当前采用的方法

任何过程改进活动的第一步都是评估当前在组织中使用的方法。找出其优势和缺陷所在。评估本身不能带来任何改进，但能提供信息，评估为你正确选择变更奠定了基础。

可以用不同的方法来评估当前过程。如果你已在尝试前面章节末尾的"下一步"。那你已经开始对你的需求方法及其结果在进行非正式的评估了。设计自我评价问卷是一种系统方法，它能以较低费用对当前过程进行评估。

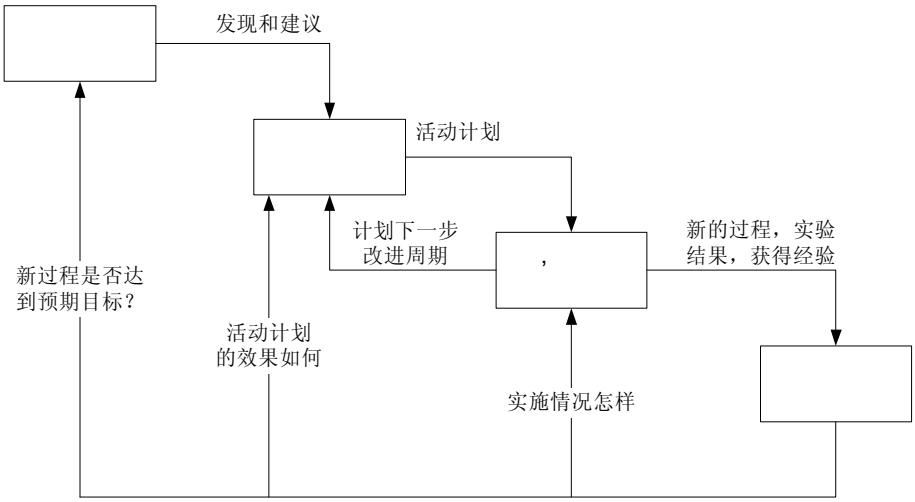

图4-3 软件开发过程改进的周期

一种更彻底的方法是让来自外部的顾问客观地评估你目前的软件开发方法。这种正式过程的评估方法要以一种已建立的过程改进框架工作为基础，如软件工程研究所（CMU/SEI 1995）开发的软件功能成熟度模型（CMM）。评估者将会检查软件开发和管理过程，而不限于需求活动。要根据你想通过过程改进取得的业务目标来选择评估方法，不要过多担心是否满足 CMM 或其它专用模型的需求。

本书附录包含了一个用于对目前需求方法的自我评估问卷表，可用这张问卷表来评估组织目前的需求工程方法。这样的自我评估将帮助你决定需求过程中的哪一个环节最需要进行改进。但仅仅根据在某项问题上的低分还不足以说明需要立即改进。要将注意力更多的放在对将来项目的成功带来最大风险和困难的领域。自我评价估中的每一个问题都是本书中某一章节的主要论题。Motorola 公司也开发了一套相似的“软件需求质量模型”用于软件需求过程评估（Smith 1998）。

正式的评估将获得一个列表——关于目前方法的长处和短处的说明——和关于改进机会的说明与推荐。不正式的评估，比如自我评估问卷，能使你了解并有助你选择改进领域。在本书的有关章节将发现许多针对自我评估问题的相应推荐方法。分析所考虑的每一项改进活动以确保它能在允许费用下实施。选择那些可能给投资带来相当回报的改进活动。

制定改进活动计划

遵从将过程改进活动看作是项目这一“哲学”，在评估后制定一个活动计划。考虑制订出描述组织整个软件过程改进初始工作的战略计划和在各个特定改进领域的战术行动计划，正如你收集需求时所采用的方法。每项战术行动计划应该指明改进行动的目标、风险承担者和一些必须完成的活动条目。如没有计划，则更容易疏忽比较重要的任务。计划也提供了跟踪过程的方法，使你能监控各活动条目完成情况。

图 4-4 举例说明了我经常使用的过程改进活动计划模板。在每一个活动计划中不要超过 10 个条目，这样使得计划简单易于取得早期成功。例如，我看到一个需求管理改进的计划包括如下活动条目：

1. 起草一个需求变更控制过程草案。
2. 评审并修改变更控制过程。
3. 以一个项目 A 来实验(pilot)变更控制过程。
4. 以实验反馈为基础修改变更控制过程。

5. 评估问题跟踪工具并选择其一来支持变更控制过程。

6. 购买问题跟踪工具并定制它来支持变更控制过程。

7. 在组织中使用新的变更控制过程和工具。

将每项活动条目交给专门的人来负责完成。不要让整个队伍小组作为活动条目的主人，队伍不会做工作，个人才会做。

如果你需要的活动条目多于 10 条，则先把注意力放在最重要的条目上，然后再处理其它条目。记住，变更是周期性反复的。本章之后介绍的过程改进蓝图将说明怎样才能把多个改进活动组成一个完整的软件开发过程改进计划。

需求过程改进的活动计划

项目： **日期：**

<项目名称> <编写计划的日期>

目标：

<成功执行这份计划后希望达到的一些目标。说明业务方面的目标，而不是过程变更方面的。>

成功度量：

<描述怎样确定过程变更是否达到了预期要求。>

组织受影响的范围：

<说明在本计划中所描述的过程变更带来影响的广度。>

人员和风险承担者：

<明确谁实施该计划，每个人的角色，及投入时间承诺（按小时/周或百分比为基础计算）。>

跟踪和报告过程：

<说明怎样跟踪计划中的活动条目进展情况，以及报告其状态结果等。>

依赖、风险和限制：

<明确对计划成功有帮助或有阻碍的各种外部因素。>

估计所有活动的完成日期：

<希望该计划什么时候完成？>

活动条目：

<为每个活动计划写出了 3-10 个活动条目。>

活动条目	负责人	截止日期	目标	活动描述	结果	所需资源
<顺序号>	<负责人>	<目标日期>	< 本活动条目标>	<实施活动条目要采取的行为>	<建立规程、模板或其它过程评估方法>	<各种所需的外部资源;包括物质材料、工具、文档或其他人员>

建立、实验和实施新的过程

到目前为止，已经对目前的需求方法进行了评估并起草了一份活动计划，指出了很可能带来收获的过程领域。现在进入较困难的一步：实施计划。许多过程改进在试图由计划付诸实践时，一开始便夭折了。

实施一项活动计划意味着开发新的、更好的方法，并且要相信它能提供一个比目前过程更好的结果。然而，并非第一次就能使新过程完美无缺。许多看起来很不错的方法付诸实施后会变得既不实用而又低效。因此，要为你建立的新过程或文档模板计划一个“实验”。运用在实验中获取的经验来调整新技术，这样当将它运用于整个目标群体时，改进活动会更有效果。请铭记下面这些关于引导实验的建议：

- 选择实验参与者（participant），他们将尝试新方法并提供反馈信息，这些参与者可以是生手也可以是老手，但他们应该对过程改进没有强烈的反对意思。
- 确定用于评估实验的标准，使得到的结果易于解释。
- 通知那些需要知道实验是什么以及为什么要实施的工程风险承担者。
- 考虑在不同的项目中实验新过程的不同部分。用这个方式可使更多的人尝试新方法，因此能提高意识，增加反馈信息。
- 作为评估的一部份工作，询问实验参与者，如果他们不得不回头采用他们原有的工作方法，他们觉得怎样。

即便是很有激励与善于理解的队伍，他们接纳变更的能力也是很有限的。所以不要一次给予项目或队伍太多的期望。编写出一整套实施计划，明确你将怎样把新方法运用于整个项目队伍以及你能提供的训练和支持；同时也要考虑管理者怎样阐明他们对新过程的期望。一种正式的关于需求工程和需求管理的文件通常要阐述清楚管理人员的任务和期望（CMU /SEI 1995）。

评估结果

过程改进周期的最后一步就是评估已实施的活动及取得的成果。这样的评估有助你在将来的改进活动中做得更好。评估实验工作进行得如何，采用新过程解决问题是否很有效。下一次在管理过程实验工作时是否需要稍作变更。

同时也要考虑整个新过程在群体中执行的情况。是否能使每个人都明白新过程或模板的好处？参与者是否理解并成功地应用了新过程？是否在下次工作中需要有所变更？

其中关键的一步是评估新实施的过程是否带来了期望的结果。尽管有一些新技术和管理方法都带来明显的改进，但更多的却需要时间来证明其全部的价值。例如，如果你实施一种新过程来处理需求变更，你就能很快看到项目变更以一种更规范的方式在进行。然而，一个新的软件需求规格说明 SRS 模板需要一段时间来证明其价值，因为分析人员和客户已习惯了一种需求文档的格式。给予新方法以足够的运行时间，选定能说明每项过程变更成功与否的衡量标准。

要接受学习曲线的事实。当从业者（practitionor）花费时间去吸收新方法时，生产率将会降低，如图 4-5 所示。这种短期的生产率降低是组织进行过程改进的一部份投入。如果你不理解这一点，你可能在得到回报之前就半途而废了，白白损失了投入而没有回报。对你的管理人员和同事进行有关学习曲线的教育，并使其明白：你采用高级的需求过程，将会获得更广泛的项目和业务回报。

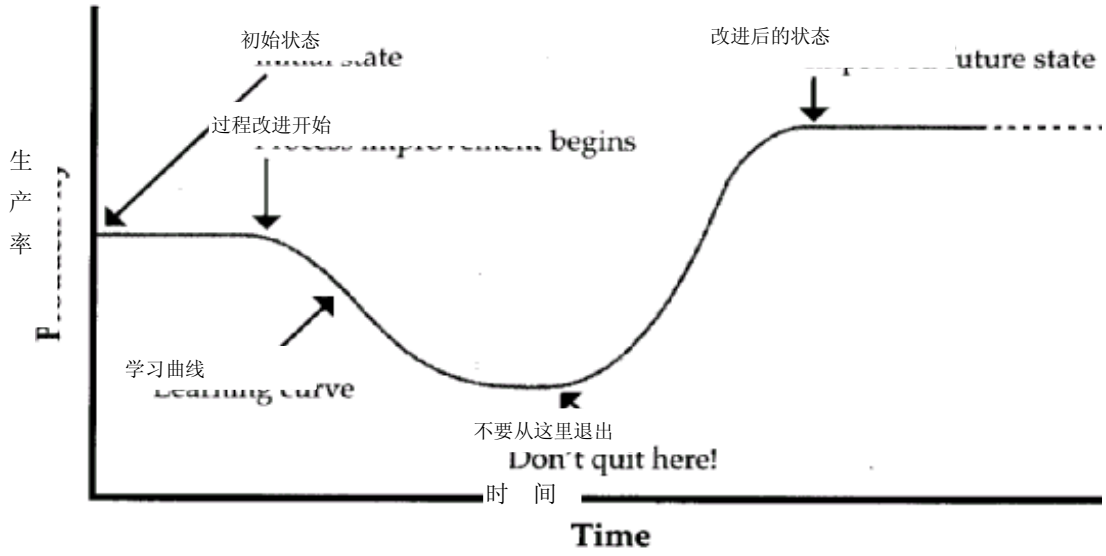

图 4-5 过程改进学习曲线

需求过程的积累材料（asset）

如果想要项目不断取得满意的结果，你需要有效地执行需求工程的各个过程：信息获取、分析、编写规格说明、验证以及管理。为了促进执行这些步骤，你应当把过程中积累的材料收集起来。过程包含已完成的活动和可交付的产品。过程中积累的材料有助于小组成员一致而有效地执行过程，还有助于大家理解他们遵从的步骤及要开发的产品。积累的材料包括下面几种类型的文档：

检查清单 清单列出各项活动，交付的结果和其它应注意或验证的条目。检查清单是用来提示记忆的，有助于确保处于忙碌中的工作人员不要忽略重要细节。

实例 一种特定类型工作产品的代表，积累起能在你组织中运用的更好的实例。

计划 概括说明怎样完成目标与完成时需要什么样的文档。

方针 确立活动期望、产品期望和交付产品期望的指导原则。过程都应遵从的方针。

过程 描述完成某个活动的任务顺序或步骤，说明要执行的任务及其在项目中所扮演的角色。不要包括示范信息。

过程描述 一组完成某些目的活动文档的定义。过程描述应包括过程目标、里程碑、参与者和执行任务的适合时间、交流步骤，期望结果以及与过程相关的输入和输出数据（Caputo 1998）。

模板 一种完成整个工作产品的指导方式。重要工程文档的模板提醒你检查是否遗漏了什么。一个结构很好的模板提供了许多捕获和组织信息的栏目(slot)。模板中包含的指导信息将帮助文档作者有效地使用它。

图 4-6 指出一些过程中应积累的材料，使需求开发和需求管理能在项目中更有效地进行，没有哪个软件过程规则书会说你必须拥有所有这些条目，但它们有助于你在整个需求方面的活动。

需求开发过程的积累材料	需求管理过程的积累材料
● 项目视图与范围模板	● 变更控制过程
● 需求开发过程	● 变更控制委员会过程
● 需求分配过程	● 需求变更影响分析检查清单和模板
● 使用实例模板	● 需求状态跟踪过程
● 软件需求规格说明模板	● 需求跟踪能力矩阵模板

图 4-6 需求开发和管理的重要过程的积累材料

图 4-6 列出的过程并不需要写在独立的文档中。例如，一个完整需求管理过程的描述可以包括变更控制过程、状态跟踪过程和影响分析清单。参见附录 J：CMM 实施向导（Caputo1998）。

下面是对图 4-6 中所列条目的简要说明，在相关的章节有详细介绍。请记住，每项工程都应调整组织的过程来满足其需要。

需求开发过程的积累材料

项目视图与范围模板。项目的视图与范围文档明确了项目的概念性功能，并提供了确定需求优先级和变更的参考。需求视图与范围文档是简明扼要的、高度概括的新项目业务需求说明。用统一的方式编写项目视图与范围文档有利于确保在项目进行过程中作决定时能考虑到所有应考虑的情况。第 6 章推荐了一份需求视图与范围文档的模板。

需求开发过程。该过程介绍了怎样确定客户及从客户那里获取需求的技术。也描述了项目 。需要创建的各种需求文档和分析模型。这个过程还指明了每项需求包含的信息种类，比如：优先级、预计的稳定性或计划发行版本号。同时还应指明需求分析及需求文档检验需要执行的步骤以及确认软件需求规格说明和建立需求基线的步骤。

需求分配过程。把高层的产品需求分成若干特定子系统是非常重要的，尤其是当开发的系统既含有软件又含有硬件或是包括多个子系统的软件产品时尤为重要（Nelsen 1990）。需求分配是在系统级需求完成和系统体系结构确定后才进行的，这个过程包含的信息是怎样执行分配以确保功能分配到合适的系统组件中，同时也说明分配的需求怎样才能追溯回它们的上两级系统需求以及在其它子系统中的相关需求。

使用实例模板。使用实例模板提供了一种把每项用户希望使用软件系统完成的任务编写成文档的标准方法。使用实例定义包括一个简要的任务介绍，必须处理的异常情况的说明和描述用户任务特点的附加信息。使用实例可作为软件需求规格说明中一条独立的功能需求。另外，你也可将使用实例与 SRS 模板合并为一个文档，既包括产品的使用实例，又包括软件功能需求，第 8 章介绍了一种使用实例模板。

软件需求规格说明模板。软件需求规格说明模板提供了一种组织功能需求和非功能需求的结构化方法。采用标准的 SRS 模板将有助于创建统一且高质量的需求文档。可能要采用多个模板以适应组织承担的不同类型和规模的项目。这样可减少因一种“万能”模板并不适合你的项目所带来的障碍。第 9 章介绍了一种 SRS 模板样例。

需求优先级确定过程。我的一个朋友 Matt 把软件项目的最后阶段称作“快速缩减范围阶段(rapid descoping phase)”，此时为满足进度时限要求，计划的功能不得不放弃掉。我们需要知道哪些项性能、使用实例或功能需求的优先级最低，以便在任何阶段，我们都可适当缩减范围。第 13 章介绍了一种优先级确定过程及一种工具，它能综合考虑需求对客户的价值、相应的技术风险及实施成本费用。

SRS 和使用实例审查清单。对需求文档的正式审查是软件质量保证的一项重要措施。审查清单指出在需求文档中发现的一些错误。在审查会议的准备中运用清单将使你的注意力集中到通常存在问题的地方。第 14 章包含了 SRS 和使用实例审查清单样例。

需求管理过程的积累材料

变更控制过程。变更控制过程能够减少因无休止、失控的需求变更引起的混乱。它明确了一种方法来提出、协商、评估一个新的需求或在已有需求上的一项变更。变更控制通常需要问题跟踪工具的支持，但请铭记工具并不能替代过程。第 17 章详细介绍了变更控制过程。

变更控制委员会过程。变更控制**委员会**(CCB)是由风险承担者的主要成员组成的，对提出的需求变更决定执行哪一项，拒绝哪一项，以及在各产品发行版本中包括哪些变更。CCB 过程描述了变更控制委员会的组成及操作过程。CCB 的主要活动是对提出的变更进行影响分析，为每项变更作出决定，并且告知那些将受到影响的人。第 17 章进一步讨论了 CCB 的组成及其功能。

需求变更影响分析清单和模板。估计提出的需求变更的成本费用和影响是决定是否执行变更的重要步骤。影响分析能帮助 CCB 作出正确的决定。如在第 18 章中说明的，影响分析清单包括许多自问自答型的问题，如：要考虑到可能的任务、边界影响、实施所确定的变更引起的相关的潜在风险。一张参与人员工作表可以作为估计任务工作量的简单方法，从这里就能明白确认变更的复杂性。第 18 章还提供了一个用于展示执行需求变更影响分析结果的模板样例。

需求状态跟踪过程。需求管理包括监控和报告每项功能需求的状态和状态改变的条件。你需要一个数据库或一种商业需求管理工具来跟踪一个复杂系统中大量的需求状态。此过程也描述了当你随时查看收集到的需求状态时输出的报告格式。如要获得关于需求状态跟踪方面更多的内容请参见第 16 章。

需求跟踪能力矩阵模板。需求跟踪能力矩阵列出了 SRS 中的所有功能需求及相应的设计模块，源文件和实施需求的过程，还有验证需求实施正确性的测试用例。跟踪能力矩阵应该也可以指出对应的上一层用户需求或系统需求。第 18 章将具体介绍需求跟踪能力。

需求过程改进路标（roadmap）

要知道改进你们组织的整个需求工程过程可不是件小事。毫无计划地进行改进很容易失败。所以，应当为实施改进需求过程开发一个路标。该路标应是软件开发过程改进战略计划的一部分内容。如果你已尝试进行前面介绍的评估方法，那么你已对采用的技术过程的优点及存在的缺陷有一定的了解。所以现在需要把这些改进活动排序以便能用最小的投资得到最多的收益。过程改进流程图描述了改进活动的一种前后次序。

因为有各种不同的情况，我不可能提供一种“万能”的路标。公式化的方法不能替代思考和常识，图 4-7 说明一个组织改进它的需求过程的路标。期望的业务结果写在图的右边方框里。主要的改进活动在其它方框里，一些中间的里程碑（圆圈）指明了取得的期望业务目标。从左到右实施每组改进活动。一旦已经建立了一个类似的路标，让一个人负责一个里程碑活动，他得为获得里程碑制定活动计划，然后把计划付诸行动！

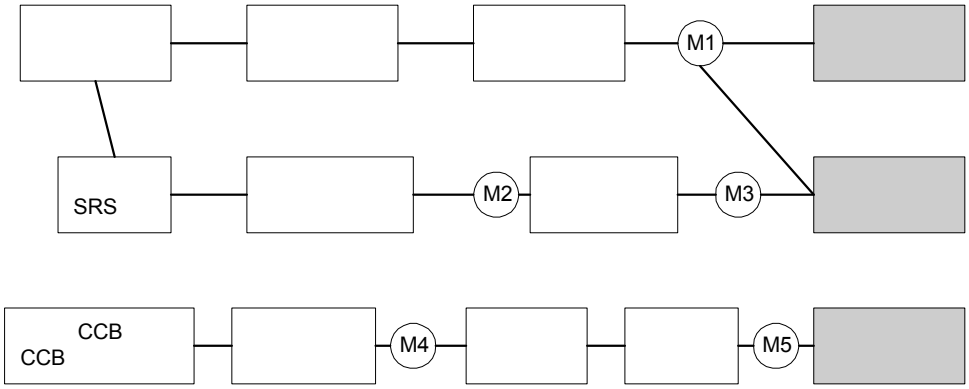

图4-7需求过程改进路标样例

下一步：

◆ 完成附录中的目前需求实践自我评估。以你目前实践缺陷的影响严重程度为基础，确定需求过程的三个最佳改进机会。

◆ 确定图 4-6 中列出的哪项需求工程的积累材料在你组织中还没有，但你认为会很有用。

◆ 在前面两步的基础上，建立一个需求过程改进路标。说服你组织的某个人来负责某项里程碑活动。让每位负责人写一份活动计划，用于实施活动，采用图 4-4 中介绍的活动计划模板。当实施计划时，跟踪各活动条目的进展情况。

第五章 软件需求与风险管理 Top

负责 Contoso 制药公司“化学制品跟踪系统”的项目管理人员 Dave 会见他的首席程序员 Helen 和首席测试员 Ramesh。他们对新项目都很有兴趣，但他们也记得在以前一个称作“药品仿真”的项目中遇到的问题。

“还记得我们直到进入测试时才发现用户对仿真程序的用户界面极为不满意吗？”Helen 问道。“我们花了五周时间重新实现，重新测试，我可再不愿玩这样的死亡游戏了。”

“的确是烦人，”Dave 附和道。“同样麻烦的是那些用户提出一大堆没人用过的特性，这样的交互导致编码花费了预计时间的三倍，我们是不管好歹，编完了事，简直是废品！”

“我们太匆忙了，以至没有时间写详细的需求说明”Ramesh 回忆道。“测试人员有一半的时间都在问程序员怎样才能判断他们的程序工作正常，以便能测试它。可是程序员设计的一些功能根本就不是用户所要求的。”

“我特别感到麻烦的是，要求开发药品仿真的管理者根本没有看需求规格说明就在上面签字确认了。”Dave 补充道：“于是我们不断遇到要求新的特性及各种变更，所以工程超期四个月，成本费用超出预算的一倍这也就不足为怪了。若再发生这样的事，我肯定会被解雇了。”

Ramesh 建议道：“也许我们应该把在仿真项目中遇到的问题一一列出来以便我们能在化学制品跟踪系统中避免重蹈覆辙。我看了篇关于软件风险管理的文章，上面介绍说我们应指出各种风险并说明了怎样才能避免它们”。

“我可不那样想”Dave 坚持道：“我们已从仿真项目学到了不少，我们不会再有那些问题了。这个项目还没有达到需要用风险管理的地步。如果要把我们可能犯的错误都写下来，好像我连怎样做软件项目都不知道似的。我不想要任何消极想法影响项目。我们必须为成功而制定计划。”

正如 Dave 的最后一句话所反映的那样，软件工程师都是绝对的乐观主义者。我们总是希望我们的下一个项目进行顺利，而不管以前项目发生的问题。带来的事实却是许多潜在威胁阻碍项目按计划进行。与 Dave 的想法恰恰相反的是，软件项目管理者必须要明确和控制他们的项目风险，并且是要从需求工程的风险开始进行的。

所谓风险是可能给项目的成功带来威胁或损失的情况。这种情况还并没有发生以带来问题，而你希望它永远不会发生。但这些潜在的问题可能会给项目成本费用、进度安排，或技术方面、产品质量、团队工作效率等带来较大的负面影响。而风险管理——一种软件工业的最佳方法——就是在风险给项目带来损失之前，就指明、评估并加以控制风险。如果不期望的事已经发生了，那就不再是风险，而是事实了。那只好通过项目事务(ongoing)状态跟踪和校正过程来处理当前的问题。

正如没有人能确切地预测未来，风险管理也仅是让你采取一些措施尽可能减少潜在问题发生的可能性或减少其带来的影响。风险管理的意思是在一种担忧转变为危机或实际困难之前处理它。这将提高项目成功的可能性且可减少不可避免的风险会造成的损失。对处于个人控制领域之外的风险应由相应层次的管理者来负责。

由于需求说明在软件项目中扮演着一个核心的角色，故精明的项目管理者会在初期就指明与需求相关的风险并积极地控制它们。典型的需求风险包括对需求的误解，不恰当的用户参与，不确定或随意变更项目的范围和目标，以及持续变更需求。项目管理者只能通过与客户、或与客户代表（如市场人员）的合作来控制需求风险。合作编写需求风险文档，共同制定减轻风险的措施，增强客户与开发人员之间的合作伙伴关系，这在第 2 章中已作介绍了。

不仔细研究是不能把风险撵走的，因此本章对需求风险管理进行简略的介绍。本章后面还会提到需求工程中常出现的一些风险因素。运用这些信息可以使你在风险攻击项目前处理风险。

软件风险管理基础 Top

除了与项目范围和需求有关的风险外，项目还面临着许多风险。依赖于外界实体，例如一个转包承揽者或生产重用部件的另一个项目就是一种常见的风险来源。项目管理一直面临各种风险挑战：不准确的估计，对准确估计的否决，对项目状态不清楚，资金的周转。技术风险威胁着高度复杂或很前沿(reading-dge)的开发项目，缺乏知识是另一种风险源，以及参与者对所用的技术和应用领域很陌生等等。强制的或总是变更的政府规范会使一个很好的计划彻底作废。

很可怕吧？这就是为什么所有的项目都应该认真地进行风险管理的原因，风险管理是不断察看水平线上是否出现了冰山，而不是以充足的信心认为船不会沉就以全速挺进。注意同其他过程一样，让你的风险管理活动与工程规模相适应。小规模的工程可以只列出一张简单的风险清单，但对于一个大规模项目的成功，正式的风险管理计划则显得非常重要。

风险管理的要素

风险管理就是使用某些工具和步骤把项目风险限制在一个可接受的范围内。风险管理提供了一种标准的方法来指出并把风险因素编成文档，评估其潜在的威胁，以及减少这些风险的建议战略（Williams，Walker， and Dorofee 1997）。风险管理包括的活动如图 5-1 所示：

风险评价（risk assessment）是一个检查工程项目并识别潜在风险区域的过程。可以通过列举通常的

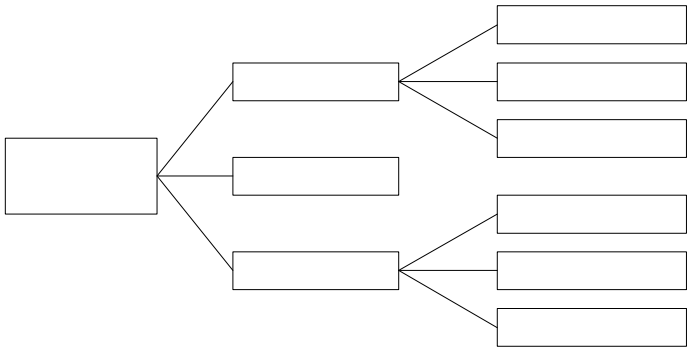

图5-1　风险管理要素

软件项目风险因素，如需求风险因素的办法来使风险识别（risk identification）更加方便容易。在本章后面描述了一些需求风险因素（Carretal 1993；McConnell 1996）。在风险分析中，应检查一些特定风险对项目可能造成的潜在后果。风险分级（risk prioritization）有助你通过评价每项风险的潜在危害值，从而更注重处理最严重的风险。风险危害值(risk exposure)包括带来损失的可能性大小和潜在损失的规模。

风险避免（risk avoidance）是处理风险的一种方法：尽量别作冒险的事。如果你不承担任何项目，如果你采用成熟的而并非处于研究阶段的技术，或者你将难以实现的特性都排除在项目之外你就可以避开风险。

但更常见的是，需要采取风险控制（risk control）的方法来管理那些已被发现为高优先级的风险。制定风险管理计划是一项处理具有重大意义风险的计划，包括降低风险的方法、应急计划、负责人和截止日期。你应尽量避免让风险成为真正的问题，或即便问题发生了，也应尽量让其影响降低到最小。风险不能够自我控制，所以风险解决方案就包括了降低、减少每项风险的执行计划。最后，通过风险监控

（risk monitoring）来跟踪风险解决过程的进展情况。这也是例外的项目状态跟踪的一部分内容。监控可以很好了解降低风险工作的进展情况，可以定期地修订先前风险清单的内容和划分的优先级。

编写项目风险文档 Top

仅仅认识到项目面临的风险是远远不够的。应该将其编写成文档并妥善进行管理，这样有利于在整个项目开发过程中风险承担者了解风险情况和状态。图 5-2 提供了一个编写一个条目（item）风险说明的模板。你可能觉得以表格形式存放在电子表格中更加方便，因为那样更易于把各项风险进行排序（列表）。但将风险列表(清单)组成一独立文档以便在整个项目进行中升级和维护。

风险条目跟踪模板

序列号：

<顺序号>

确定日期：

<风险被识别出的日期>

撤消日期：

<撤消风险确定日期>

描述：

<以“条件-结果”的形式描述风险>

可能性：

<风险转变为问题的可能性>

影响：

<如果风险变成了事实将造成的损失>

危害值：

<可能性×影响>

降低风险计划：

<一种或多种用来控制、避免、最小化及降低风险的方法>

负责人：

<解决风险的责任承担者>

截止日期：

<完成降低风险措施的截止日期>

在编写风险说明时，最好采用条件——结果的形式。也就是，先说明你关心的条件，接着是潜在的有害结果（如果风险成为事实）。有时，人们只说明了风险条件（如“客户不同意产品的需求说明”）或者只说明了结果（“我们只能满足某些主要的客户”）。最好将这样的说明句子合并成条件——结果形式的结构：“如果有些客户不赞同产品的需求说明，那我们只能满足某些主要客户的意见。”而一个条件下可能有多个结果，同时也可能出现多个条件下导致同一个结果。

模板能记录风险变为事实的可能性及对项目的消极影响，还有整个的风险危害值（可能性×影响）。我用 0.1（极不可能）到 1.0（肯定发生）来描述可能性，用 1（无甚么影响）到 10（有很深、很大的影响）来表示影响。将这两个因素相乘即可作为评估风险危害值的依据。

不要试图精确量化风险。你的目标是将最有威胁的风险和那些不急需处理的风险区别开来。大家可能更愿意用高、中和低来估计可能性及影响。但风险条目中至少应有一个为高的风险。

制定降低风险计划来明确控制风险要采取的活动，其中一些策略是尽量降低风险发生的可能性；而另一些则是减少风险发生后带来的影响。做计划时要考虑降低风险所耗费用，千万别花费 20，000 美元来控制一项仅会损失 10，000 美元的风险。为每项风险安排一个负责人，并确定完成活动的截止日期。长期或复杂的风险可能需要具有多个阶段性成果的多步骤降低风险策略计划。

图 5-3 说明了本章开始部分介绍的“化学制品跟踪系统”小组领导者讨论的一个风险。小组凭他们以前的经验估计了风险的可能性及其影响。除非他们把其它风险因素也估计出来，否则他们并不明白风险危害值 4.2 究竟有多严重。降低风险措施的前两条是通过更多的用户参与项目来减少风险发生的可能性。而采用原型法则可以利用用户关于界面的早期反馈来减少风险的潜在影响。

制定风险管理计划

一张风险列表还不等于一个风险管理计划。对于一个小项目，你可以把控制风险的计划放在软件项目管理计划里。但一个大项目则需要一份独立的风险管理计划，包括用于识别、评估、编写、跟踪风险的各种方法与途径。这份计划还应包括风险管理活动的角色和责任。你可能希望专门让一个项目风险管理人员负责可能引起麻烦的事。

通常，项目小组为他们的关键活动制定了计划，却在项目中没有按计划去实施或者未能按实际情况进行及时的调整。要坚持按照所采取的风险管理活动计划去执行。项目的进度安排上也应给风险管理留出足够时间来确保项目并未浪费早期投资在风险计划制定上。工程项目的工作分类细目结构中包括降低风险的活动、状态报告，以及更新风险清单。

和其它项目管理活动一样，你需要建立起周期性的监控措施。保持对十来个有最大危害的风险的高度重视，并追踪降低风险措施的有效性。当完成一项活动后，重新评估那项风险的可能性和影响，更新风险清单和其它相关的计划。当控制住原本有很高优先级的风险后，必然有新条目会进入前十条。请记住，不要仅仅因为完成了一项降低风险的活动而人为去增加一条风险来进行控制。你应当想想你降低风险的方法是否的确减少了风险的危害，使其减少到了一个可以接受的水平。

化学制品跟踪系统的风险条目样例

序列号：

1

确定日期：

5/4/99

撤消日期：

描述：

需求获取中无合适用户参与,导致测试之后用户界面的返工.

可能性：

0.6

影响：

7

危害值：

4.2

降低风险计划：

1．在第一阶段早期就要收集易学、易用的需求。

2．与产品代表一起召开 JAD 会议以开发需求。

3．通过与产品代表和顾问的交流，开发一个包含核心功能的用户界面原型。让产品代表和其他用户来评估此原型。

负责人：

Helen

截止日期：

在 6/16/99 前完成 JAD 会议。

图 5-3 化学制品跟踪系统的风险条目样例

与需求有关的风险

下面几页介绍的风险因素是按需求工程中获取、分析、编写规格说明、验证和管理汇总起来的，并推荐了一些方法用于降低风险发生的可能性或减轻风险发生给项目带来的影响。这张清单仅仅是一个起点，在你做项目逐渐积累经验过程中，加入你的风险因素清单和减轻风险的策略。使用这里提供的条目来帮助你识别需求风险并采用条件——结果的格式来书写风险说明。

需求获取

产品视图与范围。如果团队成员没有对他们要做的产品功能达成一个清晰的共识，则很可能导致项目范围的逐渐扩大。因此最好在项目早期写一份项目视图与范围将业务需求涵盖在内，并将其作为新的需求及修改需求的指导。

需求开发所需时间。紧张的工程进度安排给管理者造成很大的压力，使他们觉得不赶紧开始编码将无法按时完成项目，因而对需求一带而过。项目因其规模和应用种类不同（如信息系统，系统软件，商业的或军事的应用）而有着很大的不同。粗略的统计表明：需求开发工作应占全部工作量的 15%（Rubin

1999）。记录你参与的每个项目中实际需求开发的工作量，这样就能知道所花的时间是否合适并改进将来项目的工作计划。

需求规格说明的完整性和正确性。为确保需求是客户真正需要的，要以用户的任务为中心，应用使用实例技术获取需求。根据不同的使用情景编写需求测试用例，建立原型，使需求对用户来说更加直观，同时获取用户的反馈信息。让客户代表对需求规格说明和分析模型进行正式的评审。

对革新产品的需求。有时容易忽略市场对产品的反馈信息。故要强调市场调查研究，建立原型，并运用客户核心小组来获得革新产品任务的反馈信息。

明确非功能需求。由于一般强调产品的功能性要求，非常容易忽略产品的非功能性的需求。询问客户关于产品性能、使用性、完整性、可靠性等质量特性，编写非功能需求文档和验收标准，（像在 SRS 中一样）作为可接受的标准。

客户赞同产品需求。如果不同的客户对产品有不同的意见，那最后必将有些客户会不满意。确定出主要的客户，并采用产品代表的方法来确保客户代表的积极参与，确保在需求决定权上有正确的人选。

未加说明的需求。客户可能会有一些隐含的期望要求，但并未说明。要尽量识别并记录这些假设。提出大量的问题来提示客户以充分表达他们的想法、主意和应关注的一切。

把已有的产品作为需求基线。在升级或重做的项目中需求开发可能显得不很重要。开发人员有时被迫把已有的产品作为需求说明的来源。“只是修改一些错误和增加一些新特性”，这时的开发人员不得不通过现有产品的逆向工程(reverse engineering)来获取需求。可是，逆向工程对收集需求是一种既不充分也不完整的方法。因此新系统很可能会有一些与现有系统同样的缺陷。将在逆向工程中收集的需求编写成文档，并让客户评审以确保其正确性。

给出期望的解决办法。用户推荐的解决方法往往掩盖了用户的实际需求，导致业务处理的低效，或者给开发人员带来压力以至做出很差的设计方案。因此分析人员应尽力从客户叙说的解决方法中提炼出其本质核心。

需求分析

划分需求优先级。划分出每项需求、特性或使用实例的优先级并安排在特定的产品版本或实现步骤中。评估每项新需求的优先级并与已有余下的工作主体相对比以做出适宜的决策。

带来技术困难的特性。分析每项需求的可行性以确定是否能按计划实现。成功好象总是悬于一线的，于是运用项目状态跟踪的办法管理那些落后于计划安排的需求，并尽早采取纠正措施。

不熟悉的技术、方法、语言、工具或硬件平台。不要低估了学习曲线中表明的满足某项需求所需要的新技术的速度跟进情况。明确那些高风险的需求并允许一段充裕时间用来从错误开始学习、实验及原型测试。

需求规格说明

需求理解。开发人员和客户对需求的不同理解会带来彼此间的期望差异，将导致最终产品无法满足客户的要求。对需求文档进行正式评审的团队应包括开发人员，测试人员和客户。训练有素且颇有经验的需求分析人员能通过询问客户一些合适的问题，从而写出更好的规格说明。模型和原型能从不同角度说明需求，这样可解决一些模糊的需求。

时间压力对 TBD 的影响。将 SRS 中需要将来进一步解决的需求注上 TBD 记号，但如果这些 TBD 并未解决，则将给结构设计与项目继续进行带来很大风险。因此应记录解决每项 TBD 的负责人的名字，如何解决的以及解决的截止日期。

具有二义性的术语。建立一本术语和数据字典，用于定义所有的业务和技术词汇，以防止它被不同的读者理解为不同的意思。特别是要说明清楚那些既有普通含义又有专用领域含义的词语。对 SRS 的评审能够帮助参与者对关键术语、概念等达成一致的共识。

需求说明中包括了设计。包含在 SRS 中的设计方法将对开发人员造成多余的限制并妨碍他们进行最佳设计的创造性。仔细评审需求说明以确保它是在强调解决业务问题需要做什么，而不是在说怎么做。

需求验证

未经验证的需求。审查相当篇幅的 SRS 这是有些令人沮丧的，这正如要在开发过程早期编写测试用例一样。但如果在构造设计开始之前通过验证基于需求的测试计划和原型测试来验证需求的正确性及其质量，就能大大减少项目后期的返工现象。在项目计划中应为这些保证质量的活动预留时间并提供资源。从客户代表方获得参与需求评审的赞同（承诺），并尽早且以尽可能低的成本通过非正式的评审逐渐到正式评审来找出其存在的问题。

审查的有效性。如果评审人员不懂得怎样正确地评审需求文档和怎样做到有效评审，那么很可能会遗留一些严重的不足之处。故要对参与需求文档评审的所有团队成员进行培训，请组织内部有经验的评审专家或外界的咨询顾问来讲座、授教以帮助使评审工作更加有效。

需求管理

变更需求。将项目视图与范围文档作为变更的参照基础可以减少项目范围的延伸。用户积极参与的具有良好合作精神的需求获取过程可把需求变更减少近一半（Jones 1996a）。能在早期发现需求错误的质量控制方法可以减少以后发生变更的可能。而为了减少需求变更的影响，将那些易于变更的需求用多种方案实现，并在设计时更要注重其可修改性。

需求变更过程。需求变更的风险来源于未曾明确的变更过程或采用的变动机制无效或不按计划的过程来做出变更。应当在开发的各阶层都建立变更管理的纪律和氛围，当然这需要时间。需求变更过程包括对变更的影响评估，提供决策的变更控制委员会，以及支持确定重要起点步骤的工具。

未实现的需求。需求跟踪能力矩阵有助于避免在设计，结构建立及测试期间遗漏的任何需求。也有助于确保不会因为交流不充分而导致多个开发人员都未实现某项需求。

扩充项目范围。如果开始未很好定义需求，那么很可能隔段时间就要扩充项目的范围。产品中未说明白的地方将耗费比预料更多的工作量，而且按最初需求所分配好的项目资源也可能不按实际更改后用户的需求而调整。为减少这些风险，要对阶段递增式的生存期制定计划，在早期版本中实现核心功能，并在以后的阶段中逐步增加实现需求。

风险管理是你的好助手

项目管理人员可以运用风险管理来提高对造成项目损失的条件的警惕，在需求获取阶段要有用户的积极参与。精明的管理者不仅能认识到它能带来风险的条件，而且将它编入风险清单，并依据以往项目的经验估计其可能性和影响。如果用户一直没有参与，风险危害值将会扩大以至危害项目的成功。我曾说服管理人员把项目延期是由于缺少用户的积极参与，我告诉他们不能把公司的资金投入一项注定要失败的项目。

周期性的风险跟踪能使管理人员保持对风险危害变化的了解，对那些并未得到完全控制的风险能得到高层管理人员的注意。他们要么开始采取一些矫正措施，要么不管风险，依旧按原业务决策思路进行。即使不能控制项目可能遇到的所有风险，但风险管理能使你看清形势，做出的决策是有依据的。

下一步：

- 明确你当前项目面临的一些与需求有关的风险，不要把当前的问题当作风险，一定要是那些还未发生的事情。将风险因素用条件——结果形式编写成文档， 正如图 5-2 模板所示的那样。为每项风险推荐至少一种可能的降低风险的方法。
- 召集代表开发、市场、客户和管理各方面的风险承担者召开风险“集体研讨”会议。尽力找出更多与需求有关的风险因素。估计每项风险发生的可能性及其影响 ，并将两者乘积起来得到风险危害值。通过按风险危害值降序排列找到最高的五项风险。为每项风险安排一个负责人负责实施降低风险的活动。